

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MODEL EVOLUTION FOR THE REALIZATION OF COMPLEX SYSTEMS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

LIN GUO

Norman, Oklahoma

2021

MODEL EVOLUTION FOR THE REALIZATION OF COMPLEX SYSTEMS

A DISSERTATION APPROVED FOR THE  
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Janet K. Allen, Co-chair

Dr. Farrokh Mistree, Co-chair

Dr. Theodore B. Trafalis

Dr. Charles D. Nicholson

Dr. Thomas M. Neeson

© Copyright by LIN GUO 2021  
All Rights Reserved.

## **ACKNOWLEDGMENTS**

I dedicate this dissertation to my family, who gives me the greatest love and support – my beloved husband, mother, father, mother-in-law, and two daughters.

I dedicate the monograph based on this dissertation to my advisors, academic parents, and co-chairs of my committee, Professor Janet K. Allen and Professor Farrokh Mistree. I sincerely appreciate their excellent mentoring, great patience, continuous encouragement and care, spiritual and financial support, collaboration and networking opportunities across disciplines and countries, and everlasting passion. Thanks to their advice – incorporating my defense slides as Appendix E.

I thank my committee members, Dr. Theodore B. Trafalis, Dr. Charles D. Nicholson, and Dr. Thomas M. Neeson, for their valuable advice and suggestions during my Ph.D. research.

I deeply appreciate the help and recognition that I receive from Dr. Shivakumar Raman, and all the other great professors in the School of Industrial and Systems Engineering at the University of Oklahoma.

I am truly grateful for the education and financial support that I received from the University of Oklahoma. I especially appreciate my advisors' chair fund, L.A. Comp Chair and the John and Mary Moore Chair at the University of Oklahoma.

I acknowledge all my academic siblings, mentors, collaborators, and friends, for the incredible guidance, advice, help, faith, care, and opportunities they offer – Dr. Hamedzamani Sabzi, Dr. Thomas M. Neeson, Suhao Chen, Dr. Warren F. Smith, Dr. Anand Balu Nellippallil, Dr. Ashok Das, Ayushi Sharma, Dr. Shima Mohebbi, Dr. Ru Wang, Dr. Guoxin Wang, Dr. Jelena Milisavljevic-Syed, Yu Huang, Jacob G. Starks, Nathan B. Preuss, Shehnaz B. Shaik, Abigna A.

Reddy, Dr. Zhenjun Ming, Shan Peng, Xiao Shi, Chuanhao Li, Abhishek Yadav, Vishnu Kamala, Gehendra Sharma, Sara Hajhashemi, Xiwen Shang, Shuting Chen, Liangyue Jia, Yilin Jiang, Yafen Chen, and Dr. Yu Liu.

I am gratified to all the co-workers, staff, coordinators, and helpers that help me proceed and smooth my Ph.D. journey, from ISE, AME, Gallogly College of Engineering, Graduate College, GSC, Human Resource, International Student Service, Goddard, OUIT, etc.

I thank the education and all the amazing things that I received from my home city Shijiazhuang, my graduate university Shanghai Jiao Tong University, and my former mentors, employers, and colleagues.

I am thoroughly thankful to Norman Public School, KinderCare, Norman Chinese School, Cleveland County Department of Human Service, and Sooner Care, who give me financial and emotional support by helping educate and take care of my daughters. I acknowledge my neighbors Laifeng Qiu and Lily and James Dixson for involving us in a happy and healthy community and raising children together with us, so I can stay focused on my research.

I feel gratitude for the people and events that occur in my life that make me feel struggling, self-doubting, and wanting to get out of my comfort zone. Those critics, disapprovals, and neglect force me to improve and upgrade myself continuously.

I am extremely grateful to the people who read any part of this dissertation or my other publications. This motivates me significantly to be continuously productive. Thank you so much for your time and interest. This dissertation is the start of my academic career. More interesting works will come. Please stay tuned. I suggest readers start with Appendix E.

# TABLE OF CONTENTS

Acknowledgments .....	iv
List of Tables .....	xvi
List of Figures.....	xxi
Abstract.....	xxx
Chapter 1 Frame of Reference: Designing Complex Systems using Satisficing Strategy .....	1
1.1 What are Complex Systems?.....	3
1.1.1 Characteristics of Complex Systems .....	6
1.1.2 Examples of Complex Systems.....	9
1.2 Modeling Strategies and Their Foci .....	12
1.2.1 Optimizing Strategy and Satisficing Strategy.....	14
1.2.2 Why is Satisficing Desired in Engineering Design?.....	15
1.3 The Challenges in the Model-based Realization of Complex Systems.....	19
1.3.1 Models are Approximations of the Real World .....	20
1.3.2 The Purpose of Model Evolution .....	21
1.4 Problem Statement – Problems in both Strategies .....	21
1.4.1 Problems in Optimizing Strategy.....	22
1.4.2 Problems in Satisficing Strategy.....	24
1.5 Research Gaps and Hypotheses.....	26
1.5.1 Research Gaps .....	26
1.5.2 Hypotheses to Bridge the Research Gaps.....	27
1.5.3 Expected Contribution by Testifying the Hypotheses .....	28

1.6 Plan of Verification and Validation.....	30
1.7 Organization of The Dissertation .....	31
1.8 Role of Chapter 1 in this Dissertation .....	32
Chapter 2 Research Questions: How Can We Realize Model Evolution.....	34
2.1 Using Kuhn-Tucker Conditions to Explain Optimal and <i>Satisfice</i> .....	35
2.1.1 <i>The History of the Kuhn-Tucker Conditions</i> .....	35
2.1.2 <i>Necessary and Sufficient Kuhn-Tucker Conditions</i> .....	36
2.1.3 <i>The Physical Meaning of the Kuhn-Tucker Conditions</i> .....	37
2.1.4 <i>Assumptions behind Kuhn-Tucker Conditions</i> .....	39
2.2 Advantages of <i>Satisficing</i> Strategy.....	41
2.2.1 <i>Possible Features of Engineering-Design Problems</i> .....	43
2.2.2 <i>Toy Problem *I* (TP-I)</i> .....	46
2.2.3 <i>Method Requirement 1: Combination of Interior-Point Searching and Vertex Searching</i> .....	50
2.2.4 <i>Toy Problem II (TP-II)</i> .....	50
2.2.5 <i>Method Requirement 2&amp;3: Second-Order Sequential Linearization and Accumulated Linearization</i> .....	55
2.2.6 <i>Toy Problem III (TP-III)</i> .....	58
2.2.7 <i>Method Requirement 4: Using Goals and Minimizing Deviation Variables Instead of Objectives</i> .....	62
2.2.8 <i>Toy Problem IV (TP-IV)</i> .....	67
2.2.9 <i>Method Requirement 5: Allowing Some Violations of Soft Requirements, such as the Bounds of Deviation Variables</i> .....	72

2.2.10 Toy Problem V (TP-V).....	73
2.3 Summary of Differences between Optimizing and <i>Satisficing</i> Strategy .....	79
2.3.1 Differences between Optimizing and <i>Satisficing</i> Strategy.....	79
2.3.2 Summary of Differences among cDSP, Goal Programming, and Mathematical Programming.....	81
2.4 Research Questions (RQ1-RQ4) .....	85
2.4.1 Justification of the Primary Research Question regarding Requirements.....	85
2.4.2 Justified Research Questions regarding Tasks.....	87
2.5 Specification of Hypotheses (SH1-SH4).....	90
2.6 Role of Chapter 2 in this Dissertation .....	91
Chapter 3 Proposed Methods – The Design Evolution Loop.....	94
3.1 Elements of Design Improvement through Model Evolution (Task 1-4).....	98
3.1.1 Task 1: Formulation-Exploration.....	98
3.1.2 Task 2: Approximation-Exploration-Evaluation .....	99
3.1.3 Task 3: Formulation-Exploration-Evaluation.....	101
3.1.4 Task 4: Formulation-Approximation-Exploration-Evaluation .....	103
3.1.5 Model Evolution Cycle is an Open and Extendable Framework .....	104
3.2 Theoretically Verification of the Feasibility of the Specified Hypotheses (TVe1-TVe4) 105	
3.2.1 Theoretical Verification of Specified Hypothesis 1 (TVe1) .....	105
3.2.2 Theoretical Verification of Specified Hypothesis 2 (TVe2) .....	106
3.2.3 Theoretical Verification of Specified Hypothesis 3 (TVe3) .....	108
3.2.4 Theoretical Verification of Specified Hypothesis 4 (TVe4) .....	109
3.3 Overview of Proposed Methods (M1-M4).....	111



3.3.1 M1: Exploration of the Boundary using Formulation-Exploration Framework.....	111
3.3.2 M2: Improving Algorithm Robustness using Parameter Learning .....	114
3.3.3 M3: Exploring Interrelationships among Subsystems using Unsupervised Learning	117
3.3.4 M4: Exploring Critical Factors through Scenario Planning in Agent-Based Modeling .....	120
3.4 Overview of Test Problems .....	122
3.4.1 Required Characteristics of the Test Problems .....	122
3.4.2 Brief Introduction of Each Test Problem .....	127
3.5 Role of Chapter 3 in this Dissertation .....	130
Chapter 4 Type I & II Robust Design through Formulation-Exploration Framework.....	131
4.1 Frame of References on Satisficing Strategy .....	134
4.2 Managing Conflicting Goals and Uncertainties in a Dam Network.....	136
4.2.1 Problem Statement – Test Problem 1.1: Dam-Network Planning.....	138
4.2.2 Critical Review of The Literature on Dam-Network Water Resource Management..	146
4.2.3 Proposed Methods – The Three-Step Exploration Method.....	151
4.2.4 Formulation of Compromise DSP (cDSP) .....	157
4.2.5 Water Resource Planning Results and Discussion.....	160
4.2.6 Closure of Test Problem I.....	173
4.3 Positioning the Customer Order Decoupling Point of a Supply Chain.....	175
4.3.1 Problem Statement – Test Problem 1.2: CODP and the Challenges in Supply Chains .....	176
4.3.2 Literature Gap Analysis – in the Domain of Customer Order Decoupling Point Determination.....	182

4.3.3 Proposed Methods – The Formulation-Exploration Framework.....	186
4.3.4 Model Formulation.....	189
4.3.5 CODP Results and Discussion .....	192
4.3.6 Closure of Test Problem II .....	199
4.4 Role of Chapter 4 in this Dissertation .....	201
4.4.1 Summarizing How We Finish Task I – Connecting Formulation and Exploration...	201
4.4.2 Summarizing How We Realize Type I & II Robust Design .....	204
4.4.3 Role of Chapter 4.....	205
Chapter 5 Type I, II, & III Robust Design through Improving Approximation.....	206
5.1 Frame of Reference on Solution Algorithms.....	211
5.2 Problem Statement – Limitations of the ALP regarding Parameter Determination.....	213
5.2.1 Adaptive Linear Programing (ALP) Algorithm.....	213
5.2.2 Reduced Move Coefficient (RMC).....	219
5.2.3 Limitations of the ALP Regarding the Determination of the RMC .....	221
5.2.4 Hypothesis of Improving the ALP.....	222
5.3 The Adaptive Linear Programing Algorithm with Parameter Learning (ALPPL).....	223
5.3.1 Step 1 – Identify the Criteria for Evaluating the Quality of a Solution.....	224
5.3.2 Step 2 – Developing the Evaluation Indices (EIs).....	227
Index for evaluating the fulfillment of the goals – $\mu Z$ and $\sigma Z$ .....	229
Index for evaluating robustness – $\mu N_{ab}$ , $\sigma N_{ab}$ , $\mu N_{aoc}$ and $\sigma N_{aoc}$ .....	229
Index for evaluating the computational complexity – $\mu N_{acc}$ , $\sigma N_{acc}$ , $\mu N_{it}$ and $\sigma N_{it}$ .....	230
5.3.3 Step 3 – Learning the DEI and Tuning the RMC .....	231
5.4 The Hot Rolling Process Chain Problem.....	234

5.4.1 Statement of Test Problem 2 .....	234
5.4.2 Applying ALPPL .....	240
5.4.3 Parameter Learning Results and Discussion .....	245
Verification of the improvement of ALPPL over ALP. ....	248
5.5 Role of Chapter 5 in this Dissertation .....	250
5.5.1 Summarizing How We Finish Task 2: Connecting Approximation, Exploration, and Evaluation.....	250
5.5.2 Summarizing How We Realize Type I, II, & III Robust Design .....	253
5.5.3 Role of Chapter 5.....	254
Chapter 6 Type I, III, & IV Robust Design through Unsupervised Learning .....	256
6.1 Frame of Reference on Multi-Goal Problems .....	262
6.1.1 Features of Concurrent Engineering Problems .....	262
6.1.2 Two Categories of Studies on Multi-Goal Problems.....	262
6.1.3 Differences between a Goal and an Objective .....	265
6.1.4 Common Ways of Combining the Goals $\mathbf{z}(\mathbf{d})$ .....	267
6.2 Problem Statement – Test Problem 3: The Rankine Cycle Problem.....	272
6.2.1 Problem Description.....	272
6.2.2 Model Formulation.....	273
6.3 The Adaptive Leveling-Weighting-Clustering (ALWC) Algorithm.....	278
6.3.1 Clustering the Goals based on their Interrelationship.....	278
6.3.2 A Schematic of the ALWC Algorithm .....	284
6.3.3 The Algorithms in the ALWC.....	286
6.4 Unsuperfized Learning Results and Discussions .....	291

6.4.1 Clustering result .....	291
6.4.2 Improvement in Goal Achievement along the Design Scenario Expansion .....	292
6.4.3 Reducing the Euclidean Distance to the Utopia Point .....	293
6.4.4 Reducing Computational Complexity.....	294
6.4.5 Verification of the Results.....	295
6.4.6 Closing Remarks on Using ALWC to Speed up Learning .....	298
6.5 Role of Chapter 6 in this Dissertation .....	299
6.5.1 Summarizing How We Finish Task 3: Connecting Formulation, Exploration, and Evaluation.....	299
6.5.2 Summarizing How We Realize Type I, III, & IV Robust Design .....	302
6.5.3 Role of Chapter 6.....	304
Chapter 7 Type I, II, & IV Robust Design through Emergent Properties Identification and Interpretations.....	305
7.1 Frame of Reference on Designing Promotions using Agent-Based Modeling .....	308
7.2 Problem Statement – Promoting the Second-Season Cultivation in an Island Village in India .....	310
7.3 Modeling and Scenario Development .....	314
7.3.1 Build the Architecture and Set the Baseline Scenario of the Agent Based Model.....	314
7.3.2 Scenario Development.....	317
7.4 Results and Discussions .....	318
7.4.1 Exploring the Network Type and Promotion Effort and their Interaction Effects .....	319
7.4.2 Exploring the Promotion Duration.....	325
7.4.3 Anticipation and Profit Exploration .....	328

7.4.4 Closing Remarks.....	330
7.5 Role of Chapter 7 in this Dissertation .....	332
7.5.1 Summarizing How We Connect Formulation, Approximation, Exploration, and Evaluation.....	332
7.5.2 Summarizing How We Realize Type I, II, & IV Robust Design.....	335
7.5.3 Role of Chapter 7.....	337
Chapter 8 Validation of the Hypotheses in Realizing Model Evolution .....	339
8.1 Contributions .....	341
8.1.1 Summarizing the Theoretical Foundation .....	345
8.1.2 Summarizing the Test Problems .....	347
8.1.3 Summarizing the Answer to the Research Questions .....	348
8.1.4 Summarizing the Four Types of Robust Design .....	350
8.2 Application Scope of the Proposed Methods .....	352
8.2.1 Application Scope of the Design Evolution Loop.....	352
8.2.2 Application Scope of M1 – Formulation-Exploration Framework .....	352
8.2.3 Application Scope of M2 – Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL) .....	353
8.2.4 Application Scope of M3 – Adaptive Leveling-Weighting-Clustering Algorithm (ALWC) .....	353
8.2.5 Application Scope of M4 – Scenario-Planning for Simulations.....	354
8.3 Other Examples .....	354
8.3.1 Network Planning for Improving Hospital Visiting Process.....	354
8.3.2 Leveraging Social Drivers in Rural Development .....	356

8.3.3 Knowledge Management in Designing Cyber-Physical Product-Service Systems .....	357
8.4 Role of Chapter 8 in this Dissertation .....	358
Chapter 9 Closing Remarks – Advancing Model Evolution in Other Disciplines .....	360
9.1 Summary of This Dissertation .....	360
9.1.1 Motivation of Model Evolution using Satisficing Strategy .....	360
9.1.2 Contributions – Research Questions and Answers Leading to New Knowledge .....	361
9.1.3 Verification and Validation .....	362
9.1.4 Relevant Publications .....	363
9.1.5 Closing Remarks of the Summary .....	364
9.2 Way Forward – “I Statement” .....	365
9.2.1 Overarching Research Theme and Goals .....	365
9.2.2 Research Thrusts and Applications .....	366
9.2.3 Potential Cross-Disciplinary Research Opportunities .....	370
9.2.4 Closing Remarks of the Way Forward .....	372
9.3 Role of Chapter 9 in this Dissertation .....	373
References .....	375
Appendix A The 34 Weight Scenarios (WSs) and the Corresponding Achievement of the Goals .....	384
Appendix B Results of the 22 Weight Scenarios (WSs) From the Improved Model (First Iteration). .....	385
Appendix C Mathematical Formulation of the CODP as a cDSP .....	386

Appendix D The RMC Tuning Algorithm Customized for the Hot Rolling Process Chain Problem  
(Chapter 5).....390

Appendix E Lin Guo’s Defense Slides and Speech .....393

## LIST OF TABLES

Table 1. 1 The Logic Flow and the Role of Each Chapter .....	3
Table 1. 2 Advantages and Disadvantages of The Two Categories of Solution Algorithms .....	13
Table 1. 3 Several Representative Methods and Their Features .....	17
Table 1. 4 Problems in Methods of Optimizing Strategy .....	23
Table 1. 5 Problems in Methods of <i>Satisficing</i> Strategy .....	25
Table 1. 6 The Research Gaps (RG) and Hypotheses (H).....	28
Table 2. 1 The Advantages of Realizing <i>Satisficing</i> Strategy Using cDSP and ALP in the Each Stage of Engineering Design .....	42
Table 2. 2 The Features of the Toy Problems (TP) .....	45
Table 2. 3 Methods for Comparison the Two Strategies.....	45
Table 2. 4 The Optimization Model and Compromise DSP of TP-I.....	46
Table 2. 5 Solutions to TP-I ( <i>dominated solutions of each scenario</i> ) .....	48
Table 2. 6 The Optimization Model and Compromise DSP of the TP-II.....	51
Table 2. 7 Solutions to TP-II Using Each Solution Algorithm ( <i>dominated solutions, close-to-nondominated solutions or good-enough solutions</i> ) .....	53
Table 2. 8 The Optimization Model and Compromise DSP of the TP-III .....	59
Table 2. 9 Solutions to TP-III ( <i>dominated solutions, close-to-nondominated solutions or good-enough solutions</i> ).....	60
Table 2. 10 The Compromise DSP of the TP-IV .....	68
Table 2. 11 Solutions to TP-IV ( <i>close-to-nondominated solutions or good-enough solutions</i> ) ...	70
Table 2. 12 The Word-Form and Math-Form of the Compromise DSP of TP-V .....	76
Table 2. 13 Solutions to TP-V – the nondominated solution of each scenario is highlighted, the solution that gives the better achieved value of a goal but is not a nondominated solution is underlined .....	77



Table 2. 14 Justified Research Questions regarding Four Types of Robust Design .....	87
Table 2. 15 Connection between Research Questions (RQs) and Chapters (Ch).....	89
Table 2. 16 Specification of Hypotheses for Answering the Research Questions .....	90
Table 2. 17 Plan of Addressing the Research Questions in Each Chapter .....	92
Table 3. 1 Plan of Theoretically Verifying the Specified Hypotheses and Demonstrating the Proposed Methods in Each Chapter .....	95
Table 3. 2 Summary of Test Problems – The robust design type, testified methods, and uncertainties of each test problems. The uncertainties underlined in italic are managed in this dissertation.....	127
Table 4. 1 Plan of Specifying Research Question 1 (RQ1) and Empirically Verifying the Formulation-Exploration Framework (M1) .....	132
Table 4. 2. Gaps and limitations of the Methods in the Literature .....	147
Table 4. 3 Features of 14 Dams in Red River Basin .....	158
Table 4. 4 Results for Equal Weights on Preferences .....	161
Table 4. 5 Physical Meaning of the Eight WSs – Type II Uncertainty .....	163
Table 4. 6 Range of Weights of the Satisficing Space .....	165
Table 4. 7 Inflow Scenarios (ISs) – Type I Uncertainty.....	166
Table 4. 8 Active Bounds in Each IS and WS.....	166
Table 4. 9 Improvable Bounds in Each IS and WS .....	167
Table 4. 10 Suggestions for Model Improvement .....	170
Table 4. 11 Sensitive Segments of the Model of the Second Iteration.....	171
Table 4. 12 The Algorithm for Model Improvement .....	173
Table 4. 13 Four Types of Robust Design and the Interpretations in SC.....	180
Table 4. 14 Algorithm for System Capacity Analysis.....	188

Table 4. 15 Seven Scenarios – Type II Uncertainty .....	193
Table 4. 16 Results of the First Iteration .....	193
Table 4. 17 Results of the Seven Design Scenarios in the Third Iteration .....	194
Table 4. 18 Comparison of Satisficing Results with Results from Other CODP Candidate Locations .....	197
Table 4. 19 Summary of Test Problems 1.1 & 1.2 regarding Type I&II Uncertainty Management .....	204
Table 5. 1 Plan of Specifying Research Question 2 (RQ2) and Empirically Verifying the Adaptive Linear Algorithm with Parameter Learning (ALPPL) (M2) .....	207
Table 5. 2 Advantages and disadvantages of the two categories of solution algorithms .....	212
Table 5. 3 Criteria for the Evaluation of Approximation Performance.....	227
Table 5. 4 Develop the Evaluation Indices (EIs) from the Information Obtained from ALP Running .....	228
Table 5. 5 The Parameter Learning Process – for RMC tuning .....	233
Table 5. 6 Weight Vectors Used in (Nellippallil, Rangaraj et al. 2018) as Different Design Scenarios.....	240
Table 5. 7 Results of EIs Using Sample RMC Values with Nineteen Design Scenarios.....	242
Table 5. 8 The Initial DEI.....	243
Table 5. 9 The Record of the EIs, DEI, RMC, Best RMC of the Fourteen Iterations of RMC Tuning .....	245
Table 5. 10 ALPPL with RMC Tuning versus ALP with Golden Section Search.....	249
Table 5. 11 Summary of Test Problems 2 regarding Type I, II&III Uncertainty Management..	253
Table 6. 1 Plan of Specifying Research Question 3 (RQ3) and Empirically Verifying the Adaptive Leveling-Weighting-Clustering (ALWC) Algorithm (M3) .....	257

Table 6. 2 The Features and Limitations of Some Classic Multi-Objective (Multi-Goal) Solution Algorithms and Methods .....	263
Table 6. 3. A part of the normalized deviations of a six-goal cDSP with 81 Iterations with <b>L = 3</b> , <b>p = 2</b> .....	289
Table 6. 4 The clustering results along iterations .....	291
Table 6. 5 The summary of the clustering results ever returned to update the leveling .....	291
Table 6. 6 Statistics of the Results.....	292
Table 6. 7 Statistics of the Euclidean Distance to the Utopia Point of the Results under Each Clustering Scenario .....	294
Table 6. 8 Meaning of the Three Clusters .....	296
Table 6. 9 Summary of Test Problems 3 regarding Type I, II, & IV Uncertainty Management.	303
Table 7. 1 Plan of Specifying Research Question 4 (RQ4) and Empirically Verifying the Scenario Planning Framework in Agent-Based Modeling (M4).....	306
Table 7. 2 Some Representative Applications of Agent-Based Modeling (ABM) for New Technology Acceptance and Policy Impact .....	309
Table 7. 3 The SE's Target based on the Current Situation .....	313
Table 7. 4 Transitions between Different States for an Agent .....	315
Table 7. 5 Scenarios for Testing Each Factor.....	317
Table 7. 6 The Expected Outcome of the Scenario Planning.....	318
Table 7. 7 The Summary of the Results of the Scenario Planning.....	319
Table 7. 8 Promotion Effort Exploration – Migration Household in the Promotion Year and in the End-of-Project Year with Different Network Scenarios .....	324
Table 7. 9 Migration Population (Households) during the Four Years with Different Promotion Durations .....	328
Table 7. 10 Summary of Test Problems 4 regarding Type I, II, & IV Uncertainty Management .....	336

Table 8. 1 The Support for Chapter 8 in Previous Sections .....	341
Table 8. 2 Summary of Addressing the Research Questions and Verifying the Hypotheses – with section number.....	343
Table 8. 3 Addressing the Research Question 1 and Verifying the Hypotheses .....	344
Table 8. 4 New Knowledge in this Dissertation.....	349
Table 8. 5 Summary of the Realization of the Four Types Robust Design.....	351
Table 9. 1 Research Thrusts and Application in My Early Career.....	365

## LIST OF FIGURES

Figure 1. 1 Organization of Chapter 1 .....	2
Figure 1. 2 The Model World and Its Corresponding Physical World .....	5
Figure 1. 3 The Evolution Cycle of Complex Systems Realization.....	6
Figure 1. 4 The Thermal System around the Rankine Cycle .....	11
Figure 1. 5 The Optimization Strategy and the <i>Satisficing</i> Strategy .....	13
Figure 1. 6 The Optimal Solutions are Included in the Satisficing Solutions .....	15
Figure 1. 7 The Correspondence between Physical World and Model World.....	20
Figure 1. 8 Establishing Connections among Multiple Stages of Engineering-Design Evolution Cycle, Formulation, Approximation, Exploration, and Evaluation.....	22
Figure 1. 9 The Research Gaps and the Potential Contributions by Filling the Research Gaps ...	27
Figure 1. 10 Expected Contributions.....	29
Figure 1. 11 Dissertation Layout and Plan of Verification and Validation.....	31
Figure 1. 12 Organization of the Chapters .....	32
Figure 2. 1 Organization of Chapter 2.....	35
Figure 2. 2 The first-order necessary Kuhn-Tucker conditions are satisfied at $\mathbf{x}^*$ .....	37
Figure 2. 3 The convexity requirements for satisfying the second-order sufficient Kuhn-Tucker conditions .....	38
Figure 2. 4 Lagrange multipliers fail to identify an optimal for a highly convex objective.....	39
Figure 2. 5 The Assumptions When Using the Optimizing Strategy and <i>Satisficing</i> Strategy .....	41
Figure 2. 6 The Two Objective Functions in the X-f(X) Space of TP-I .....	47
Figure 2. 7 The Solution Points to TP-I on the Objective Space Using Five Algorithms – the Same .....	49

Figure 2. 8 The Solution Points to TP-I on the  $x$ - $f(x)$  Space. (a) is the 3D illustration of Objective 1,  $f_1(x)$ . (b) is the 3D illustration of Objective 2,  $f_2(x)$ . Since the solutions are the same when using different formulations and algorithms, so all three points are the same for all the five methods in Table 2.5. ....49

Figure 2. 9 The Two Objective Functions on the  $X$ - $f(X)$  Plane of TP-II – The first objective  $f_1x$  is non-convex.....52

Figure 2. 10 The Solution Points to TP-II on the Objective Space Using Four Algorithms – Solutions returned by Trust-constr and SLSQP are not “good enough,” solutions returned by ALP are “good enough” and diverse, and solutions returned by NSGA II contain nondominated solutions but are not diverse .....54

Figure 2. 11 The Solution Points to TP-II on the  $x$ - $f(x)$  Space – Using Trust-constraint and SLSQP are easy to fall into local optima. Green, blue, red, and dark red dots are the solutions using Trust-constr, SLSQP, ALP, and NSGA II, respectively. ....54

Figure 2. 12 Illustration of the Sequential Linearization using the ALP with Different Views When the Quadratic Approximated Paraboloid Has Real Roots .....56

Figure 2. 13 Linearization using the ALP When the .....57

Figure 2. 14 Using the Accumulated Constraints from Multiple Linearization Iterations for Convex or Slightly Non-Convex Equations and Using Single Linearized Constraint for Significantly Non-Convex Constraint .....58

Figure 2. 15 The Two Objective Functions of TP-III on the  $x$ - $f(x)$  Space – The second objective  $f_2x$  is enlarged by 50 times versus that of TP-II.....60

Figure 2. 16 The Solution Points to TP-III on the Objective Space Using Two Algorithms – Solutions returned by NSGA II are closer to the nondominated solution and more diverse but sensitive to parameter setting and require higher computational power. ....61

Figure 2. 17 The Solution Points to TP-III on the  $X$ - $f(X)$  Plane – Little performance differences between ALP and NSGA II.....61

Figure 2. 18 Satisficing solutions in different cases.....65

Figure 2. 19 The Left-Hand Side (Objective Function) and the Right-Hand Side (Target) of the Two Goals of TP-IV on the X-f(X) Space .....	69
Figure 2. 20 The Solution Points to TP-IV on the Objective Space Using Two Algorithms – NSGA II finds more nondominated solutions, whereas ALP finds solutions close to nondominated solutions but with better weighted combined goal-achieved value .....	71
Figure 2. 21 The Solution Points to TP-IV on the X-f(X) Space – Little performance differences between ALP and NSGA II .....	71
Figure 2. 22 The Functionality of an Elephant Stand (TP-V) .....	74
Figure 2. 23 The Dimension of an Elephant Stand (TP-V) .....	75
Figure 2. 24 The Box Chart Solution Points to TP-V on the Objective Space Using Two Algorithms .....	78
Figure 2. 25 The Solution Points to TP-V on the Deviation Space Using Two Algorithms – Round dots are solutions using the ALP and Triangle dots are solutions using NSGA II .....	79
Figure 2. 26 Four Types of Robust Solution .....	86
Figure 2. 27 Justified Research Questions RQ1-RQ4 and Their Connections with the Design Loop .....	89
Figure 2. 28 Research Questions RQ1-RQ4 and Specified Hypotheses SH1-SH4 .....	91
Figure 3. 1 Organization of Chapter 3 .....	95
Figure 3. 2 Illustration of Research Questions (RQ1-RQ4), Specified Hypotheses (SH1-SH4), Theoretical Verification of Specified Hypotheses (TVe1-TVe4), and Methods (M1-M4) in the Context of Design Evolution Cycle.....	97
Figure 3. 3 The Methods and Procedures Involved in Formulation-Exploration – Realizing the model evolution through the items highlighted in red.....	99
Figure 3. 4 The Methods and Procedures Involved in Approximation-Exploration-Evaluation – Realizing the model evolution through the items highlighted in red .....	101

Figure 3. 5 The Methods and Procedures Involved in Formulation-Exploration-Evaluation – Realizing the model evolution through the items highlighted in red .....	102
Figure 3. 6 The Methods and Procedures Involved in Approximation-Formulation-Exploration-Evaluation – Realizing the model evolution through the items highlighted in red .....	104
Figure 3. 7 Theoretical Verification of Specified Hypothesis I (TVe1) – Exploring the sensitivity of the segments of the model boundary and improve accordingly .....	106
Figure 3. 8 Theoretical Verification of Specified Hypothesis 2 (TVe2) – Learn, evaluate, and update metaheuristics to improve model approximation.....	108
Figure 3. 9 Theoretical Verification of Specified Hypothesis 3 (TVe3) – Learn system nature such as interrelationship among subsystems and reorganize them based on it.....	109
Figure 3. 10 Theoretical Verification of Specified Hypothesis 4 (TVe4) – Capture and quantify emergent properties through scenario planning in simulations .....	110
Figure 3. 11 The Control Factors and Noise Factors Bring Variation in Goal Function .....	113
Figure 3. 12 Formulation-Exploration Framework .....	114
Figure 3. 13 Learn and Update Metaheuristics in an Algorithm Using Parameter Learning.....	117
Figure 3. 14 Learn and Speed up the Learning of Systems using Machine Learning.....	119
Figure 3. 15 The Process of Learning Critical Factors in a Simulation .....	122
Figure 3. 16 Test Problems for RDI-II – Refining the model formulation and identifying <i>satisficing</i> solutions relatively insensitive to the variation in parameters and decision variables .....	123
Figure 3. 17 Test Problems for RDI-II – Refining the model formulation and identifying <i>satisficing</i> solutions relatively insensitive to the variation in parameters and decision variables .....	124
Figure 3. 18 Finishing Theoretical Structural Validity in Chapter 1, 2, and 3 .....	130
Figure 4. 1 Organization of Chapter 4.....	131
Figure 4. 2 Specified Research Question 1 and the Relevant Stages to be Connected in Design Evolution Cycle .....	134
Figure 4. 3 Dams along the Red River Basin .....	140



Figure 4. 4 The 14-Dam Network .....	141
Figure 4. 5 A Small Part of the Dam-Network in the Red River Basin .....	142
Figure 4. 6 The Pools of a Reservoir .....	144
Figure 4. 7 Illustration of the Equality Constraints for Dam (Reservoir) d .....	145
Figure 4. 8 Three Steps for the Exploration of the Solution Space .....	153
Figure 4. 9 Method for Exploration of the Solution Space .....	154
Figure 4. 10 Visualization of the Eight WSs in the Ternary Plot.....	163
Figure 4. 11 Feasible Weight Area of Goal 1 – Reservoir .....	163
Figure 4. 12 Feasible Weight Area of Goal 2 – People.....	164
Figure 4. 13 Feasible Weight Area of Goal 3 – Fish.....	164
Figure 4. 14 Satisficing Weight Area for Three Goals.....	164
Figure 4. 15 Bring the Solution Away from the Boundary by Restricting the RHS .....	169
Figure 4. 16 Applying the Physical Boundary by Relaxing RHS and Then Bring the Solution Away from the Physical Boundary by Restricting RHS.....	169
Figure 4. 17 The satisficing area of the weights of original model (a) and improved model (b)	171
Figure 4. 18 Improvement through Iterating .....	173
Figure 4. 19 Possible location of CODP in a SC.....	177
Figure 4. 20 CODP of Different Industries .....	177
Figure 4. 21 Multiple Conflicting Goals in SCs.....	180
Figure 4. 22 Formulation-Exploration Framework .....	186
Figure 4. 23 A Three-Echelon SC .....	192
Figure 4. 24 CODP of Different Weight Scenarios.....	194
Figure 4. 25 CODP, Achieved Value of Goals in Different Phases of a Product Life Cycle – Managing Type I Uncertainty .....	198

Figure 4. 26 The Methods and Procedures Involved in Formulation-Exploration – Establish the information exchange, knowledge awareness, and instructions sharing among the three highlighted processes.....	203
Figure 5. 1 Organization of Chapter 5.....	207
Figure 5. 2 Specified Research Question 2 and the Relevant Stages to be Connected in Design Evolution Cycle.....	209
Figure 5. 3 The ALP Algorithm.....	214
Figure 5. 4 The Approximation and Obtained Solution using the ALP in Two Iterations.....	215
Figure 5. 5 The Original Nonlinear Constraint, the Second-Order Paraboloid, and the Secant Plane [4].....	216
Figure 5. 6 When the Second-Order Paraboloid Has No Intersection with Plane $\mathbf{x1x2}$ , The First-Order Tangent is Used to Approximate $\mathbf{NFj}$ .....	218
Figure 5. 7 The Golden Section Search for The RMC in the ALP.....	220
Figure 5. 8 Possible Patterns of the Performance of the RMC in a Sub-Range.....	222
Figure 5. 9 The Concept of Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL).....	224
Figure 5. 10 A Relatively Sensitive Solution and a Robust Solution (Relatively Insensitive to Uncertainties).....	226
Figure 5. 11 Unnecessary Accumulated Constraints versus Necessary Accumulated Constraints.....	227
Figure 5. 12 ALPPL Includes Parameter Initialization and the RMC Tuning.....	232
Figure 5. 13 The Satisficing Weight Set When Setting RMC=0.1.....	239
Figure 5. 14 The Satisficing Weight Set When Setting RMC=0.5.....	239
Figure 5.15 The Satisficing Weight Set When Setting RMC=0.8.....	240
Figure 5. 16 EIs and DEI of the Sample RMC Values.....	241
Figure 5. 17 Identifying the Insensitive Range of RMC Value Using Twenty RMC Values.....	247

Figure 5. 18 The Fourteen RMC Values in the RMC Tuning.....	248
Figure 5. 19 The Comparison of ALPPL And ALP regarding the RMC Updating.....	249
Figure 5. 20 The Procedures Involved in Approximation-Exploration-Evaluation – Establish the information exchange, knowledge awareness, and instructions sharing between deduction and decision.....	252
Figure 6. 1 Organization of Chapter 6.....	256
Figure 6. 2 Specified Research Question 3 and the Relevant Stages to be Connected in Design Evolution Cycle.....	259
Figure 6. 3 Archimedean Strategy (Weighted Sum).....	268
Figure 6. 4 Pre-Emptive Strategy (Lexicographic Ordering).....	269
Figure 6. 5 An Ensemble Strategy using a Mixture of Archimedean and Pre-emptive Strategy	271
Figure 6. 6 The Thermal System.....	273
Figure 6. 7 Rankine Cycle (Temperature and Entropy).....	274
Figure 6. 8 Using Multiple Design Scenarios to Obtain a Deviation Matrix.....	279
Figure 6. 9 Cluster Analysis Using a Deviation Matrix.....	279
Figure 6. 10 The <i>Satisficing</i> Solutions to a Three-Goal cDSP under Two Design Scenarios Illustrated in a Two-Dimensional Solution Space.....	281
Figure 6. 11 The <i>Satisficing</i> Solutions to a Three-Goal cDSP under Two Design Scenarios Illustrated in Two Two-Dimensional Goal Spaces.....	282
Figure 6. 12 The Orthogonality between the Deviation Vectors of Two Goals using Two Design Scenarios.....	283
Figure 6. 13 The Flowchart of the Adaptive Leveling-Weighting-Clustering (ALWC) Loop ...	285
Figure 6. 14 Weight Vectors of a Three-Goal Problem with $p = 3$ .....	287
Figure 6. 15 An Example of Improving Goal 3 by 20% While Worsening Goal 1 and Goal 2 by 80% Respectively.....	294

Figure 6. 16 Scatter plots of any two goals using deviations of 1-level, 21 weight vectors .....	297
Figure 6. 17 The Procedures Involved in Formulation-Exploration-Evaluation – Establish the information exchange, knowledge awareness, and instructions sharing among formulation deduction, decision, and action .....	301
Figure 7. 1 Organization of Chapter 7 .....	305
Figure 7. 2 Specified Research Question 3 and the Relevant Stages to be Connected in Design Evolution Cycle .....	308
Figure 7. 3 The satellite map of Kudagaon .....	311
Figure 7. 4 The SE’s plan for facilitating second-season cultivation.....	313
Figure 7. 5 The Flowchart of the Agents’ State Transitions .....	316
Figure 7. 6 The Trigger Conditions and/or the Duration of the Transitions between the States of the Agents .....	317
Figure 7. 7 Results for Two Network Types When Promotion Reaches All Households .....	322
Figure 7. 8 Results for Two Network Types When Promotion Reaches 50% of Households....	323
Figure 7. 9 The Migration Households with Different Promotion Effort .....	324
Figure 7. 10 Simulation Results for Different Promotion Durations – Using a Distance-Based Network with a 75-Meter Influence Radius .....	327
Figure 7. 11 Simulation Results of Three Scenarios of Anticipation $\beta_2$ and Profit $\alpha$ .....	329
Figure 7. 12 Scenario Planning for Identifying Critical Factors in Simulation.....	331
Figure 7. 13 The Procedures Involved in Formulation-Approximation-Exploration-Evaluation – Establish the information exchange, knowledge awareness, and instructions sharing among formulation, decision, and action .....	334
Figure 7. 14 Finishing Empirical Structural Validity in Chapter 4, 5, 6, and 7 .....	338
Figure 8. 1 Organization of Chapter 8.....	340
Figure 8. 2 Finishing Empirical Performance Validity in Chapter 8 .....	359

Figure 9. 1 A Knowledge-Based Design Guidance for CPPSS .....	368
Figure 9. 2 Managing Complex Systems with Different Types of Causality .....	370
Figure 9. 3 Finishing Theoretical Performance Validity in Chapter 9 .....	374

## **ABSTRACT**

George Box said, “All models are wrong, but some are useful.” In the design of complex systems, types of complexity need to be managed. Giving the complexities that a decision maker may encounter, corresponding adjustments or improvements should be made to the design. In this dissertation, it is defined that all kinds of engineering design are comprised of four stages – formulation, approximation, exploration and evaluation – and the four stages form the model evolution loop or design evolution loop. By running the design evolution loop iteratively, a designer can handle the complexities and improve the design. Such improvements include but not limited to more robust to uncertainties, more efficient in design evolutions, easier interpretations of phenomena, etc.

In the design of complex systems, as lack of data and information, heuristics are used to proceed the design, so that designers can explore the solution space and gain insight to improve the design. Those heuristics include but not limit to model structures, sub-problems identification and integration, approximation rules, and scale of details incorporated in the model. There is lacking mechanisms to evaluate the quality of the design associated with the heuristics.

In this dissertation, it is hypothesized that by running the design evolution loop and exploring the solution space, designers can do the things as follows to improve the design.

- Evaluating system performances associated with various heuristics (structure of the model, critical parameter setting, rules making, etc.).
- Replacing the heuristics with insight obtained from exploration of the solution space to improve the design.

- Managing the complexity of module structure, such as analyzing and simplifying the structure of a large number of goals.
- Interpreting the behavior and the property of the model into the knowledge that supports the decision making.
- Capturing and managing newly observed properties or a more detailed complexity that are not incorporated into the modeling at first – the emergent properties.
- Automating the steps in the above.

The intellectual merits in this dissertation are the expandable computational framework for designing complex systems and managing multiple types of uncertainty– the design evolution loop, and the methods fitting into it. By using *satisficing* strategy and incorporating machine learning to explore the solution space, heuristics in each of the four stages (formulation, approximation, exploration, and evaluation) can be updated or replaced by knowledge gained from experiments, calculations and analyses. In addition, knowledge on tradeoffs between different categories of design requirement – such as (but not limited to) approximation accuracy, computational complexity, design preference diversity, reformulation flexibility, and the degree of design automation – can be collected, stored and reused.

## DEFINITION OF TERMS

**Accumulated constraints** – The linearized constraints that are accumulated along the iterations. designers need accumulated constraints to ensure a nonlinear problem to be linearized to a linear problem with a certain level of accuracy.

**Active bounds** – For a solution, if the value of a variable is on its upper or lower bound, then the bound is an active bound.

**Active constraints** – The constraints with zero constraint capacity (or zero slack or surplus). If a solution is on the boundary formed by a constraint, then the constraint is an active constraint. In some literature, active constraints are also considered as “binding,” and the inactive constraints are defined as “slack.” In other words, for an inequality constraint, when being plugged in the solution point, its left-hand side value equals to its right-hand side value, then such an inequality constraint is an active constraint; otherwise it is an inactive constraint.

**Aspiration** – The set in the solution space in which the solutions meet the target of all goals<sup>1</sup>. The aspiration is an deterministic, ideal set that may violate the constraints or bounds, and may not be achieved by solving the cDSP.

**Boundary of the solution space** – The constraints or bounds of the model that bound the feasible solution space.

---

<sup>1</sup> Unlike the objective of Linear Programming problems, each goal of goal programming problems has a target value, which allow us to plot the goal functions on the solution space.



**Constraint capacity** –The slack or the surplus (the difference between left-hand-side value and right-hand-side value) of an inequality constraint when plugging a solution in the constraint.

**Emergent property (collective property)** – An emergent property is a property which a collection or complex system has, but which the individual members do not have.

**Feasible solution space** – The space inside the searching space, bounded by constraints and bounds of variables, containing and only containing feasible solutions.

**Heuristics** – Heuristics are the assumptions, instincts, experiences, common senses, or domain expertise that used in a method to speed up the process of finding a solution. Heuristics can be mental shortcuts that ease the cognitive load of making a decision<sup>2</sup>. Heuristics are employed in methods that are not guaranteed to be optimal.

**Insight** – Insight is the understanding of a specific cause and effect within a particular context. The term insight can have several related meanings: a piece of information, the act or result of understanding the inner nature of things or of seeing intuitively, an understanding of cause and effect based on identification of relationships and behaviors within a model, context, or scenario.

**Iteration** – The process in which the problem is linearized and solved. Multiple iterations are needed to get convergence in each design scenario. The purpose of having multiple iterations is to linearize the nonlinear problem at different points, solve each linearized problem and get the best linearization and solutions by the end of a synthesis cycle.

---

<sup>2</sup> This definition is from Wikipedia: <https://en.wikipedia.org/wiki/Heuristic>.

**Linearization point** – The point (spot) in the solution space where a linearization algorithm is applied to linearize the nonlinear functions of the cDSP. The linearization point does not change within one iteration but is updated from iteration to iteration.

**Mane-goal problem** – A problem with more than three goals.

**Nonlinear functions** – In this dissertation, it is defined that the nonlinear goals and nonlinear constraints are nonlinear functions.

**Robust design** – In this dissertation, robust design means the design that is relatively insensitive to one or more types of uncertainty. Type I uncertainty – the uncertainty brought by noise factors, for example, parameters. Type II uncertainty – the uncertainty brought by control factors such as decision variables. Type III uncertainty – the variation in the model structure. Type IV – the uncertainty brought by managing the first three types of uncertainty. We are aware of other definitions of *robust design*, but in this dissertation, in the context of designing complex systems, we define *robust design* as the above.

**Satisficing solution space** – The space inside the feasible solution space, identified by the decision maker as the set contains and only contains *satisficing* solutions.

**Satisficing weight set** – The set of weight scenarios applying which the *satisficing* solutions can be acquired.

**Searching space** – The space inside the solution space and bounded by the bounds of variables. In searching space, designers search for candidate solutions. The solutions in the searching space may violate constraints but satisfy the upper and lower bounds of all variables.

**Solution space** – The space of all potential solutions for a problem, including infeasible solutions.

**Synthesis cycle** – The cycle in which the problem is linearized and solved for multiple iterations and get convergence, and then the solution and relevant information of each iteration are evaluated. The critical parameters do not change within one synthesis cycle but are updated from cycle to cycle.

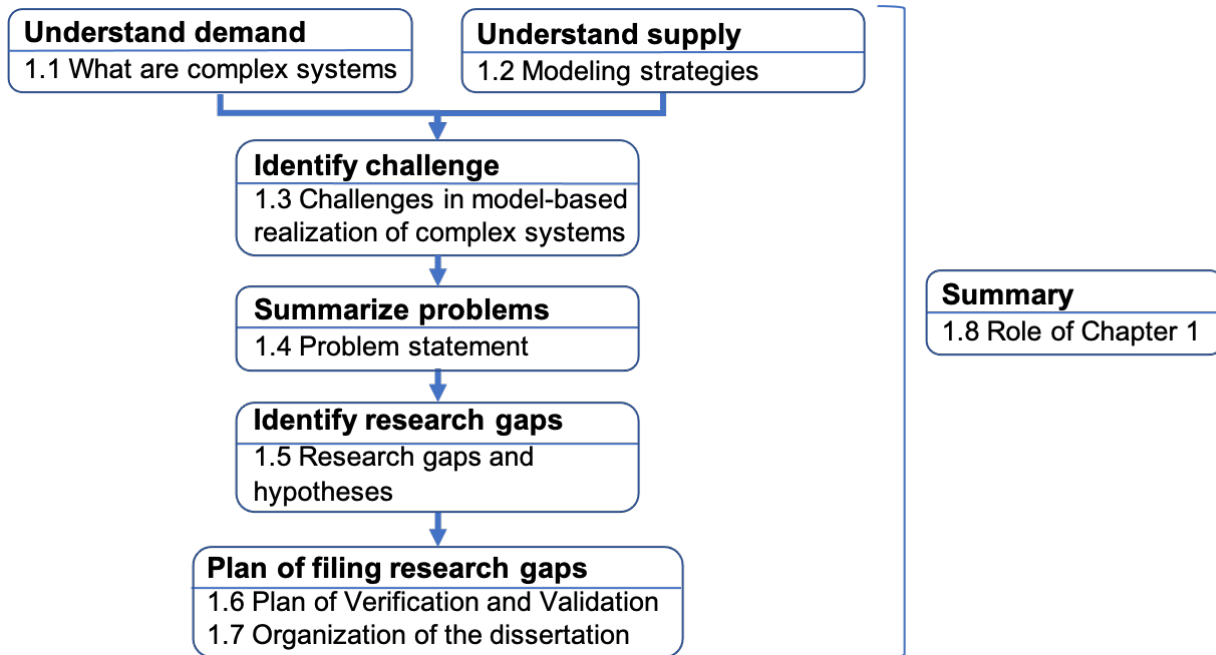
**Weight scenarios (weight vector)** – The different scenarios of values of the weights of multiple goals to represent various design preferences.

# CHAPTER 1 FRAME OF REFERENCE: DESIGNING COMPLEX SYSTEMS USING SATISFICING STRATEGY

## *– THE MODEL EVOLUTION CYCLE OF DESIGNING COMPLEX SYSTEMS*

*In Chapter 1, the context is established. The “why question” is answered – “why do we need model evolution. give the motivation of model evolution and describe why we manage complex systems in a certain way – using satisficing methods.*

In Chapter 1, the reference is framed, the motivation is introduced, and the organization is briefed; see Figure 1.1. We analyze the demand of designing complex systems by introducing their characteristics and examples in Section 1.1, analyze the supply of designing complex systems by introducing two typical modeling strategies in Section 1.2, discuss the challenges in complex-system realization regarding the gaps in demand and supply in Section 1.3, identify the research gaps in Section 1.4, pose the research question in Section 1.5, give our plan of answering the research question in Section 1.6 and 1.7, and summarize the role of Chapter 1 in this dissertation in Section 1.8.



**Figure 1. 1 Organization of Chapter 1**

In Table 1.1, it is summarized the logic flow in this dissertation and the role of each chapter. In Chapter 1, the research gaps (RG) are identified and given the corresponding hypotheses (H) to fill the research gaps. In Chapter 2, research questions (RQ) are posed. In Chapter 3, the hypotheses are theoretically verified, based on which, the corresponding methods are proposed. From Chapter 4 to Chapter 7, test problems are used to empirically verify the hypotheses and demonstrate the utility of the proposed methods, research questions are specified in the context of each test problem and answered through managing the test problems using the proposed methods. In Chapter 8, gives the closure of the answers to the research questions, and hypotheses are empirically validated. In Chapter 9, the theoretical extension of the research is provided which brings the topic into the beyond.

**Table 1. 1 The Logic Flow and the Role of Each Chapter**

Chapter	Ch1	Ch 2	Ch 3	Ch 4-7	Ch 8	Ch 9
Actions	RG H	RD RQ SH	TVe M	EVe SQT AQ	CQ EVa	TE
Nomenclature	RG – give research gaps H – give hypotheses RD – tie to roust design RQ – pose research questions SH – specify hypotheses TVe – theoretically verify hypotheses M – introduce methods EVe – empirically verify hypotheses SQT – specify research questions in the context of test problems AQ – answer research questions CQ – closure the answers to research questions EVa – empirically validate hypotheses TE – theoretically extend the research					

Table 1.1 is expanded by adding the summary of each action in the later chapters. Hereafter begins major part of Chapter 1.

### 1.1 What are Complex Systems?

A system is a set of entities that from a unified whole through their interactions, relationships, or dependencies. A complex system is a system composed of many components which may interact with each other<sup>3</sup>, such as a manufacturing system, a village, an economic entity. The relationships between the parts of a complex system are hard to predict and control (Ladyman, Lambert et al. 2013).

The most prominent feature of the realization of complex systems in the current age is the real-time information sharing and system evolution based on a highly integrated human-cyber-physical

---

<sup>3</sup> This definition is from Wikipedia: [https://en.wikipedia.org/wiki/Complex\\_system](https://en.wikipedia.org/wiki/Complex_system).

system. As George Box said, “All models are wrong but some are useful” (Box 1979, Box and Draper 1987), models are incomplete, inaccurate, and embody different levels of fidelity, the solution may be optimal to the model but may not be optimal to the real problem (the real complex system) which is way more complicated than the model.

If one categorizes everything in two worlds – model world and physical world, see Figure 1.2, he or she can observe that the model world is a simplified world, whereas the physical world is a much more complicated one.

In the model world, designers desire the problems to be linear, continuous, and convex. There are a lot of factors can be controlled, uncertainties can be predicted accurately and managed well, and all the necessary information can be collected in time. However, the physical world is quite the opposite. Although designers try to model the physical world by incorporating as many complexities as possible, there is always an intellectual disconnection between the two worlds.

## Model World



- Linear, continuous, and convex
- A lot factors to control but few requirements
- Confirmed and known goal combination
- No uncertainty or predictable uncertainty
- Have all necessary information

## Physical World



[www.usapangecon.com](http://www.usapangecon.com)

- Nonlinear, discrete, and non-convex
- Fewer factors to control but a lot of requirements
- Need to explore the ways to combine the multiple goals
- Multiple types of uncertainty
- Need to know more about the model robustness and ways to improve the model formulation

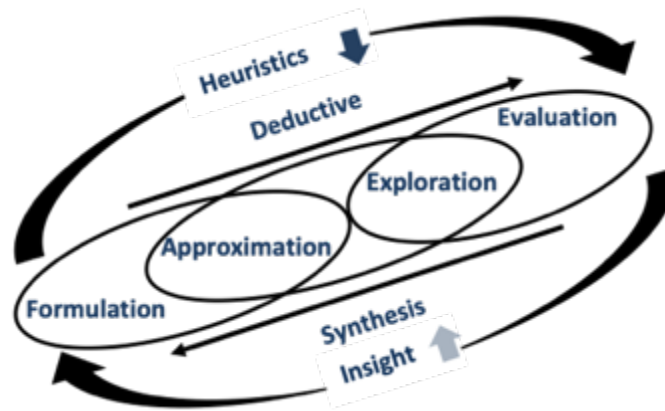
**Figure 1. 2 The Model World and Its Corresponding Physical World**

Although there is an intellectual disconnection between physical complex systems their corresponding decision models, designers need their models to be useful in giving decision support. The principal motivation in this dissertation is to make decision models of complex systems useful in support decision making through making a transformative influence in the realization and evolution of complex systems resulting with rapidly changing requirements.

In this dissertation, the research requirements for mathematics embodied in the realization and evolution of complex systems are identified; see Figure 1.3. There are four stages of the evolution cycle of complex systems realization, formulation, approximation, exploration, and evaluation. By acting forward from formulation to approximation, exploration, and evaluation, deductive methods are applied, and heuristics are used to proceed the system realization. With results and observations



obtained, calculations and analyses can be done, insight is acquired, and corresponding feedbacks can be given to make improvements or adjustments in the previous stages. By acting backward, synthesis methods are applied, and insight can be used to replace the heuristics. Through managing the interactions between different stages of the evolution cycle, knowledge on the connections between stages as well as general knowledge on the evolution of decision models can be acquired and synthesized.



**Figure 1. 3 The Evolution Cycle of Complex Systems Realization**

### ***1.1.1 Characteristics of Complex Systems***

The behaviors of a complex system are difficult to be inferred from the properties of the system (Smith 2003). If a researcher ignores such difficulties, he or she may formulate a model that are neither right nor useful. Different examples of complex systems are studied across many disciplines, however, the complex systems from various fields share some characteristics. The major characteristics of complex systems that are domain-independent include but not limit to 1) a complex system may have multiple, conflicting goals, 2) the preferences among the goals may evolve during the design or operation, 3) the number of goals may be large and hard to visualize using traditional methods, 4) functional relationships between elements may be nonlinear, 5) coupled decisions are required such as selecting an option associated with determining the

geometry, and 6) there are emergent properties in a complex system. These characteristics are managed in this dissertation. The descriptions of the characteristics are given as follows.

***Multiple conflicting goals.*** As there can be multiple users or stakeholders of a complex system, they may compete the same limited resource of the system, and their expectations from the system may conflict with each other, the designer or the entity who is in charge of the complex system needs to comply with users' different goals. For example, the users of a dam-network include the residents in the neighborhood, the farmers or industrial workers near the dam-network, and the fish and other wild animals that live along the river basin. They all consume the water in the river. Meeting each user's water demand are the multiple conflicting goals of the dam-network system.

***Evolving preferences among the goals.*** As a complex system needs to adapt to the changing environment, and the users' urgency of demand may be different over time, thus the priority of the goals may evolve. For example, a new product in its introduction stage, the stability of its supply chain should be prioritized, whereas in its maturity stage, the profit should be the most important goal.

***A large number of goals that are hard to visualize.*** As there can be a large number of elements and components incorporated in a complex system, the number of goals to be achieved by designing and operating the system can be large. Some of the goals may be highly orthogonal thus cannot be linearly combined. To visualize the tradeoffs between the goals and present the solutions that satisfy multiple goals are challenging tasks.

***Nonlinearity.*** Complex systems often have nonlinear behavior. Nonlinearity is a term in statistics to describe to a situation where there is not a straight-line or direct relationship between an

independent variable and a dependent variable<sup>4</sup>. Nonlinearity also indicates that a change in the size of the input of a system does not produce a proportional change in the size of the output. For example, the step cost, which is a cost that does not change steadily with changes in activity volume, but rather at discrete points.

***Coupling decision making.*** Multiple types of decisions are usually comprised in design and operating a complex system. The two typical decisions in systems realization are selection decision and compromise decision (Mistree, Hughes et al. 1993). The decision variables reflect such decision types – integer variables indicate selection decisions (such as selecting a material) and continuous variables indicate compromise decisions (such as designing the dimension). When there are mixed variables, it means that selection decision and compromise decision should be made at the same time and one type of decisions affects the other.

***Emergent properties.*** Emergent properties are properties of a group of elements that cannot be identified in any of the individual element. For example, in a village with 40% of the population refuses to change their lifestyle, if the rest 60% adopts some new technology and improve their social economic status, then almost the whole population will quickly adopt the new technology and change their life style because of the social pressure, although each individual still claims he or she may not accept such change.

---

<sup>4</sup> This definition is from Investopedia: <https://www.investopedia.com/terms/n/nonlinearity.asp>.

### ***1.1.2 Examples of Complex Systems***

The examples given in this section are the ones used in Chapter 4, 5, 6, and 7 to test the proposed methods. Each example may not contain all the characteristics of complex systems identified in Section 1.1.1 but contains at least one of them.

**A dam-network system.** The dams along a river basin form a network. Managing the supply and sensible distribution of fresh water to support human activity while sustaining vigorous, effective ecosystems is a major ecological challenge. The reservoir behind each dam stores water. The water in each reservoir is released to downstream by controlling the dam. There three user-groups in the basin – people, fish in the reservoirs, and fish in the streams between reservoirs. To meet water demands, there are three goals:

- 1) To reach the target for water storage in reservoirs.
- 2) To meet people’s demand for water – including agricultural and municipal demand.
- 3) To meet the water requirements for the fish in streams.

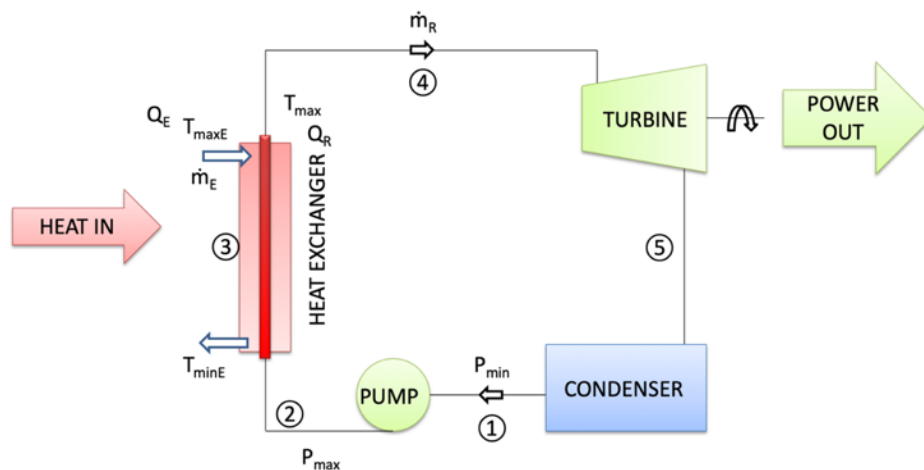
All the users compete the water resource in the river basin. There are uncertainties in the water inflow, such as precipitation and tributary inflow, as well as the water outflow, such as the demand of each user-group. Besides the complexity in the above, the demand of each user-group varies with the season, and the priority of each goal evolve. Therefore, in a dam-network system, the complexity includes managing multiple conflicting goals and the evolving preferences of the goals (Guo, Zamanisabzi et al. 2019). In Chapter 4, the dam-network planning problem is used as a test problem to testify the utility of the Three-Step Exploration Method in Chapter 4.

**A multi-stage manufacturing system.** Hot rolling is a multi-stage manufacturing process in which a reheated billet, slab, or bloom that is produced after the casting process is further thermo-mechanically processed by passing through a series of rollers (Nellippallil, Song et al. 2017, Nellippallil, Rangaraj et al. 2018). In this problem, the requirement is to produce steel rods with improved mechanical properties like yield strength, tensile strength, and hardness. These mechanical properties are defined by the microstructure after cooling, which includes, the phase fractions (ferrite and pearlite phases are only considered in this problem), pearlite interlamellar spacing, ferrite grain size, and chemical compositions (Nellippallil, Rangaraj et al. 2018). The microstructural requirements are to achieve a high ferrite fraction value, low pearlite interlamellar spacing, and low ferrite grain size value within the defined ranges. The requirement is to carry out the integrated design of the material and the process by managing the cooling rate (cooling process variable), final austenite grain size after rolling (rolling microstructure variable) and the chemical compositions of the material. In a multi-step manufacturing system, there are relatively complicated relationships between the independent and dependent variables, and many of such functional relationships are nonlinear. The goals of different steps and sub-process conflict with one another and their priorities change along with the circumstances. Therefore, in a multi-stage manufacturing system, the complexity includes multiple conflicting goals, evolving preferences among the goals, and nonlinearity between variables. In Chapter 5, the hot rolling problem is used to testify the utility of the Adaptive Linear Programming (ALP) Algorithm with Parameter Learning (ALPPL).

**A concurrent engineering system.** The thermal system around the Rankine cycle is a concurrent engineering system. The Rankine cycle is a mathematical representation of a heat engine that converts heat into mechanical work while undergoing phase change (Macquorn Rankine 1853,

Wikipedia 2019). The major components of the system are a power producing turbine, a pump to pressurize the flow to the turbine and two heat exchangers, a condenser, and a heater. See Figure 1.3. The system designer should deal with heat source issues (left side of Figure 1.4) and power use issues (right side of Figure 1.4) and the choice of working fluids. The common working fluid in a Rankine cycle is water. Uses of other fluids (often organic in chemistry) have given rise to the development of “organic Rankine cycles”. Of course, geometric specification and design analysis of physical elements in the system also represent opportunities for model and design space exploration. The designer needs to achieve six goals in this Rankine cycle:

- 1) Achieve zero moisture in steam leaving the turbine.
- 2) Maximize Rankine cycle efficiency.
- 3) Maximize temperature exchanger efficiency.
- 4) Maximize system efficiency indicator 1.
- 5) Maximize system efficiency indicator 2.
- 6) Maximize heat transfer effectiveness in exchanger.



**Figure 1. 4 The Thermal System around the Rankine Cycle**

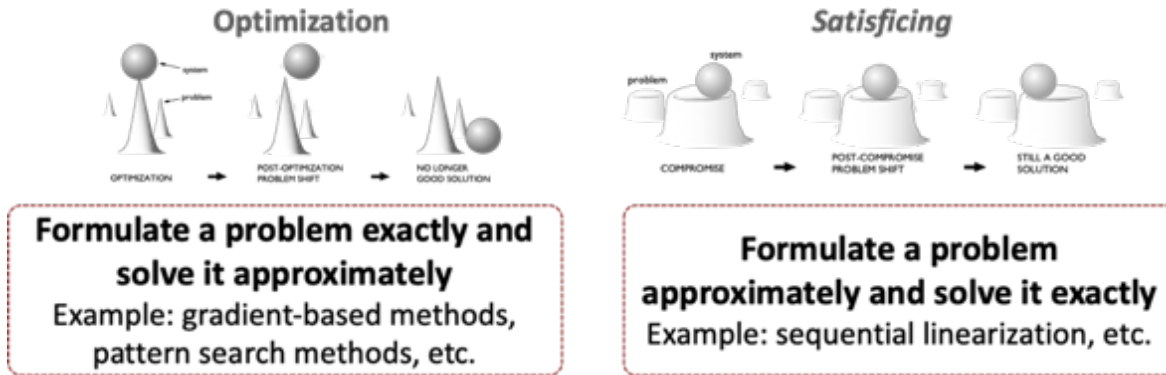
Therefore, in a concurrent engineering system such as a thermal system around the Rankine cycle, the complexity includes multiple conflicting goals, a large number of goals, and coupling decision

making. In Chapter 6, the thermal system design problem is used to testify the utility of the Adaptive Leveling Weighting (ALW) Algorithm.

**A cyber-socio-technical system.** A relatively isolated, underdeveloped village, as a small human society, is a complex system. When social entrepreneurs want to improve the villagers' social and economic status by promoting new technologies, the village is to become a cyber-socio-technical system. Such a system may have emergent properties – the collective properties that cannot be predicted from individual behaviors. The cognition and acceptance of new technologies and new lifestyles resulting from new technologies are the complexities that should be managed by the social entrepreneurs. To manage such complexities, simulations and predictions of the emergent properties and social behaviors should be addressed. In Chapter 7, the cyber-socio-technical system problem is used to testify the utility of the SDF\_ABM\_SD (social development framework using agent-based modeling and systems dynamic).

## 1.2 Modeling Strategies and Their Foci

Design methods and solution algorithms for dealing with complex problems fall into two categories; see Figure 1.5 and Table 1.2: formulate a complex problem exactly and solve it approximately or approximate a complex problem and solve it exactly. We name the first strategy as “optimizing” strategy and second strategy as “*satisficing*” strategy. Examples of the optimizing strategy are gradient-based methods (Williams and Zipser 1995), pattern search methods (Rios and Sahinidis 2013), penalty function methods (Viswanathan and Grossmann 1990), etc.; According to Herbert A. Simon, The decision maker has a choice between an optimal decision from an imaginary world, or decisions that are “good enough,” that *satisfice*, for a world approximating the complex real one more closely (Simon and Kadane 1975).



**Figure 1. 5 The Optimization Strategy and the *Satisficing* Strategy**

**Table 1. 2 Advantages and Disadvantages of The Two Categories of Solution Algorithms**

#	Category	Example Methods	Advantages	Disadvantages
<b>Optimizing Strategy</b>	Formulate a problem exactly and solve it approximately	Gradient-based methods, pattern search methods, penalty function methods, etc.	<ul style="list-style-type: none"> <li>- Maintaining a relatively accurate model along the solution search (given the information that the designer has on hand).</li> </ul>	<ul style="list-style-type: none"> <li>- The solution is still an approximate, inaccurate one;</li> <li>- Cannot get the information of the dual and use it to facilitate problem solving or post-solution analysis;</li> <li>- Heuristics are used in solution algorithms, which may result in premature convergence or unnecessarily high computational complexity.</li> </ul>
<b>Satisficing Strategy</b>	Approximate a problem and solve it exactly	ALP, SLP, SQL, etc.	<ul style="list-style-type: none"> <li>- Solutions are on the vertices of the approximated problem so the dual of the approximated problem can be explored;</li> <li>- Solutions may be away from the boundary of the original problem so they are relatively insensitive to variations;</li> <li>- The approximation of the problem can be improved by accumulating the linearized constraints during iterating and an approximated problem with acceptable level of accuracy can be obtained.</li> </ul>	<ul style="list-style-type: none"> <li>- Introducing information loss while doing approximation, making the solution inaccurate;</li> <li>- Heuristics are used in approximation algorithms, which may result in premature convergence or unnecessarily high computational complexity.</li> </ul>



### ***1.2.1 Optimizing Strategy and Satisficing Strategy***

Using the optimizing methods may lead to relatively higher computational complexity and the solution is usually not on the vertex of the feasible space. For a problem with features such as nonlinear, non-convex, multiple objectives, different units among objectives, using optimizing methods may fail to identify a solution even there is a feasible solution space because solutions should meet necessary and sufficient Kuhn-Tucker conditions, which make the solution set too small, hard to find, and sensitive to uncertainties captured or uncaptured into the decision model.

In contrast, the methods using the *satisficing* strategy such as Sequential Linear Programming, allow designers to obtain vertex solutions of the approximated linear problem. Such solutions meet the necessary Kuhn-Tucker conditions but may not meet the sufficient Kuhn-Tucker conditions, so we name these solutions “good enough” solutions or *satisficing* solutions. The *satisficing* solutions set contains the optimal solutions set, see Figure 1.6. Therefore, the *satisficing* set is easier to be found, especially when the problem is complex. If solutions are on the vertices, it enables designers to use duality embodied in linear programming to explore the solution space without having to perform numerical differentiation that is required if the solution is obtained using methods identified in Category 1 (Mistree, Hughes et al. 1981, Mistree and Kamal 1985). Further, *satisficing* methods facilitate quick identification of robust solutions<sup>5</sup>. Like most metaheuristic methods, they rely on fixed parameter heuristics to set parameters.

*Satisficing* strategy. *Satisficing* is a decision-making strategy of cognitive heuristic that entails searching through the available alternatives until an acceptability threshold is met. The term

---

<sup>5</sup> Solutions relatively insensitive to approximations made to make the solution to the problem tractable.

*satisficing*, a portmanteau of satisfy and suffice, was introduced by Herbert A. Simon in 1956 (Simon 1956). The *satisficing* solutions contain optimal solutions.



**Figure 1. 6 The Optimal Solutions are Included in the Satisficing Solutions**

How can designers identify *satisficing* solutions and realize model evolution in the conception of *satisficing*? In this dissertation, the compromise Decision Support Problems (cDSP) is used as the construct and the Adaptive Linear Programming (ALP) algorithm is used as the approximation and solution algorithm to identify *satisficing* solutions, and the Formulation-Exploration framework, Adaptive Linear Algorithm with Parameter Learning (APPLL), and Adaptive-Leveling-Weighting-Clustering (ALWC) algorithm are used to improve decision models for complex systems under *satisficing* strategy.

### ***1.2.2 Why is Satisficing Desired in Engineering Design?***

Why is *satisficing* strategy desired in realizing complex systems and model evolution? Because a complex system may encounter multiple types of the complexity and uncertainty, which make it extremely difficult or impossible to “make things work” using optimizing strategy, thus *satisficing* strategy more often can make things work. “Make things work” in this dissertation means making the system run efficiently, healthily, and sustainably. Their definitions in the context of complex systems and the interpretation in model language are given as follows.

**Efficient** – Stakeholders, users, operators, and designers of the complex system can achieve maximum system performance with minimum expense. In decision models, it means achieving

the objective to the best with minimum computational complexity and no or minimum violation of constraints and bounds.

**Healthy** – A system can reach a status with no potential danger or failure, and low security risk. In decision models, it means no sacrifice or loss of performance that affects system health in order to temporarily meet certain requirements.

**Sustainable** – A system can run constantly. In decision models, it means a model can run in iterations with a relatively good enough and stale performance.

In Table 1.3, we list three typical methods and their features. From reviewing the literature, it is observed that there are four categories of features being well studied: managing complexity, converge condition or solution feature, solution algorithms, and decision support.

The compromise Decision Support Problem (cDSP) is one of the *satisficing* constructs. Mistree and the coauthors (Mistree, Hughes et al. 1993) implement a solution algorithm, Adaptive Linear Programming (ALP) algorithm in DSIDES. Using DSIDES, designers can explore the solution space of nonlinear, nonconvex, multi-goal engineering-design problems, and manage four types of uncertainties (Choi, Austin et al. 2005), but may not find optimal solutions that satisfy sufficient Kuhn-Tucker conditions. The benefit of meeting only the necessary Kuhn-Tucker conditions but not the sufficient conditions is that the solution can be relatively insensitive to the errors embodied in the model or the changes that affect the sufficient Kuhn Tucker conditions. The details are given in Section 2.1. The research gaps using *satisficing* strategy are stated in Section 1.4.2.

Goal Programming is another *satisficing* method but using it does not guarantee *satisficing* solutions. The performance of Goal Programming heavily depends on the solution algorithm or the solver that a designer uses. For example, using NSGA-II or NSGA-III, the nonlinear problems

and nonconvex problems can be well managed when the searching is sufficient, whereas using Conjugate Gradient may not guarantee a feasible solution.

Mathematical programming seeking optimal solutions is the construct using optimizing strategy. Like Goal Programming, the performance of a optimization model regarding the features that a designer can manage depends on the solution algorithm and the solver. Since at the optimal solution point, both the necessary Kuhn-Tucker conditions and the sufficient Kuhn-Tucker conditions must be met, the optimal solution is relatively hard to identified and easy to lose to the infeasible area. he details are given in Section 2.1. The research gaps using optimizing strategy are stated in Section 1.4.1. The differences among cDSP, Goal Programing, and mathematical programming seeking optimal solution are discussed in Section 2.3.1.

**Table 1. 3 Several Representative Methods and Their Features**

cDSP	Construct	Method	Managing complexity					Converge condition / solution feature		Solution algorithms			Decision support	
			Managing nonlinearity	Managing nonconvexity	Managing uncertainty	Managing multi/many objectives (goals)	Managing hard requirements (constraints)	Necessary Kuhn-Tucker conditions	Sufficient Kuhn-Tucker conditions	Interior-point searching	(Linearizing and) obtaining vertex solutions	Comprehensively using interior-point searching and searching for vertex solutions	Can apply visualization tools to facilitate decision making	Managing knowledge discovery and reuse
	cDSP using ALP in DSIDES													
*														
*	Nonconvexity <0.15													
*	Multi-goal													
*														
*	Target on feasible solution space													
*	Return vertex solutions													
*														
*	Tradeoffs among multiple goals; * <i>satisficing</i> design space													
*	Manually Support different types of goal combination													

Mathematical	Goal programming					
Mathematical programming using nonlinear solver	Goal programming using nonlinear	Goal programming using regular solver	cDSP using ALWC	cDSP using ALPPL	cDSP using DCI	cDSP using EMI
*	*					
Depends on the solution algorithm or the solver	Depends on the solution algorithm or the solver	Depends on the solution algorithm or the solver	* Uncertainty in design preferences	* Uncertainty in approximation	* Uncertainty in design capacity	* Uncertainty in parameters, variables
	Many		* Many-goal			
*						
*	*					
*						
*	*					
Depends on the solution algorithm or the solver	Depends on the solution algorithm or the solver	Depends on the solution algorithm or the solver	* Clusters of the many goals	* Tradeoffs among multiple goals; * <i>satisficing</i> design space	* Tradeoffs among multiple goal; * <i>satisficing</i> design space; * DCI distribution	* Tradeoffs among multiple goals; * <i>satisficing</i> design space; EMI distribution
* Tradeoffs among multiple goals; * near Pareto front	* Tradeoffs among multiple goals		* Knowledge on interrelationship among many goals	* Knowledge on reduced move coefficient	* Knowledge on capacity	* Knowledge on margin
* Manually Support changing weight	* Manually Support changing weight vectors		* Automatically Support different types			

Stochastic programming	Mathematical programming using NSGA-III	Mathematical programming using NSGA-II	Mathematical programming using NSGA
* Uncertainty in parameters and variables			
	* Many-goal		* Multi-goal
*			
*			
	*		
*			
* Knowledge on stochasticity			
		* Knowledge on searching strategy	

### 1.3 The Challenges in the Model-based Realization of Complex Systems

The realization of systems is a process of implementing a given input-output behavior. In other words, given an input-output relationship of a system, the realization of the system is a quadruple of (time-varying) matrices. In this dissertation, we focus on the model-based realization of complex systems, that is to use mathematical models to realize the quadruple of matrices. Models are approximations of the physical world, however, there is intellectual disconnection between the physical world and the model world. That is one of the limitations of model-based realization of complex systems. In this dissertation, it is hypothesized that designers can use model evolution to reduce the impact of the intellectual disconnection between the physical world and the model world.

### 1.3.1 Models are Approximations of the Real World

“All models are wrong but some are useful.” (Box 1979, Box and Draper 1987) All models are approximations of the real world. An example of the mapping and correspondence between the physical world and the model world is illustrated in Figure 1.7. No matter how accurate a model is, there is an intellectual disconnection between the physical world and the model world. The assumptions, simplifications, and heuristics used in modeling and solving are means of approximating the physical world. There is not a single model that can capture all the information in the physical world. Some solutions are sensitive to the incompleteness and inaccuracy of a model, whereas other solutions are relatively insensitive to the incompleteness and inaccuracy of the model. Given that, the evolution of the model regarding the improvement on model formulation, approximation, exploration, and evaluation is important in obtaining the solution space that is relatively insensitive to the model inaccuracy.

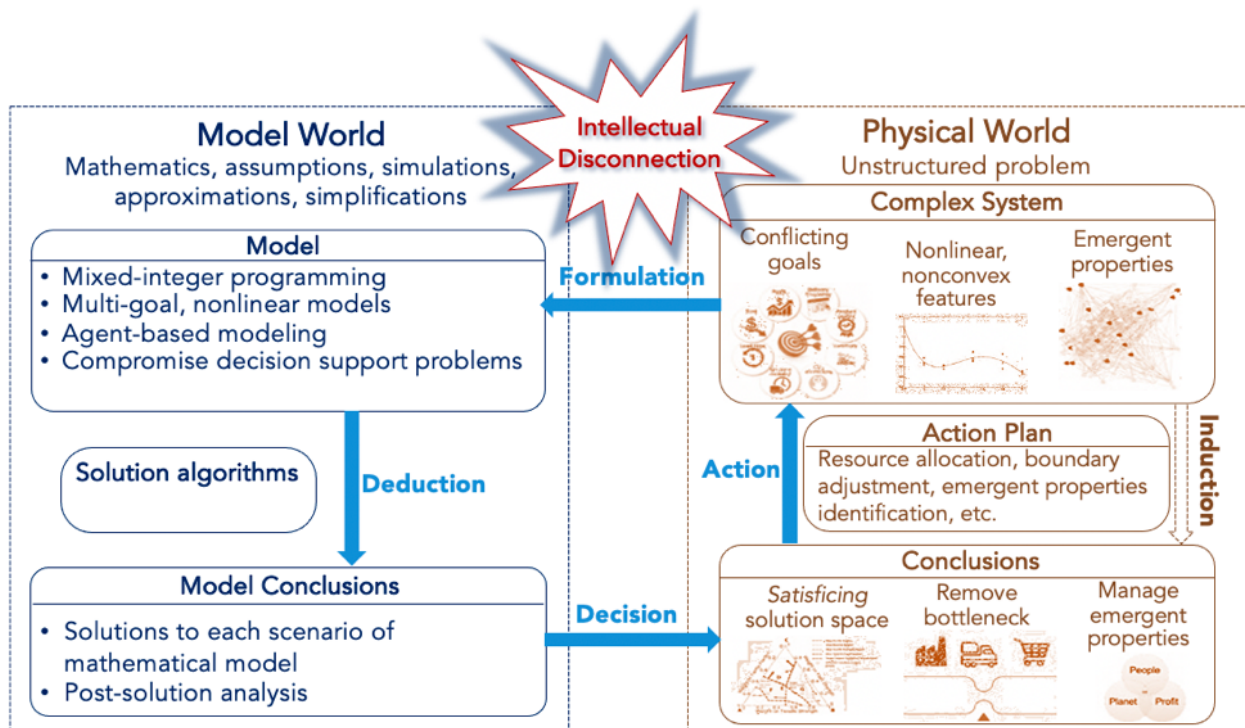


Figure 1. 7 The Correspondence between Physical World and Model World

### ***1.3.2 The Purpose of Model Evolution***

The concept of “evolutionary model” comes from biological evolution, specifically indicating the models of DNA evolution. Inspired by this conception, in this dissertation, we expand “model evolution” to all complex systems. In model evolution, two capabilities of a model are improved.

1) The capability of a model to capture and incorporate more useful information of the physical world, or in other words, model accuracy. 2) The capability of a model to deliver the solutions that are relatively insensitive to uncertainties, or in other words, model robustness. The uncertainties include variation in parameters variations (Type I), variation in decision variables (Type II), uncertainty in model structure (Type III), uncertainty brought by managing the previous three types of uncertainty (Type IV) (Choi, Austin et al. 2005), model errors, model inaccuracies, changes in the environment, etc.

In this dissertation, the purpose of model evolution is improving the insensitivity of the solution space to the uncertainties encountered in the model.

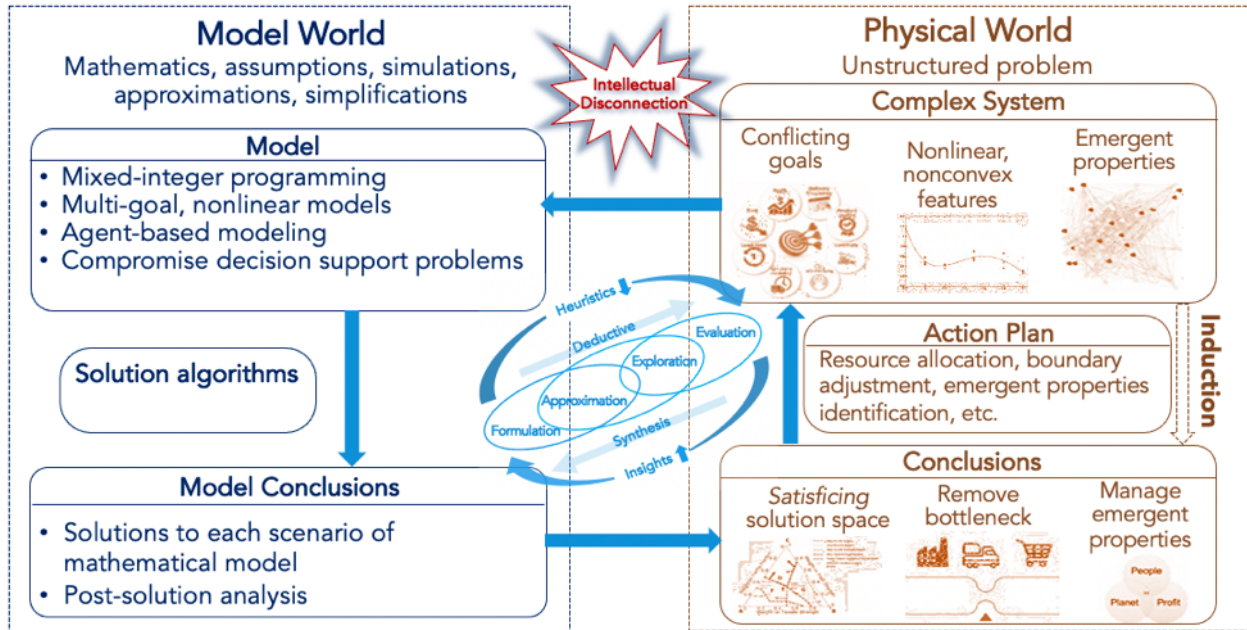
### **1.4 Problem Statement – Problems in both Strategies**

To design complex systems with limited information, multiple interactions among the four stages of the evolution cycle (Figure 1.3) are expected to establish; see Figure 1.9. By establishing connections, information can be passed through different stages of the design, and corresponding actions to improve the design can be taken iteratively. Therefore, the intellectual disconnection between the model world and the physical world can be managed.

As it is introduced in Section 1.2, there are two strategies of designing a complex system, the optimizing strategy and the *satisficing* strategy, yet for both strategies, there are limitations. We summarize them into the research gaps that we dedicate to fill in this dissertation, which are briefly



introduced regarding each strategy – optimizing (Section 1.4.1) and *satisficing* (Section 1.4.2), and the mathematical explanations of the research gaps are introduced in detail in Chapter 2.



**Figure 1. 8 Establishing Connections among Multiple Stages of Engineering-Design Evolution Cycle, Formulation, Approximation, Exploration, and Evaluation**

### ***1.4.1 Problems in Optimizing Strategy***

Albert Einstein said, “So far as the theories of mathematics are about reality, they are not certain; so far as they are certain, they are not about reality.”

Using optimizing strategy, designers assume that their abstractions of the reality, that are their mathematical models, are certain. Even when they incorporate uncertainties in their mathematical models, they assume that those uncertainties are mathematically representable. The general representation of a mathematical model using optimizing strategy is given as follow – Equation 1.1 to 1.3.

### **Given**

$$f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n \quad \text{Equation 1. 1}$$

$$\Omega = \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k\} \quad \text{Equation 1. 2}$$

**Find**

$$x^*: f(x^*) \geq f(x), \forall x \in \Omega \quad \text{Equation 1. 3}$$

The objective of an optimization problem is maximizing a function consisted of decision variables (Equation 1.1), satisfying the constraints (Equation 1.2). Through critically reviewing the literature using optimizing methods in Section 1.2 and identifying the limitations of decision models in Section 1.3, we conclude the problems using optimizing strategy in Table 1.4.

**Table 1. 4 Problems in Methods of Optimizing Strategy**

<b>Uncontrollable factors</b>	<b>List of Requirements</b>	<b>Problems (phenomenon)</b>	<b>Mathematical explanation location</b>
Boundary changes	Accurate and exact boundary	The optimal solution to the mathematical model may not work for its corresponding physical engineering-design problem.	Section 2.1-2.3
Design preferences evolve	Evolving design preferences can be captured, quantified, and represented in the model		
Unpredictable, unparameterizable uncertainties	Uncertainties can be parameterized and predicted		
Unknown interactions	Interactions can be represented in the model		
Emergent properties	Emergent properties can be captured and incorporated into the model		

As the boundary of the physical system changes, design preferences evolve, unpredictable uncertainties, unknown interactions among subsystems or between the system and its environment cannot be represented in the model, emergent properties add complexity to the modeling and solution analysis, etc. These are the factors that we cannot control or percept during the process of abstracting physical systems into models and interpreting model solutions into physical phenomena. As we have limited knowledge on the association amongst those factors, the design may fail. The designers may lose the optimal solution. In other words, the optimal solution to the

mathematical model may not work for its corresponding physical engineering-design problem. Why? Because the designers seek optimal solutions to wrong models. Therefore, using optimizing strategy to manage engineering-design problems is like trying to maximize the optimality of the solution to a wrong model, without the awareness of the inaccuracy of the model and the consideration of the robustness of the solutions to the inaccuracy. However, this is the phenomenon of the research gap. The mathematical explanation of the research gap in in Section 2.1 to 2.3.

#### ***1.4.2 Problems in Satisficing Strategy***

Using *satisficing* strategy, designers are aware that their abstractions of the reality, that are their decision models, may be incomplete or in accurate. So, they seek solutions that are “good enough” and relatively insensitive to the errors of the model, instead of optimal. One of the representations of a decision model under *satisficing* strategy is given as follow – Equation 1.4 to 1.6

##### **Given**

$$f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n \quad \text{Equation 1. 4}$$

$$\Omega = \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k, \} \quad \text{Equation 1. 5}$$

##### **Find**

$$x^s: \mathcal{P}_{x \in \Omega}(f(x) = \text{Target}) \quad \text{Equation 1. 6}$$

The merit function of a decision model using *satisficing* strategy is identifying the nearest projection of the objective function  $f(x)$  onto the feasible space bounded by constraints. Or in other words, the aim is minimizing the deviation between the target and the actual achieved value of a goal, as Equation 1.7-1.9

##### **Given**

$$f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n$$

**Equation 1. 7**

$$\Omega = \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k, f(x) + d^- - d^+ = Target\}$$

**Equation 1. 8**

**Find**

$$d^*: d^*(x) \leq d(x)$$

**Equation 1. 9**

Although using *satisficing* strategy, designers may identify solutions that are relatively insensitive to model errors and uncertainties, there are limitations in methods under *satisficing* strategy. Through critically reviewing the literature using *satisficing* methods in Section 1.2 and identifying the limitations of decision models in Section 1.3, we conclude the problems using *satisficing* strategy in Table 1.5.

**Table 1. 5 Problems in Methods of *Satisficing* Strategy**

Uncontrollable factors	List of Requirements	Problems (phenomenon)	Mathematical explanation location
Boundary changes	Solutions are insensitive to boundary variations	Adding too much buffer to ensure feasibility; relying on heuristics, metaheuristics, and domain knowledge to make decisions; no information passing through different stages of the design; no mechanism to evaluate and update the heuristics and metaheuristics.	Section 2.1-2.3
Design preferences evolve	Solutions are insensitive to Design preferences evolving		
Unknown interactions	Solutions are insensitive to unknown interactions		
Emergent properties	Solutions are insensitive to the emergent properties even they cannot be incorporated into the model		
Relying on heuristics and metaheuristics to make rules	Accurate and exact boundary		
Relying on domain knowledge to make rules	Evolving design preferences can be captured, quantified, and represented in the model		
No information passing through different stages of engineering designs	Uncertainties can be parameterized and predicted		

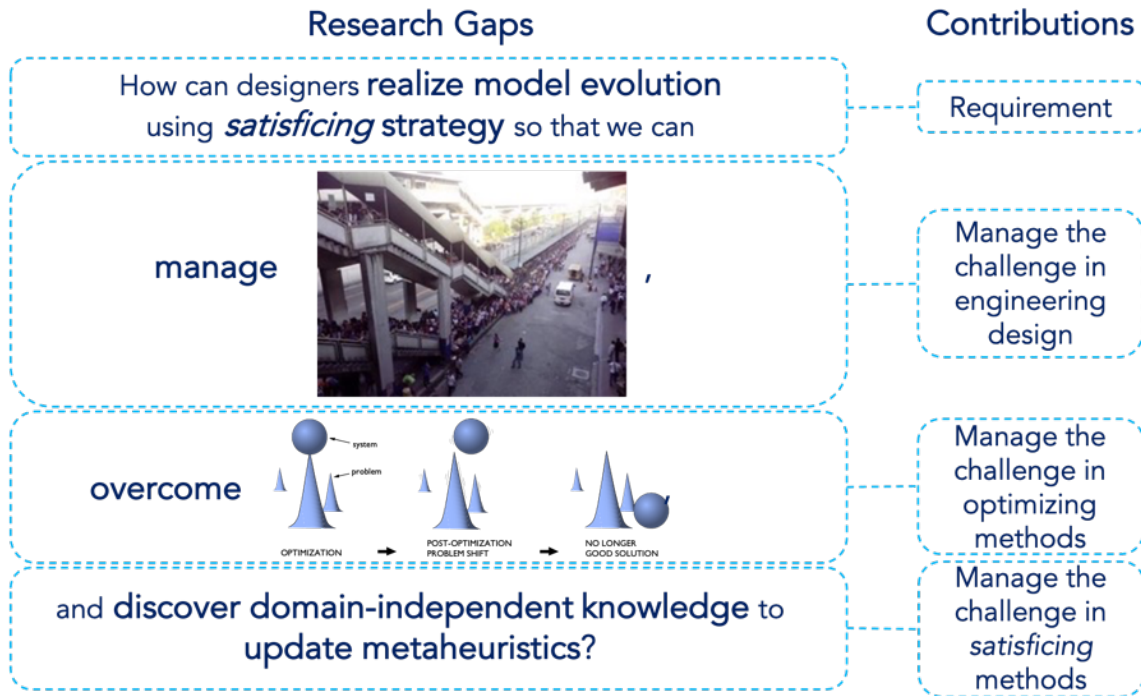
Given the problems in both strategies, we summarize the research gaps and pose the hypotheses in Section 1.5.

## **1.5 Research Gaps and Hypotheses**

### ***1.5.1 Research Gaps***

Based on the problems in two design strategies, optimizing and *satisficing*, we summarize the research gaps to be filled in this dissertation, that is, how can designers manage the problems in both strategies, meanwhile, finish the tasks of engineering design? The research gaps and the potential contributions by filling the research gaps are illustrated in Figure 1.9.

- ***Research Gaps – How can designers realize model evolution using satisficing strategy so that they can manage chaos in the physical world, reduce the risk of losing an optimal solution, and discover domain-independent knowledge to update metaheuristics?***



**Figure 1. 9 The Research Gaps and the Potential Contributions by Filling the Research Gaps**

### 1.5.2 Hypotheses to Bridge the Research Gaps

In this dissertation, it is hypothesized that *by connecting the multiple stages of design and passing information through them, designers can improve their decision models in iterations*, which in this dissertation is defined as “model evolution.” During the model evolution, designers can incorporate more chaos in the physical systems into their decision models and manage their impact on the solution space, identifying solutions that are relatively insensitive to errors and uncertainties that the decision models may encounter, and discover domain-independent knowledge to update metaheuristics.

**Table 1. 6 The Research Gaps (RG) and Hypotheses (H)**

Chapter	Ch1	Ch2	Ch3	Ch4-7	Ch8	Ch9
<b>Actions</b>	<p><i>RG: How can designers realize model evolution using satisficing strategy so that they can manage chaos in the physical world, reduce the risk of losing an optimal solution, and discover domain-independent knowledge to update metaheuristics?</i></p> <p><i>H: by connecting the multiple stages of design and passing information through them, designers can improve their decision models in iterations.</i></p>	<p>RD RQ SH</p>	<p>TVe M</p>	<p>EVe SQT AQ</p>	<p>CQ EVa</p>	<p>TE</p>
<b>Nomenclature</b>	<p>RG – give research gaps  H – give hypotheses  RD – tie to robust design  RQ – pose research questions  SH – specify hypotheses  TVe – theoretically verify hypotheses  M – introduce methods  EVe – empirically verify hypotheses  SQT – specify research questions in the context of test problems  AQ – answer research questions  CQ – closure the answers to research questions  EVa – empirically validate hypotheses  TE – theoretically extend the research</p>					

**1.5.3 Expected Contribution by Testifying the Hypotheses**

By proving the hypotheses, it is expected to contribute a design method that allows designers to manage the following attributes by exploring the solution space and identify a set of solutions that weighs all the attributes, which is defined as *satisficing* solution space in this dissertation; see Figure 1.10.

**Accurate** – the mapping from the physical system to a mathematical model is relatively accurate and the accuracy can be improved during the exploration of the solution space.

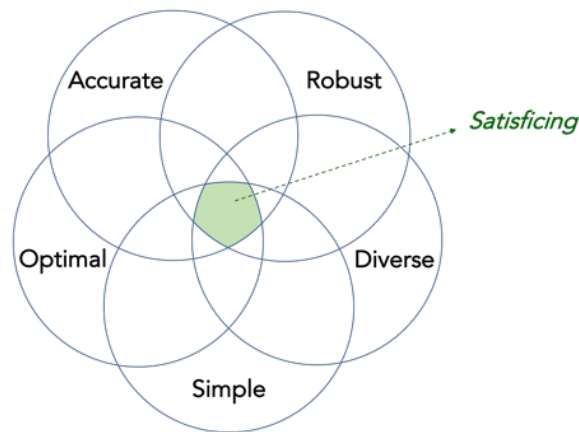
**Robust** – the *satisficing* solutions are relatively insensitive to the errors and uncertainties embodied in the model. There are four types of uncertainty (Choi, Austin et al. 2005), noise factors, control factors, uncertainties in the model structure, and uncertainties in the process chain of managing

the first three types of uncertainty. A thorough introduction of the four types of uncertainty and the corresponding robust design methods are given in Section 2.4.1.

**Diverse** – the solutions identified as *satisficing* solutions are relatively different from each other, which allows designers to have more alternatives in a variety of situations.

**Simple** – the computational complexity of the method is relatively low.

**Optimal** – the achieved value of the goals is close enough to the target of the goals.



**Figure 1. 10 Expected Contributions**

Using interior-point searching algorithms, designers may focus on identifying optimal and diverse solutions. Stochastic optimization methods allow designers to attempt taking into account the accuracy, optimality, and diversity of the solutions. However, the accuracy and exhaustiveness of the randomness of a stochastic model may rely on assumptions and heuristics which may not be correct or accurate. Using the methods belong to the *satisficing* strategy methods such as cDSP-ALP-DSDIES (in Table 1.3), designers have mechanisms to obtain robust solutions and maintain an acceptable level of computational complexity, but they do not improve the accuracy of their decision models.



*In this dissertation, the contribution is to identify a satisficing solution space to the decision model of complex engineering-design problems and provide decision support based on knowledge of the tradeoffs among the five attributes.*

## **1.6 Plan of Verification and Validation**

The plan of verification and validation are illustrated in Figure 1.11. In Chapter 1, gives the context of the dissertation – models are approximations of the physical world thus model evolution is required so as to improve the robustness of the model realization. In Chapter 2, gives the theoretical foundation and justified research questions. In Chapter 3, the overview of the methods is described. In Chapter 4 to Chapter 7, test problems are used to verify the proposed methods and algorithms, which are the new knowledge and contributions in this dissertation. The research questions are answered one by one in Chapter 4 to 7. In Chapter 8, summarized the answers to the research questions and the utility of the proposed methods to other problems. In Chapter 9, summarized the contributions, limitations, and research way forward (the career proposal).

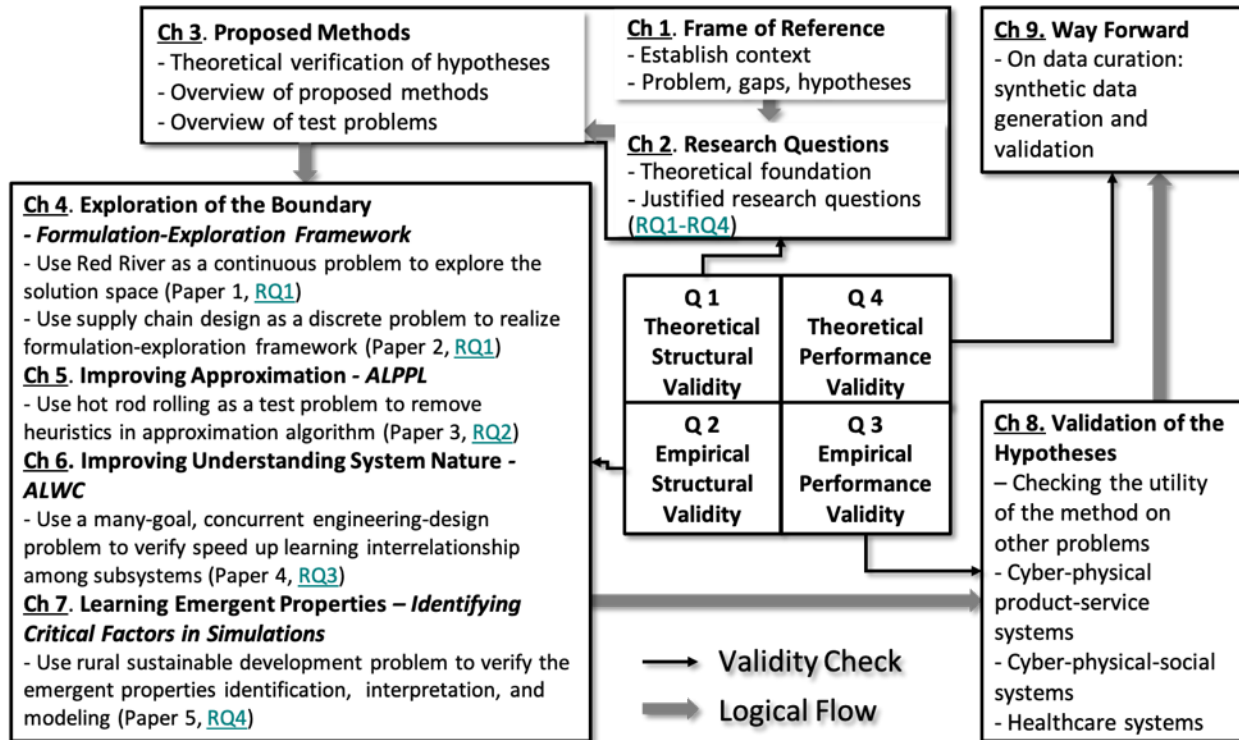


Figure 1. 11 Dissertation Layout and Plan of Verification and Validation

### 1.7 Organization of The Dissertation

There are nine chapters in this dissertation. An overview of this dissertation is presented as roadmap in Figure 1.12. The figure is intended to help navigate through the dissertation and develop an overall picture as to what is discussed in each chapter thereby establish context.

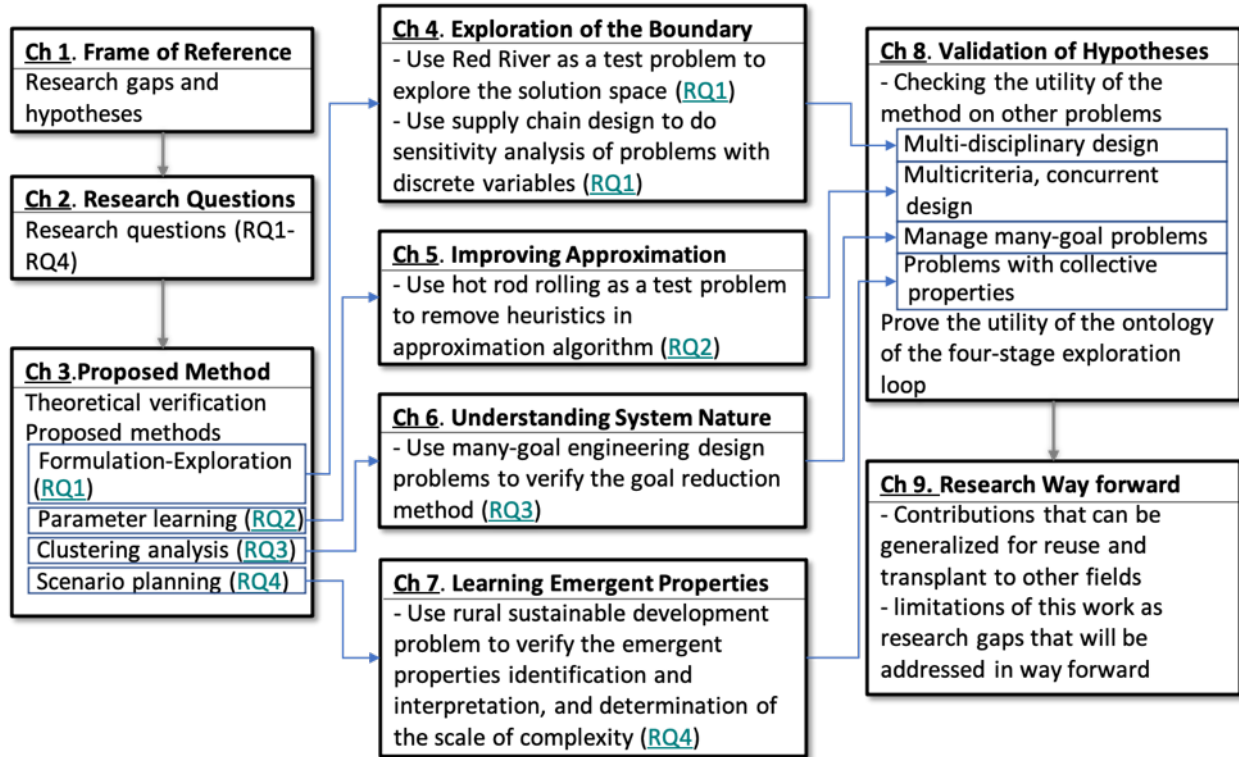


Figure 1. 12 Organization of the Chapters

### 1.8 Role of Chapter 1 in this Dissertation

In Chapter 1, we establish the context of this dissertation. We answer the “why question” – “why do we need model evolution. give the motivation of model evolution and describe why we manage complex systems in a certain way – using satisficing methods.

In this chapter, we give the context of the realization of complex systems – introduction to the characteristics and challenges of model-based realization of the complex systems. By analyzing the methods in optimizing strategy and *satisficing* strategy, we identify research gaps of both methods and pose the primary research question. The plan of verification and validation and the organization of the dissertation are illustrated.

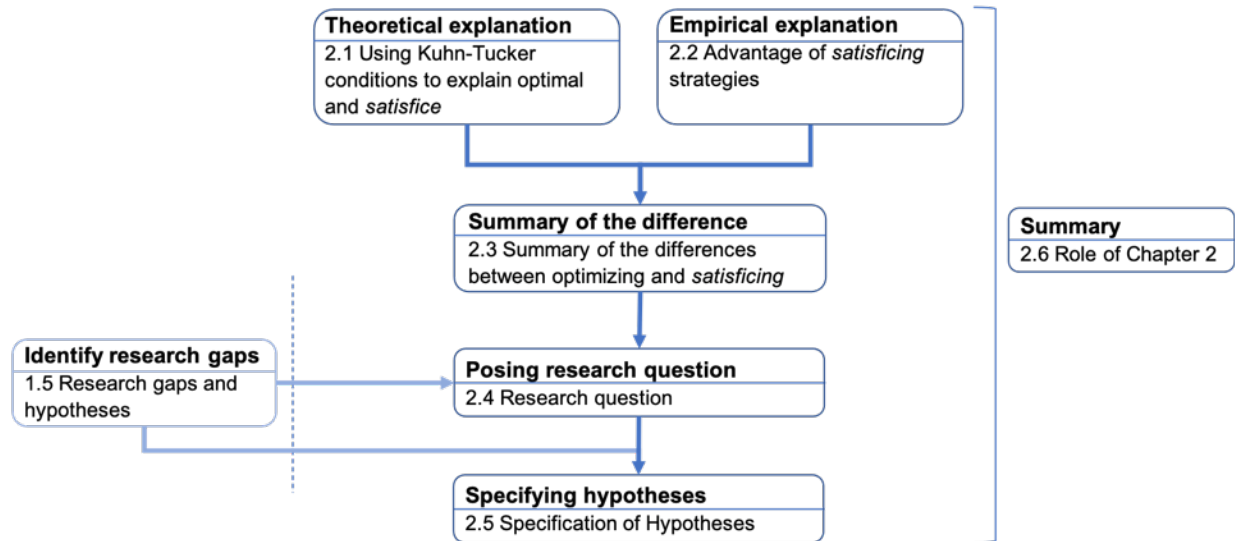
This chapter is revisited for checking structural soundness of the dissertation where literature review, design approach, developed method, and validation of hypotheses are discussed in following chapters.

## CHAPTER 2 RESEARCH QUESTIONS: HOW CAN WE REALIZE MODEL EVOLUTION

### – THEORETICAL FOUNDATION AND JUSTIFIED RESEARCH QUESTIONS

*In Chapter 2, the primary research question is justified into several research questions in the context of robust design and the model evolution. The “how question” is answered – “how can designers realize the model evolution using satisficing strategy?” It is further explained why we suggest designers realize model evolution in a certain way – why satisficing strategy is chosen, and specifically, cDSP, ALP, and DSIDES are used to realize satisficing strategy and selected as the foundational method to process the tasks in the model evolution.*

In Chapter 2, see Figure 2.1, in Section 2.1, the mathematical explanations of the differences between optimizing and *satisficing* strategy are given; in Section 2.2, five toy problems are used to illustrate the advantages of using *satisficing* strategy in engineering design; in Section 2.3, the mathematical and practical differences are summarized; based on the discussions, in Section 2.4, the primary research question is justified in the context of “what should we do” – the robust design – and “what would we do” – the model evolution; the role of Chapter 2 is reviewed in Section 2.5.



**Figure 2. 1 Organization of Chapter 2**

The main issues in the design of complex systems are: (1) dealing with high complexity, uncertainty in rapidly changing requirements, (2) capability of the method in order to capture features of and analyze the system behavior, and (3) operability of automating the method and outputting reusable knowledge. In this chapter, literature is reviewed regarding the forward mentioned issues and research opportunities are located in this dissertation and in the future work (career proposal).

## **2.1 Using Kuhn-Tucker Conditions to Explain Optimal and *Satisfice***

### ***2.1.1 The History of the Kuhn-Tucker Conditions***

The Kuhn-Tucker approach or Karush-Kuhn-Tucker (KKT) approach generalizes the method of Lagrange multipliers. A nonlinear problem with constraints can be represented as a function – the Lagrange function. Harold W. Kuhn and Albert W Tucker proposed the Kuhn-Tucker approach in 1951 (Kuhn and Tucker 1951). The optimal point of the Lagrange function is a saddle point, so the Kuhn-Tucker approach is also known as saddle-point Theorem. Later it was found out that

William Karush had summarized the necessary conditions in his master's thesis in 1939 (Karush 1939). So, Kuhn-Tucker conditions is also named as Karush-Kuhn-Tucker (KKT) conditions.

### ***2.1.2 Necessary and Sufficient Kuhn-Tucker Conditions***

The KKT conditions include first-order necessary conditions (Equation 2.1-2.5) and second-order sufficient conditions (Equation 2.6-2.8).

First-order necessary conditions

Stationary:

$$\nabla f(x^*) + \sum_{i=1}^m \mu_i \nabla g_i(x^*) - \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x^*) = 0 \quad \text{Equation 2. 1}$$

Primal feasibility:

$$g_i(x^*) \geq 0, \forall i = 1, \dots, m \quad \text{Equation 2. 2}$$

$$h_j(x^*) = 0, \forall j = 1, \dots, \ell \quad \text{Equation 2. 3}$$

Dual feasibility:

$$\mu_i \geq 0, \forall i = 1, \dots, m \quad \text{Equation 2. 4}$$

Complementary slackness:

$$\mu_i g_i(x^*) = 0, \forall i = 1, \dots, m \quad \text{Equation 2. 5}$$

Second-order sufficient conditions:

For the Lagrangian:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) \quad \text{Equation 2. 6}$$

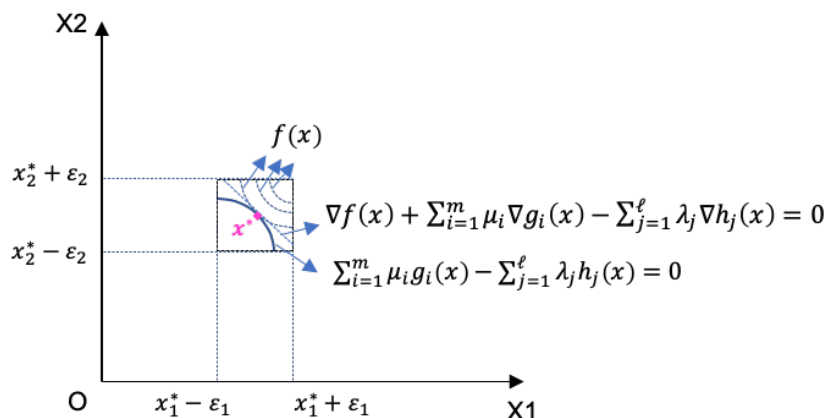
$$\Rightarrow s^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) s \geq 0, \text{ where } s \neq 0 \quad \text{Equation 2. 7}$$

And

$$[\nabla_x g_i(x^*), \nabla_x h_j(x^*)]^T s = 0 \quad \text{Equation 2. 8}$$

### 2.1.3 The Physical Meaning of the Kuhn-Tucker Conditions

The essence of the necessary conditions is that at the solution point  $x^*$ , where both the primal and the dual are feasible, the gradient vector of the objective  $\nabla f(x^*)$  can be represented as the non-zero linear combination of the gradient matrix of all equality constraints  $\nabla h_j(x^*)$  and the active inequality constraints<sup>6</sup>  $\nabla g_i(x^*)$  for  $i$  iff  $g_i(x^*) = 0$ , as it is shown in Figure 2.2.



**Figure 2. 2 The first-order necessary Kuhn-Tucker conditions are satisfied at  $x^*$**

The essence of the sufficient conditions is that at the solution point  $x^*$ , there exists a nonzero vector  $s$  that is orthogonal to the gradient matrix of all active inequality and equality constraints, such that the second-order matrix of the Lagrange's equation with respect to decision variables  $x^*$  and

---

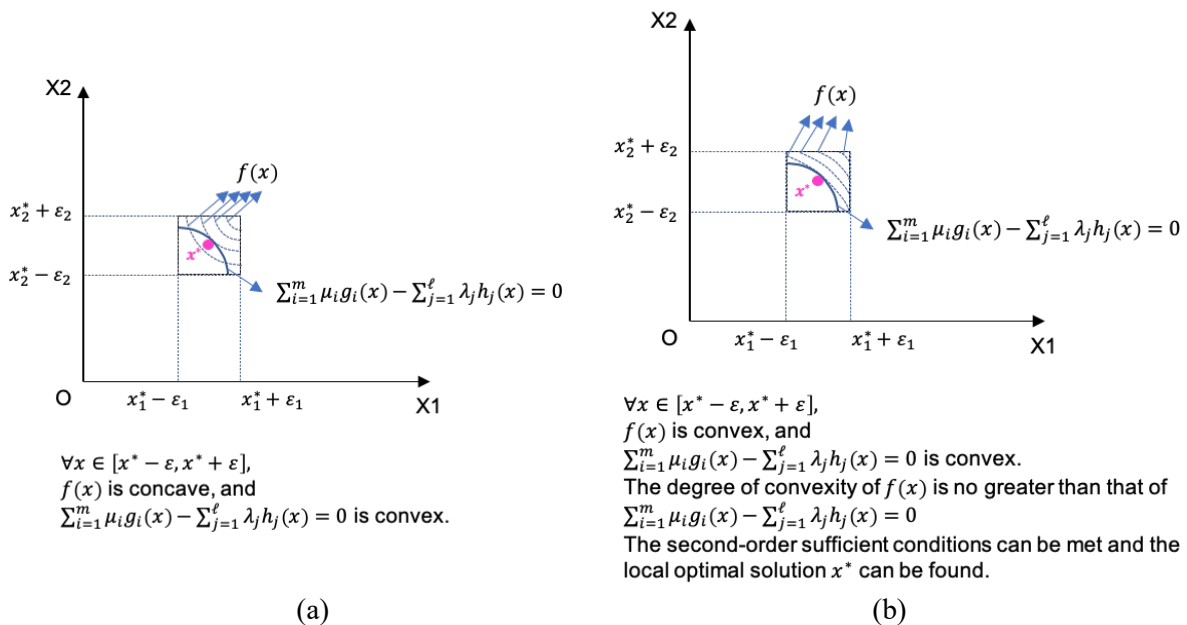
<sup>6</sup> In this dissertation, we define an active constraint (or an active inequality constraint) as the constraint with zero slack or surplus when plugging in the solution point. Or, in other words, for an inequality constraint, when being plugged in the solution point, its left-hand side value equals to its right-hand side value, then such an inequality constraint is an active constraint; it is also known as binding constraint; see "DEFINITION OF TERMS."



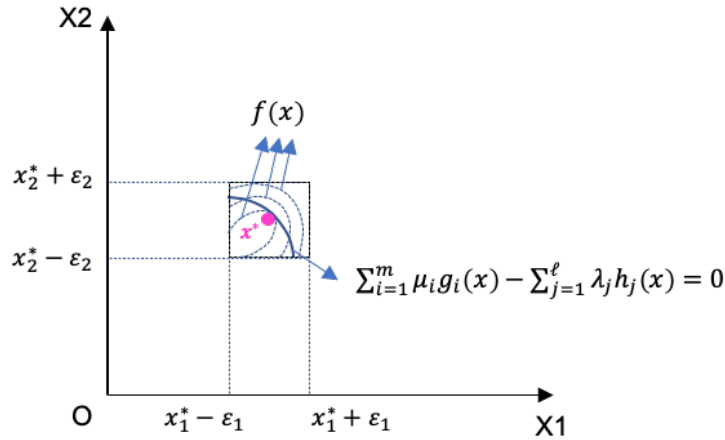
Lagrange multipliers  $\lambda^*$  and  $\mu^*$  is conditionally positive semidefinite:  $s^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) s \geq 0$ ,  $\forall s \in \mathcal{S}$ .

Why is it conditionally positive semidefinite? What is the condition?

The second-order sufficient condition requires that the Lagrange equation is convex at  $x^*$ , that is, in a local range that contains the solution point  $x^*$ , the convexity degree of the objective should not exceed the convexity degree of the constraints combined by Lagrange multipliers. See Figure 2.3. However, the second-order sufficient conditions significantly hinder the access of a solution for the problems with an objective with a relatively higher degree of convexity, that is, the convexity degree of the objective is higher than the convexity degree of the feasible set bounded by constraints boundary and bounds, such as the case in Figure 2.4.



**Figure 2. 3 The convexity requirements for satisfying the second-order sufficient Kuhn-Tucker conditions**



$\forall x \in [x^* - \varepsilon, x^* + \varepsilon]$ ,  
 the degree of convexity of  $f(x)$  is greater than that of  
 $\sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^l \lambda_j h_j(x) = 0$   
 The second-order sufficient conditions cannot be met and  
 the local optimal solution  $x^*$  cannot be found.

**Figure 2. 4 Lagrange multipliers fail to identify an optimal for a highly convex objective**

#### 2.1.4 Assumptions behind Kuhn-Tucker Conditions

There are assumptions behind applying the Kuhn-Tucker approach to solve a nonlinear programming problem, especially when designers seeking optimal solutions which meet both necessary and sufficient Kuhn-Tucker Conditions. These assumptions are also mathematical requirements for the problem that can be applied with Kuhn-Tucker conditions to obtain an optimal solution. When we use Kuhn-Tucker approach or Lagrange function to solve a nonlinear problem, we automatically accept these assumptions.

***Assumption/Requirement 1 – mathematical models are 100% complete and accurate representations of physical problems.***

To use Lagrange functions to solve nonlinear problems, one naturally assumes that the mathematical models are complete and accurate and capture all the necessary details of the physical problems, and all segments of a model have the same level of fidelity. Thus, an optimal

solution to a mathematical problem is also optimal to its corresponding physical problem or at least feasible for the physical problem.

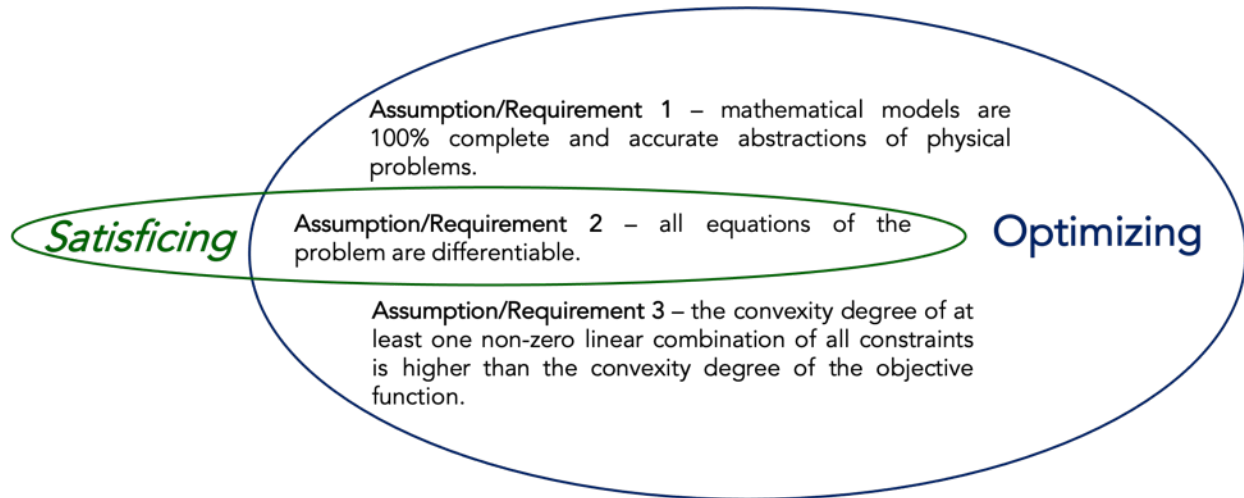
***Assumption/Requirement 2 – all equations of the problem are differentiable.***

Although according to Kuhn-Tucker conditions, it is only required that, within a small finite area around the optimal solution, the objective function, active constraints, and equality constraints should be and differentiable, as we need to solve the Lagrange function globally to obtain the optimal solution, it is required that all equations of the problem should be differentiable.

***Assumption/Requirement 3 – the convexity degree of at least one non-zero linear combination of all constraints is higher than the convexity degree of the objective function.***

This assumption is especially for Kuhn-Tucker sufficiency. The second-order sufficient conditions only guarantee to find optimal solutions to the problems as those in Figure 2.2 but cannot find an optimal solution to the problem as shown in Figure 2.3.

In summary, when using optimizing strategy seeking optimal solution to a decision model, designers have all three assumptions, whereas when using satisficing *strategy*, designers have only one assumption. The assumptions of the two strategies are illustrated in Figure 2.5.



**Figure 2. 5 The Assumptions When Using the Optimizing Strategy and *Satisficing* Strategy**

## 2.2 Advantages of *Satisficing* Strategy

### *- Advantages in Each Stage of Model Evolution*

Using optimizing strategy, designers use the Lagrange function to find solutions that satisfy both necessary and sufficient Kuhn-Tucker conditions, whereas using *satisficing* strategy, designers only satisfy necessary Kuhn-Tucker conditions. Therefore, when seeking optimal solutions, designers have three assumptions, or in other words, the mathematical model needs to meet the three offered-mentioned requirements. In other words, when seeking optimal solutions, designers accept the three assumptions behind Kuhn-Tucker Conditions: models are 100% complete and accurate, all equations are differentiable, and the problem is convex.

When seeking *satisficing* solutions, the assumptions are different. Designers are aware that their models can be incomplete and inaccurate, so they aim at good enough solutions that are relatively insensitive to the error and incompleteness of the models. When their decision models are discrete or non-convex, we describe the difference between optimizing and *satisficing* strategy in Section 2.3.

To illustrate the differences between the two strategies regarding the returned solutions when managing engineering design problems, especially complex systems design and improvement, first, we use five representative “toy problems<sup>7</sup>” to illustrate the differences in appearance between optimizing and *satisficing*. Then, we analyze in principle, why there are such differences.

There are four advantages using *satisficing* strategy, which in this dissertation, is realized by using cDSP, ALP and DSIDES. The advantages include the advantage in formulation, approximation, solution, and exploration. In Table 2.1, we summarize the advantages of using *satisficing* in the four stages in engineering design. The illustrations of the differences through examples are given from Section 2.2.1 to Section 2.2.10.

**Table 2. 1 The Advantages of Realizing *Satisficing* Strategy Using cDSP and ALP in the Each Stage of Engineering Design**

Stage	Feature	Advantage	Introduction and Discussion
Formulation	Using Goals and Minimizing Deviation Variables Instead of Objectives	At a solution point, only the necessary Kuhn-Tucker conditions are met, whereas the sufficient Kuhn-Tucker conditions do not have to be met. Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and unparameterizable uncertainties.	Section 2.2.7
Approximation	Using second-order sequential linearization	Designers can have a balance between linearization accuracy and computational complexity.	Section 2.2.5 and 5.2.1
	Using accumulated linearization	Designers can manage nonconvex problems in a way, and deal with highly convex, nonlinear problems relatively more accurately.	

---

<sup>7</sup> The problems in Section 2 are “toy problems” to illustrate the difference of the results that the methods in the two strategies can return. They are separated from the test problems in Chapter 4, 5, 6, 7, and 9. The test problems in the later chapters are used to verify the proposed method and the new knowledge in this dissertation.

<b>Exploration</b>	Combining interior-point searching and vertex searching	Designers can avoid being stuck into local optimum to some extent and identify <i>satisficing</i> solutions relatively insensitive to starting points changing.	Section 2.2.3
<b>Evaluation</b>	Allowing some violations of soft requirements, such as the bounds of deviation variables	Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility. As a result, goals and constraints with different scale can be managed	Section 2.2.9

### 2.2.1 Possible Features of Engineering-Design Problems

For engineering-design problems, there are typical features that often take place. In this dissertation, we define the *satisficing* strategy is the strategy that by using which, the formulation construct and solution algorithm allow designers to manage those typical features of engineering-design problems. In this dissertation, we identify six features, listed in the first column of Table 2.3, as typical features that designers often encounter in engineering-design problems. We choose the five toy problems and add one more complexity to each toy problem. The toy problems are very simple, but they represent some basic, primary complexity of those fancy, complex engineering-design problems. So, we use these toy problems to do experiments and comparisons. There can be much more features, challenges, and difficulties of complex engineering-design problems. In this section, we only tackle five basic features because a lot of complexities are combinations and evolution of the six features in different times and spaces. The *satisficing* methods used here can be foundations or cornerstones of more complex methods or technologies.

The methods we use in each strategy are listed in Table 2.3. In the “scipy.optimize” package, besides the three algorithms (COBYLA, Trust-constr, and SLSQP), there are other seven algorithms, Nelder-Mead, Powell, CG (Conjugate Gradient), BFGS, Newton-CG, L-BFGS-B, and TNC. However, Nelder-Mead, Powell, CG, and BFGS cannot manage problems well with constraints. Newton-CG, L-BFGS-B, and TNC require Jacobian which is not user-friendly for

engineering designers, as for nonlinear engineering-design problems, usually, designers do not want to identify the Jacobian matrix of the nonlinear constraints and formulate the problems by presenting the coefficient matrix. Therefore, we use COBYLA, Trust-constr, and SLSQP to solve the toy problems with one or more the of features in Table 2.2.

*We use NSGA II/III as a benchmark or verification method.* NSGA II/III is a well-known solution algorithm belongs to the optimizing category. It is a fast sorting and elite multi objective genetic algorithm. Unlike the single objective optimization technique, NSGA II/III simultaneously optimizes each objective without being dominated by any other solution. As NSGA II/III can manage all six features listed in Table 2.2, we use it as a benchmark tool to verify the solution quality of the *satisficing* algorithm.

*Since NSGA II/III can manage the five typical features of engineering-design problems, why do we study satisficing algorithm?* Because NSGA II/III has drawbacks that prevent designers from further exploring the solutions space and improving the decision models.

First, NSGA II/III cannot give designers insight on the nature of the decision model or the possible ways to improve the model. NSGA II/III is an interior searching algorithm, which uses metaheuristics to search for solutions that generationally improve the optimality and diversity of the solutions, but for information, such as the bottleneck of the model, or the sensitivity of each part of the model, or anything else that may indicate model improvement, cannot be provided along with the searching.

Second, the performance of NSGA II/III (include convergence speed, optimality of solutions, and diversity of solutions) is sensitive to hyperparameter setting. Typical hyperparameters, the population size and generation number, should be predefined by designers. However, designers

only have the idea that a larger population size or a larger generation number can return better solutions, but they may not have a clue how large is “good enough.” In Toy Problem II and III, we can show how different the solutions can be when setting the population as 20 and 50 for the same problem.

Third, NSGA II/III requires much more computational power than satisficing algorithms such as the Adaptive Linear Programming (ALP) algorithm. we will discuss it in detail in the “summary of observations” for each toy problem in this section.

The toy problems, solutions using different algorithms, comparisons, and conclusions are given as follows.

**Table 2. 2 The Features of the Toy Problems (TP)**

Toy Problem (TP)	I	II	III	IV	V
<b>Feature</b>					
Two objectives	*	*	*	*	*
Nonlinear	*	*	*	*	*
Non-convex		*	*	*	*
Objectives with various units (scale)			*	*	*
Target of goals with various levels of achievability				*	*
More than two objectives					*

**Table 2. 3 Methods for Comparison the Two Strategies**

Strategy	Optimizing	<i>Satisficing</i>
<b>Item</b>		
Model formulation construct	Mathematical programming or Goal programming	Compromise Decision Support Problem
Solution algorithm	Constrained Optimization by Linear Approximation (COBYLA) algorithm	Adaptive Linear Programming (ALP) algorithm
	Trust-region constrained (trust-constr) algorithm	
	Sequential Least Squares Programming (SLSQP) algorithm	
	Nondominated Sorting Generation Algorithm II/III (NSGA II/III)	
Solver	Python Scipy.optimize	DSIDES



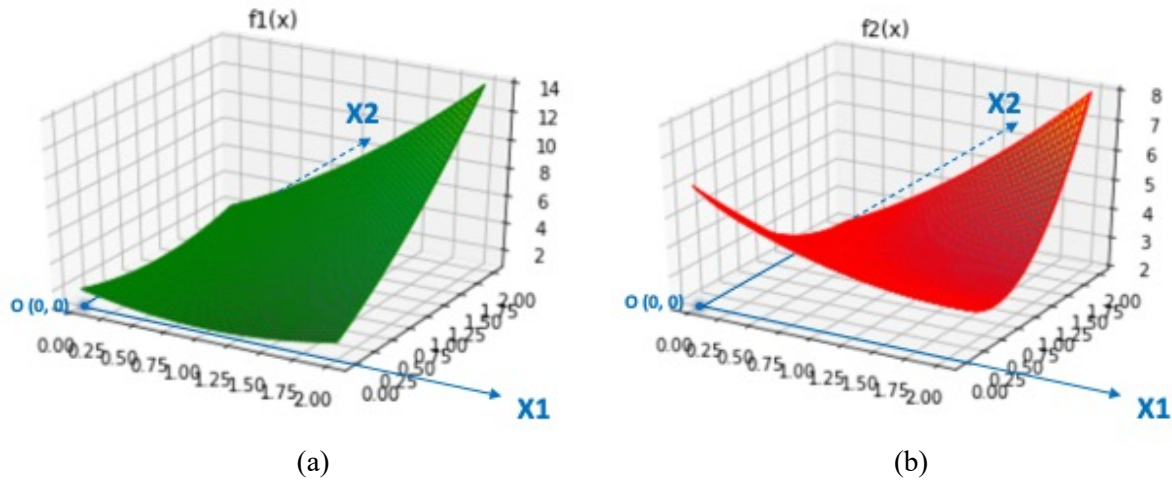
### 2.2.2 Toy Problem \*I\* (TP-I)

– Problem with multiple, nonlinear objectives (goals)

Formulation: the two types of formulation of the five toy problems are given in Table 2.4. We give the optimization formulation and its corresponding compromise DSP formulation of the five Toy Problem \*I\* in Table 2.4. The two objectives are visualized in the solution space (decision-variable space, X-Plane) as Figure 2.6 shows. The solutions to the problem using the corresponding algorithms are given in Table 2.6.

**Table 2. 4 The Optimization Model and Compromise DSP of TP-I**

Strategy TP	Optimizing	Satisficing
I	<p><u>Objective Functions</u></p> $f_1(x) = (x_1 - 1)^2 + (x_2 - 1)^2 + 3 \cdot x_1 \cdot x_2$ $f_2(x) = \frac{1}{2} \cdot (x_1 - 2)^2 + (x_2 - 2)^2 + x_1 \cdot x_2^2$ <p><u>Constraints and Bounds</u></p> $s. t. \begin{cases} x_1 \cdot x_2 \leq 1 \\ 0 \leq x_1 \leq 2 \\ 0 \leq x_2 \leq 2 \end{cases}$ <p><u>Combination of Objective Functions</u></p> $Max \sum_{i=1}^2 w_i \cdot f_i(x)$	<p><u>Given</u></p> $x_1, x_2, d_1^{\pm}, d_2^{\pm}$ $f_1(x) = (x_1 - 1)^2 + (x_2 - 1)^2 + 3 \cdot x_1 \cdot x_2$ $f_2(x) = \frac{1}{2} \cdot (x_1 - 2)^2 + (x_2 - 2)^2 + x_1 \cdot x_2^2$ <p><u>Find</u></p> $x_1, x_2, d_1^{\mp}, d_2^{\mp}$ <p><u>Satisfy</u></p> <p><u>Goals:</u></p> $\frac{f_1(x)}{14} + d_1^- = 1$ $\frac{f_2(x)}{8} + d_2^- = 1$ <p><u>Constraints:</u></p> $x_1 \cdot x_2 \leq 1$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><u>Bounds:</u></p> $0 \leq x_1, x_2 \leq 2$ $0 \leq d_1^{\pm}, d_2^{\pm} \leq 1$ <p><u>Minimize</u></p> $Z = \sum_{i=1}^2 w_i \cdot d_i^-$



**Figure 2. 6 The Two Objective Functions in the X-f(X) Space of TP-I**

Target of the goals: for compromise DSP, a right-hand side value is needed for each objective – designers need to assign a target value for each goal and minimize the deviation between the achieved value and the target of the goal. Assuming that based on the domain expertise, or the results from exploring of the solution space, we can determine the target of Goal 1 and Goal 2 to be 14 and 8.

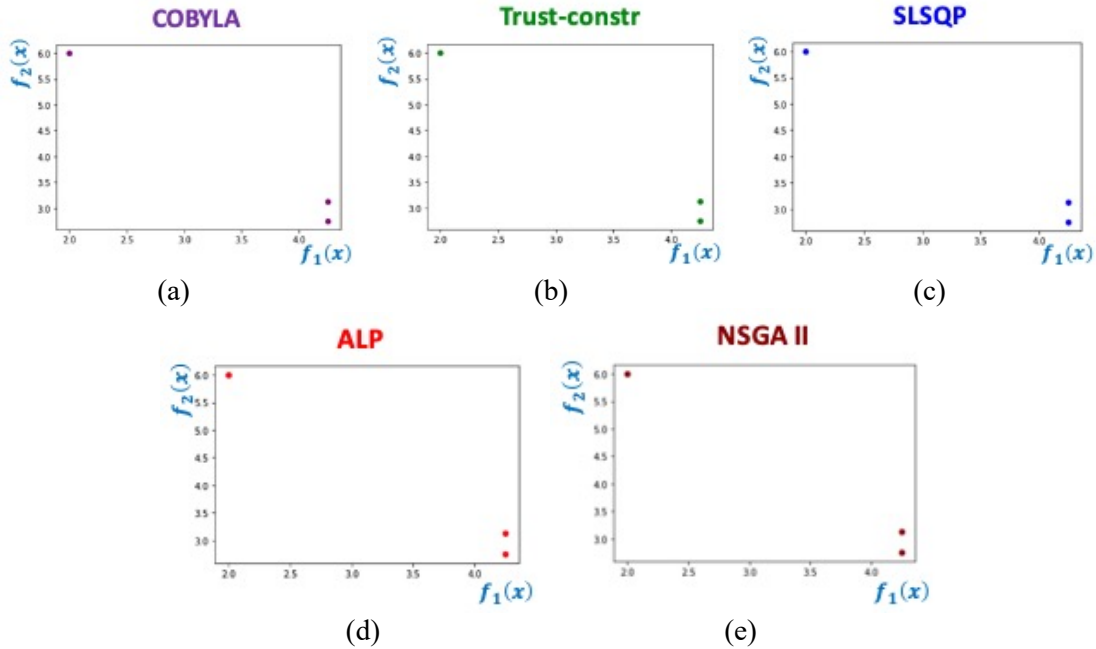
Weight to combine the goals or objectives: as the problem has multiple goals (objectives) but the priority of the goals is unknown. So, for the toy problems, we assume that we use Archimedean strategy (scalarization) to combine the multiple goals, and through exploring different weight scenarios, the tradeoffs among the goals can be better studied. Therefore, further decision support on weight scenario selection can be provided to designers.

**Table 2. 5 Solutions to TP-I (dominated solutions of each scenario)**

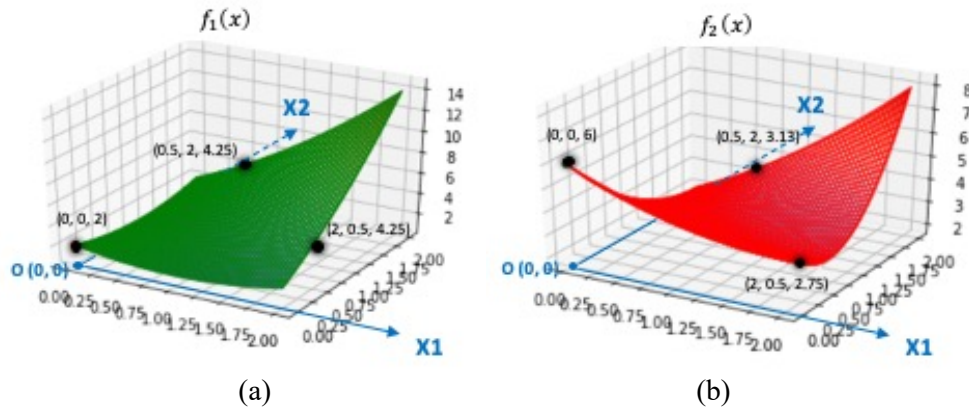
Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II/III	
		Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$
(1, 0)	(0.5, 1)	(2, 0.5)	4.25	(2, 0.5)	4.25	(2, 0.5)	4.25	(2, 0.5)	4.25	(2, 0.5)	4.25
	(0, 0)	(0, 0)	2	(0, 0)	2	(0, 0)	2				
	(2, 0.5)	(2, 0.5)	4.25	(2, 0.5)	4.25	(2, 0.5)	4.25				
(0, 1)	(0.5, 1)	(0.5, 2)	3.13	(0, 0)	6	(0, 0)	6	(0, 0)	6	(0, 0)	6
	(0, 0)	(0, 0)	6			(2, 0)	4				
	(2, 0.5)										
(0.5, 0.5)	(0.5, 1)	(0.5, 2)	3.69	(0.5, 2)	3.69	(0.5, 2)	3.69	(0.5, 2)	3.69	(0, 0)	4
	(0, 0)	(0, 0)	4	(0, 0)	4	(0, 0)	4				
	(2, 0.5)	(2, 0.5)	3.5	(2, 0.5)	3.5	(2, 0.5)	3.5				
(0.7, 0.3)	(0.5, 1)	(0.5, 2)	3.91	(0.5, 2)	3.91	(0.5, 2)	3.91	(0.5, 2)	3.91	(0.5, 2)	3.91
	(0, 0)	(0, 0)	3.2	(0, 0)	3.2	(0, 0)	3.2				
	(2, 0.5)	(2, 0.5)	3.8	(2, 0.5)	3.8	(2, 0.5)	3.8				
(0.3, 0.7)	(0.5, 1)	(2, 0)	3.4	(0, 0)	4.8	(0, 0)	4.8	(0, 0)	4.8	(0, 0)	4.8
	(0, 0)	(0, 0)	4.8								
	(2, 0.5)	(2, 0.5)	3.2			(2, 0.5)	3.2				

**Results:** by using five weight scenarios (1, 0), (0, 1), (0.5, 0.5), (0.7, 0.3), and (0.3, 0.7), no matter which solution algorithms is used, three solutions are always obtained, as visualized on objective space in Figure 2.7 and on X-f(X) Space in Figure 2.8. As the objective of TP-I is to maximize the two objective functions  $f_1(x)$  and  $f_2(x)$ , in Figure 2.7, the ideal solutions should be close to the up-right corner of the objective space, and in Figure 2.8, the value of the objective should be as

high as possible. The two axes  $x_1$  and  $x_2$  are marked with arrows.  $f_1(x)$  and  $f_2(x)$  are the vertical axis of Figure 2.8 (a) and (b) respectively.



**Figure 2. 7 The Solution Points to TP-I on the Objective Space Using Five Algorithms – the Same**



**Figure 2. 8 The Solution Points to TP-I on the  $x$ - $f(x)$  Space. (a) is the 3D illustration of Objective 1,  $f_1(x)$ . (b) is the 3D illustration of Objective 2,  $f_2(x)$ . Since the solutions are the same when using different formulations and algorithms, so all three points are the same for all the five methods in Table 2.5.**

Observation: for a test problem with nonlinear functions and multiple objectives, but without any non-convex function, or the objective functions are with various unit and scale, same solutions can

be obtained using optimizing strategy and *satisficing* strategy. The results of using optimizing solution algorithms such as COBYLA, Trust-constr, and SLSQP are relatively sensitive to the starting point.

*So, in Section 2.2.3, we address “Why using the ALP, the solutions are insensitive to the starting point?”*

### **2.2.3 Method Requirement 1: Combination of Interior-Point Searching and Vertex Searching**

- *Why using the ALP, the solutions are insensitive to the starting point?*
- The combination of interior-point searching and Simplex – In the ALP, there is a module named “XPLORE.” When using XPLORE, “m” starting points within the bounds of each variables are randomly selected and tested the goal-achieved value and the feasibility, then feasible solutions are ranked based on their goal-achieved value. The top “n” solutions are used one-by-one as the starting point, around which, the problem is linearized to a linear problem and then solved by the Dual Simplex. Usually, “m” is set as a number in [800, 1200], and “n” is set as a number in [5, 20]. Therefore, no matter what starting point is assigned by the user, using “XPLORE” allows the feasible points with relatively good goal-achieved values to be selected as the starting points hence to some extent avoid being stuck into the local optima.

### **2.2.4 Toy Problem II (TP-II)**

**– Problem with multiple, nonlinear, and non-convex objectives (goals)**

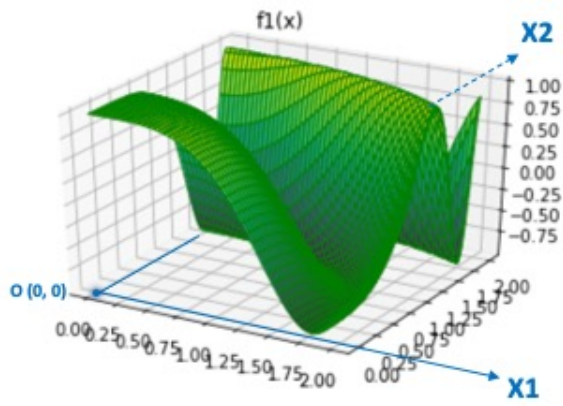
In TP-II, we use a non-convex objective (goal) and a cubic polynomial objective (goal). The formulation, visualization, solutions, and visualization of solutions on objective space and on X-

f(X) Space are given in Table 2.6, Figure 2.9, Table 2.7, Figure 2.10, and Figure 2.11, respectively. For NSGA II/III, when we set the “population<sup>8</sup>” as 20, the solutions are often dominated solutions, and when we set the “population” as 50, the solutions are much better and can converge to nondominated solution under each scenario. We only give the results by using NSGA II with population 50 and generation 100 in Table 2.7, Figure 2.10 and Figure 2.11 because of the high quality of the solutions.

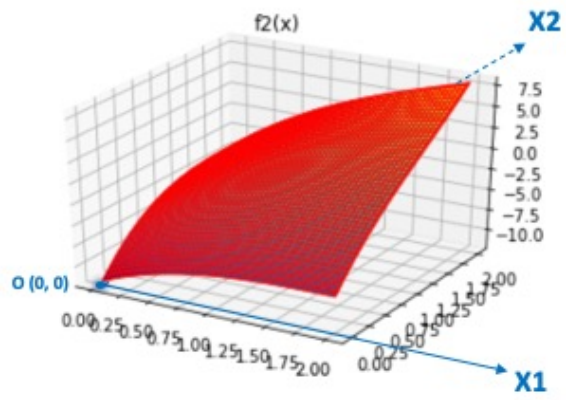
**Table 2. 6 The Optimization Model and Compromise DSP of the TP-II**

Strategy TP	Optimizing	Satisficing
<b>II</b>	<p><u>Objective Functions</u></p> $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = \frac{1}{2} \cdot (x1 - 2)^3 + (x2 - 2)^3 + x1 \cdot x2^2$ <p><u>Constraints and Bounds</u></p> $s. t. \begin{cases} x1 \cdot x2 \leq 1 \\ 0 \leq x1 \leq 2 \\ 0 \leq x2 \leq 2 \end{cases}$ <p><u>Combination of Objective Functions</u></p> $Max \sum_{i=1}^2 w_i \cdot f_i(x)$	<p><u>Given</u></p> $x1, x2, d1^{\pm}, d2^{\pm}$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = \frac{1}{2} \cdot (x1 - 2)^3 + (x2 - 2)^3 + x1 \cdot x2^2$ <p><u>Find</u></p> $x1, x2, d_1^{\mp}, d_2^{\mp}$ <p><u>Satisfy</u></p> <p><u>Goals:</u></p> $\frac{f_1(x)}{1.2} + d1^- = 1$ $\frac{f_2(x)}{8} + d2^- = 1$ <p><u>Constraints:</u></p> $x1 \cdot x2 \leq 1$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><u>Bounds:</u></p> $0 \leq x1, x2 \leq 2$ $0 \leq d1^{\pm}, d2^{\pm} \leq 1$ <p><u>Minimize</u></p> $Z = \sum_{i=1}^2 w_i \cdot di^-$

<sup>8</sup> The “population” in NSGA II is the population size, a parameter predetermined by the user.



(a)



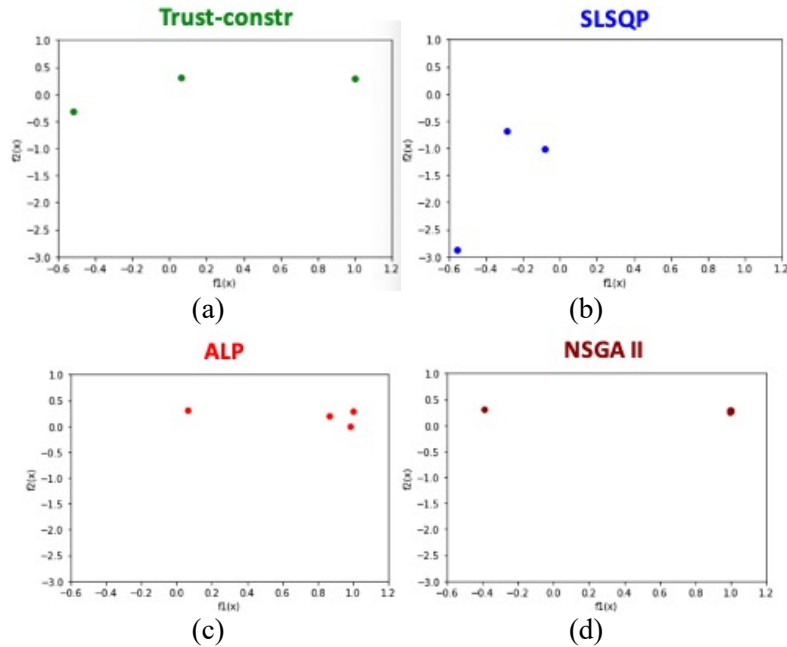
(b)

**Figure 2. 9 The Two Objective Functions on the X-f(X) Plane of TP-II – The first objective  $f_1(x)$  is non-convex**

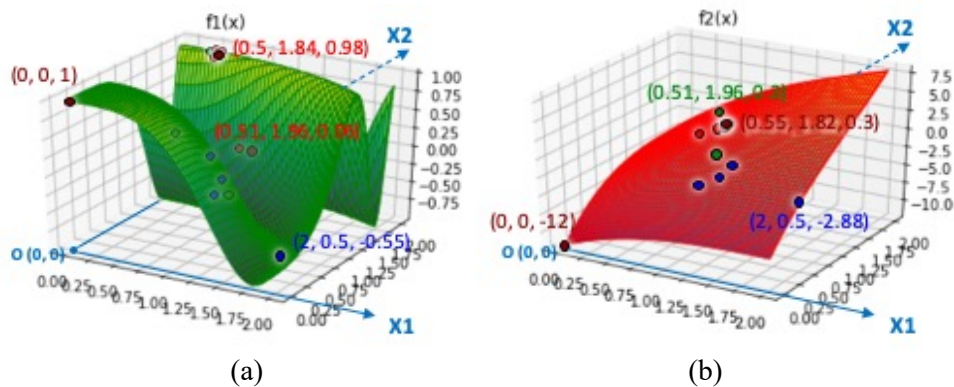
**Table 2. 7 Solutions to TP-II Using Each Solution Algorithm (*dominated solutions, close-to-nondominated solutions or good-enough solutions*)**

Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II/III (Population=50, Generation=100)	
		Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$
(1, 0)	(0.5, 1)	Cannot manage nonconvex equations with bounds	(0.96, 1.06)	0.51	(0.91, 1.01)	-0.28	(0.5, 1.84)	0.97	(0, 0)	1	
	(0, 0)				(0.86, 0.97)	-0.08					
	(2, 0.5)				-	-					(0.95, 1.08)
(0, 1)	(0.5, 1)		-	-	(0.91, 1.01)	-0.69	(0.51, 1.96)	0.3	(0.5, 2)	0.31	
	(0, 0)		(0.51, 1.96)	-0.31	-	-					
	(2, 0.5)		-	-	(2, 0.5)	-2.88					
(0.5, 0.5)	(0.5, 1)		(0.96, 1.06)	0.42	(0.91, 1.01)	-0.49	(0.52, 1.87)	0.54	(0.55, 1.82)	0.64	
	(0, 0)		(0.55, 1.82)	-0.64	-	-					
	(2, 0.5)		-	-	(2, 0.5)	-1.71					
(0.7, 0.3)	(0.5, 1)		(0.96, 1.06)	0.45	(0.91, 1.01)	-0.41	(0.5, 1.84)	0.68	(0.56, 1.8)	0.79	
	(0, 0)		-		-						
	(2, 0.5)		-		-	(2, 0.5)					-1.25
(0.3, 0.7)	(0.5, 1)	(0.96, 1.06)	0.38	(0.91, 1.01)	-0.57	(0.55, 1.82)	0.49	(0.55, 1.81)	0.5		
	(0, 0)	(0.55, 1.82)	-0.5	-	-	(0, 0)	4.8				
	(2, 0.5)	-	-	(2, 0.5)	-2.18						





**Figure 2. 10 The Solution Points to TP-II on the Objective Space Using Four Algorithms – Solutions returned by Trust-constr and SLSQP are not “good enough,” solutions returned by ALP are “good enough” and diverse, and solutions returned by NSGA II contain nondominated solutions but are not diverse**



**Figure 2. 11 The Solution Points to TP-II on the  $x$ - $f(x)$  Space – Using Trust-constraint and SLSQP are easy to fall into local optima. Green, blue, red, and dark red dots are the solutions using Trust-constr, SLSQP, ALP, and NSGA II, respectively.**

Observation: for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, some optimizing algorithms (e.g. COBYLA) cannot manage the non-convexity, whereas some other optimizing algorithms are easy to converge local optima. NSGA II/III is sensitive to parameter setting but can return high-quality solutions (if the population size is large enough)

which are nondominated but not quite diverse and require relatively high computational power. The ALP can return “good enough” and relatively diverse solutions.

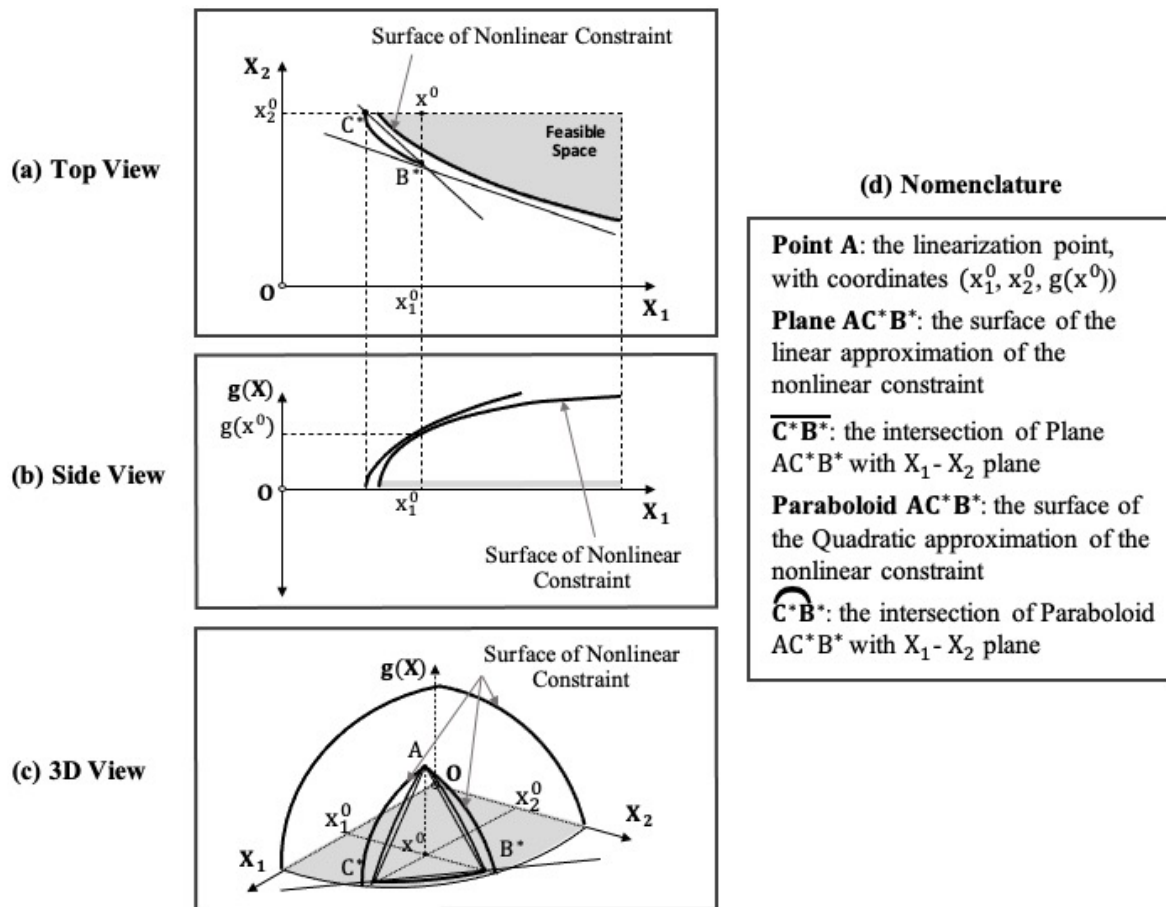
In Section 2.2.5, we address “*Why does the ALP manage non-convex problems and return solutions close to the nondominated solutions (the solutions returned by NSGA II/III)?*”

### ***2.2.5 Method Requirement 2&3: Second-Order Sequential Linearization and Accumulated Linearization***

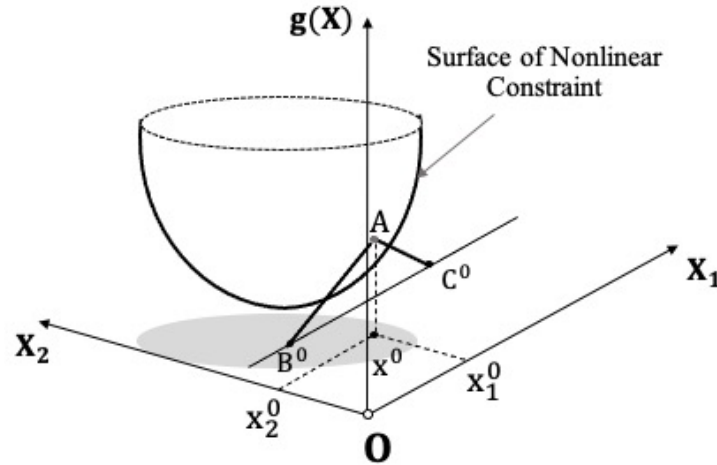
- *Why does the ALP manage non-convex problems and return solutions close to the nondominated solutions (the solutions returned by NSGA II/III)?*
- Two mechanisms of the ALP allow it to linearize the non-convex function relatively accurately and converge with good enough solutions.
- ***First***, using “second-order sequential linearization.” The use of the second-order derivatives function (when the paraboloid being used to approximate the nonlinear function has two real roots) and the first-order derivatives (when the paraboloid has no real root) of the nonlinear functions make the linear problem relatively robust. The nonlinear equation is first approximated into a paraboloid and then approximated into a linear equation. For example, it is as shown in Figure 2.12. First, the surface of the nonlinear constraint is approximated into a paraboloid at point  $x^0$ , represented as Paraboloid  $AB^*C^*$  in Figure 2.12 (c), using second-order derivative (diagonal items of the Hessian matrix of the nonlinear constraint). Then, if the paraboloid has real roots,  $B^*$  and  $C^*$ , which means the paraboloid intersects with X-plane, the paraboloid is linearized as Plane  $AB^*C^*$ . On the contrary, if the paraboloid does not have real root, as

it is shown in Figure 2.13, that is when the paraboloid does not intersect with X-plane, it is linearized as Plane  $AB^0C^0$  using first-order derivatives.

- By using the second-order sequential linearization, designers can have a balance between linearization accuracy and computational complexity. More detailed discussion on the accuracy and flexibility of using second-order sequential linearization is in Section 5.2.1.



**Figure 2. 12 Illustration of the Sequential Linearization using the ALP with Different Views When the Quadratic Approximated Paraboloid Has Real Roots**



**Figure 2. 13 Linearization using the ALP When the**

- Second***, using “accumulated constraints.” The accumulated constraints are only applied when the local convexity of an equation is  $\geq -0.15$ . Such an equation can be a goal or an equality or inequality constraint. The local convexity is the gradient of an equation at a local area around a linearization point. “-0.15” is a number determined by experiments and heuristics. When the local convexity of an equation is greater than or equal to -0.15, it means the equation in the local area around the linearization point is either convex or slightly non-convex. In Figure 2.14 (a) and (b), the nonlinear equations are convex or slightly non-convex, so the equations are linearized around a new linearization points, and the linear constraints in multiple linearization iterations, defined here as “accumulated constraints,” are used to substitute the nonlinear equation. However, if the local convexity of an equation is  $< -0.15$ , as shown in Figure 2.14 (c), which means the equation is significantly non-convex, we only use a single linearized constraint in each iteration.
- Using accumulated constraints, designers can manage nonconvex problems in a way, and deal with highly convex, nonlinear problems relatively more accurately. The details of this mechanism are further discussed in Section 5.2.1.

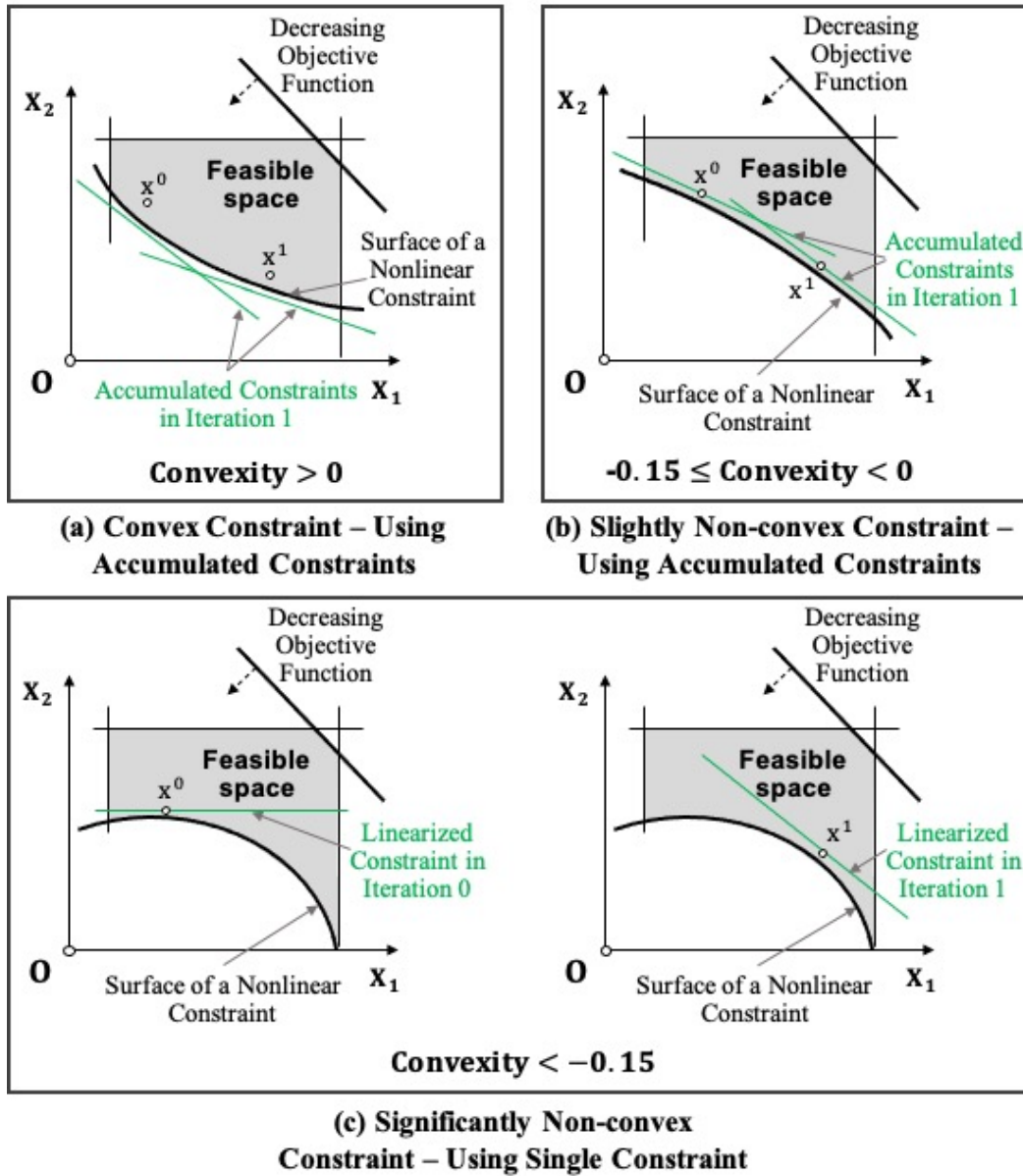


Figure 2. 14 Using the Accumulated Constraints from Multiple Linearization Iterations for Convex or Slightly Non-Convex Equations and Using Single Linearized Constraint for Significantly Non-Convex Constraint

### 2.2.6 Toy Problem III (TP-III)

– Problem with multiple, nonlinear, and non-convex objectives (goals) and the objectives (goals) are with various scales

When adjust the scale of one objective of TP-II, so it becomes TP-III. Accordingly, the target of the goal of the cDSP is enlarged by the same times, as it is shown in Table 2.8. To avoid negative achieved value of objectives, we add lower bound of the two objectives. The formulation, visualization, solutions, and visualization of solutions on objective space and on X-Plane of TP-III are given in Table 2.8, Figure 2.15, Table 2.9, Figure 2.16, and Figure 2.17, respectively. Because the scale of the objectives varies dramatically, combining the objectives using weights makes the problem “lose sense,” which include several situations: 1) the objective with a large scale “dominates” the other objective, 2) the linearized function of the weighted combined objective at a local area is singular, 3) the scale of the second objective is so large that it dominates the penalty added as the constraint, which makes the constraints always be violated.

**Table 2. 8 The Optimization Model and Compromise DSP of the TP-III**

Strategy TP	Optimizing	<i>Satisficing</i>
<b>III</b>	<p><u>Objective Functions</u></p> $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><u>Constraints and Bounds</u></p> $s. t. \begin{cases} x1 \cdot x2 \leq 1 \\ f_1(x) \geq 0 \\ f_2(x) \geq 0 \\ 0 \leq x1 \leq 2 \\ 0 \leq x2 \leq 2 \end{cases}$ <p><u>Combination of Objective Functions</u></p> $Max \sum_{i=1}^2 w_i \cdot f_i(x)$	<p><u>Given</u></p> $x1, x2, d1^{\pm}, d2^{\pm}$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><u>Find</u></p> $x1, x2, d1^{\mp}, d2^{\mp}$ <p><u>Satisfy</u></p> <p><u>Goals:</u></p> $\frac{f_1(x)}{1.2} + d1^- = 1$ $\frac{f_2(x)}{400} + d2^- = 1$ <p><u>Constraints:</u></p> $\begin{aligned} x1 \cdot x2 &\leq 1 \\ f_1(x) &\geq 0 \\ f_2(x) &\geq 0 \\ d_i^- \cdot d_i^+ &= 0, i = 1, 2 \end{aligned}$ <p><u>Bounds:</u></p> $\begin{aligned} 0 &\leq x1, x2 \leq 2 \\ 0 &\leq d1^{\pm}, d2^{\pm} \leq 1 \end{aligned}$ <p><u>Minimize</u></p> $Z = \sum_{i=1}^2 w_i \cdot (d_i^- + d_i^+)$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><u>Bounds:</u></p>

	$0 \leq x_1, x_2 \leq 2$ $0 \leq d_1^\pm, d_2^\pm \leq 1$ <p><u>Minimize</u></p> $Z = \sum_{i=1}^2 w_i \cdot d_i^-$
--	---

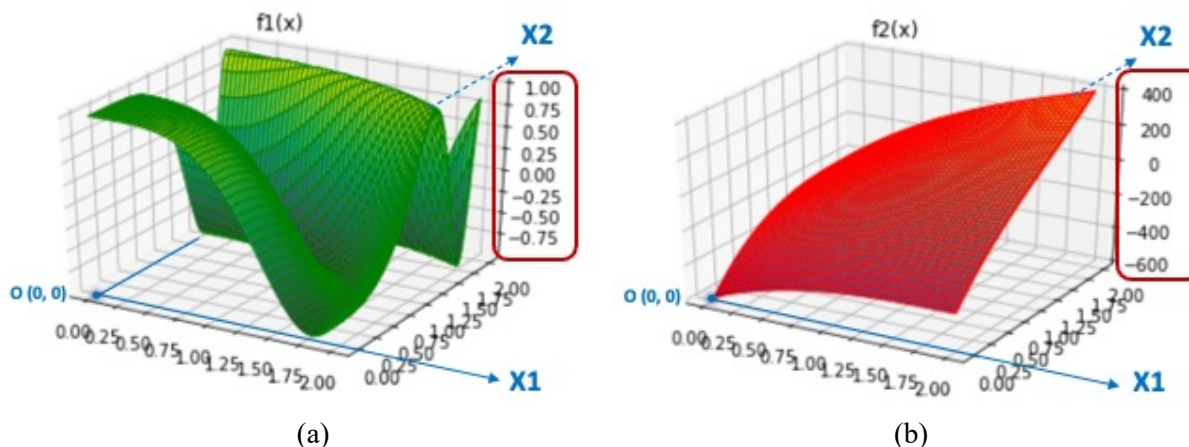


Figure 2. 15 The Two Objective Functions of TP-III on the x-f(x) Space – The second objective  $f_2(x)$  is enlarged by 50 times versus that of TP-II

Table 2. 9 Solutions to TP-III (*dominated solutions, close-to-nondominated solutions or good-enough solutions*)

Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II/III – Population (P)=20/50	
		Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$
(1, 0)	(0.5, 1)	Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints				(0.51, 1.82)	1	P=20: (0.55, 1.85) P=50: (0.515, 1.82)	P=20: 0.99 P=50: 1
	(0, 0)										
	(2, 0.5)										
(0, 1)	(0.5, 1)	Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints				(0.51, 1.96)	15.27	P=20: (0.52, 1.92) P=50: (0.509, 1.96)	P=20: 14.86 P=50: 15.36
	(0, 0)										
	(2, 0.5)										
(0.5, 0.5)	(0.5, 1)	Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints				(0.55, 1.82)	7.5	P=20: (0.52, 1.91) P=50:	P=20: 7.69 P=50:
	(0, 0)										

	(2, 0.5)				(0.53, 1.88)	7.78	
(0.7, 0.3)	(0.5, 1)			$\frac{(0.55, 1.82)}{1.82}$	4.9	P=20: $\frac{(0.55, 1.82)}{1.82}$ P=50: (0.53, 1.87)	P=20: $\frac{4.9}{1.82}$ P=50: 5.01
	(0, 0)						
	(2, 0.5)						
(0.3, 0.7)	(0.5, 1)			$\frac{(0.55, 1.82)}{1.82}$	10.01	P=20: $\frac{(0.54, 1.85)}{1.82}$ P=50: (0.53, 1.89)	P=20: $\frac{10.49}{1.82}$ P=50: 10.6
	(0, 0)						
	(2, 0.5)						

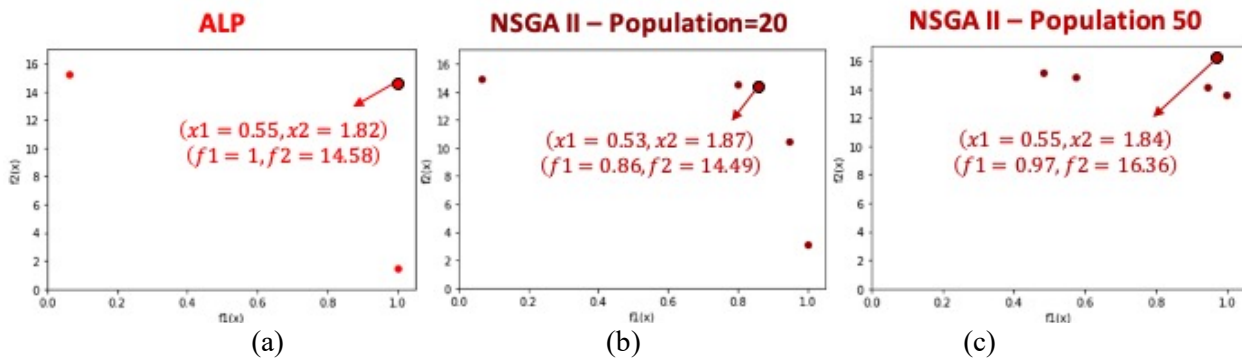


Figure 2. 16 The Solution Points to TP-III on the Objective Space Using Two Algorithms – Solutions returned by NSGA II are closer to the nondominated solution and more diverse but sensitive to parameter setting and require higher computational power.

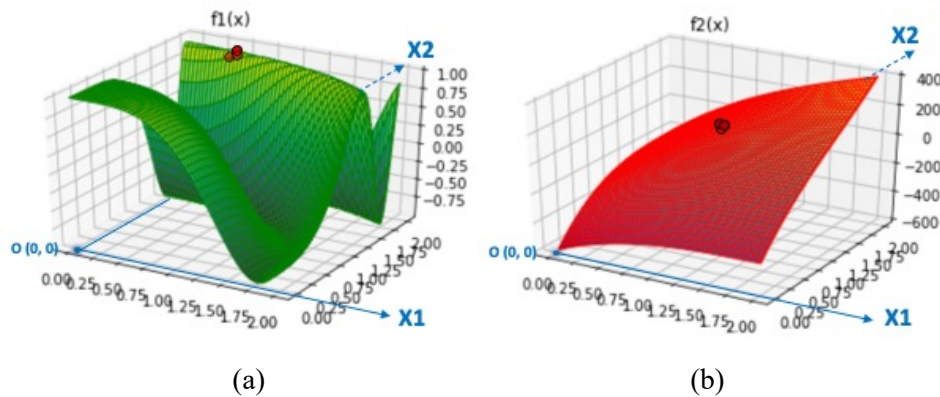


Figure 2. 17 The Solution Points to TP-III on the X-f(X) Plane – Little performance differences between ALP and NSGA II



Observation: for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, and the scale of the objectives varies largely, *why some optimizing algorithms (e.g. COBYLA) cannot work it out? The answer is given as follows.*

### 2.2.7 Method Requirement 4: Using Goals and Minimizing Deviation Variables Instead of Objectives

- Using cDSP, designers minimize the deviation variables that measure the distance between the real achieved value of a goal and the target of the goal.

$$\min Z = \sum_{i=1}^2 w_i \cdot (di^- + di^+) \quad \text{Equation 2. 9}$$

- So, the Lagrange equation does not depend on decision variables X:

$$\frac{\partial L(x, \mu, \lambda)}{\partial x} \equiv 0 \quad \text{Equation 2. 10}$$

- So, the variation of the decision variables does not affect the feasibility of a solution.
- The goal does not dominate one another in completion (or achievement, or fulfillment) due to the difference in the actual achieved value:

$$\frac{f_i(x)}{T_i} \cong \frac{f_j(x)}{T_j}, \text{ given that } T_i \gg T_j \quad \text{Equation 2. 11}$$

Both the first-order and the second-order Lagrange equations are functions of parameters  $\mathcal{P}$  and decision variables  $x$  of the decision model, and Lagrange multipliers  $\mu, \lambda$ .

$$\nabla_x L(x, \mu, \lambda) = \psi(\mathcal{P}, x, \mu, \lambda) \quad \text{Equation 2. 12}$$

$$\nabla_{xx}^2 L(x, \mu, \lambda) = \nabla_x \psi(\mathcal{P}, x, \mu, \lambda) = \psi'(\mathcal{P}', x) \quad \text{Equation 2. 13}$$

Hence, whether a local optimal solution  $x^*$  can be maintained as an optimal solution (given that the model may be incomplete, and something may change but may not be captured into the decision model) depends on the balance of the two equations, and such a balance depends on the value and stability of the variables and coefficients  $\{x, \mathcal{P}, \mu, \lambda\}$ . Suppose mathematically, we identify a local optimal solution  $x^*$  to an engineering-design problem, however, if any uncertainty happens to any parameter or any unparameterized factors that breaks the balance of Equation 2.12 and 2.13, as Equation 2.14 and 2.15, then we lose  $x^*$  as the solution. In fact, this happens quite often to engineering problems because models are always incomplete and inaccurate and with different levels of fidelity. Suppose for a N-dimension, Q-parameter problem, the probability of the uncertainty to a parameter or an equation is  $P$ , and the probability of such an uncertainty breaks the Kuhn-Tucker conditions is  $p$ , then the probability of a local optimal solution  $x^*$  to stay optimal under uncertainty is  $\mathbb{P}(x^*|P)$ , as Equation 2.16 shows, assuming the superposition and mutual influence does not cause any greater uncertainty. However, Equation 2.16 only represents the parameterizable uncertainties, so we use “almost equal to” instead of “equal to.” The actual  $\mathbb{P}(x^*|P)$  considering the mutual influence among parameterizable uncertainties and unparameterizable uncertainties can be even larger.

$$\nabla_x L(\tilde{x}^*, \tilde{\mu}, \tilde{\lambda}) \neq \mathbf{y}(\tilde{\mathcal{P}}, \tilde{x}^*, \tilde{\mu}, \tilde{\lambda}) \quad \text{Equation 2. 14}$$

$$\nabla_{xx}^2 L(\tilde{x}^*, \tilde{\mu}, \tilde{\lambda}) \neq \nabla_x \mathbf{y}(\tilde{x}^*, \tilde{\mu}, \tilde{\lambda}) \quad \text{Equation 2. 15}$$

$$\mathbb{P}(x^*|P) \approx \prod_{q=1}^Q [1 - p(\tilde{\mathcal{P}}_q|P)] \prod_{n=1}^N [1 - P(\tilde{x}_n|P)] \prod_{i=1}^m [1 - P(\tilde{\mu}_i|P)] \prod_{j=1}^{\ell} [1 - P(\tilde{\lambda}_j|P)] \quad \text{Equation 2. 16}$$

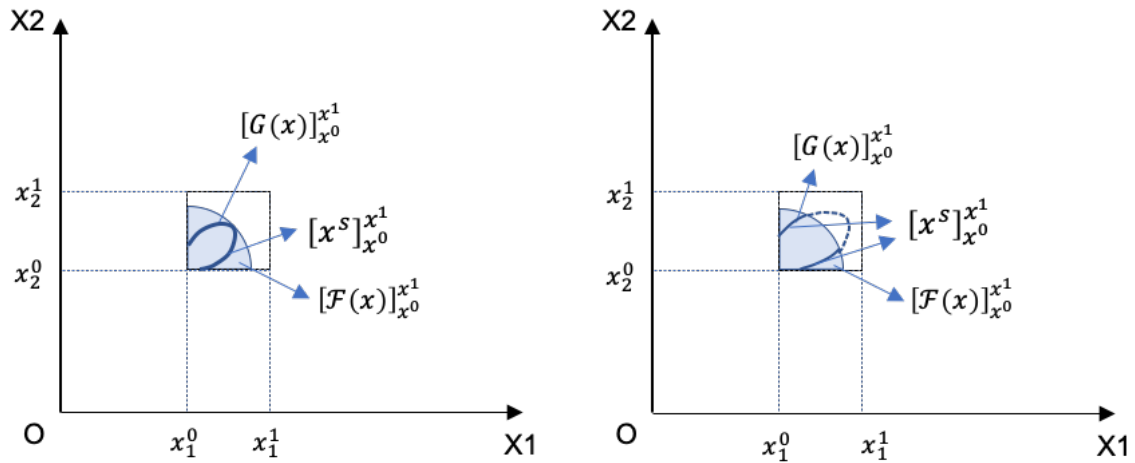
To avoid “no solution” caused by the strong convexity of the objective, or to avoid losing a solution due to uncertainties that breaks the Kuhn-Tucker conditions, we turn to *satisficing* solutions  $\mathbf{x}^s$  instead of optimal solutions  $\mathbf{x}^*$ . We obtain *satisficing* solutions by using goals instead of objectives. A target value  $\mathbf{t}$  as the right-hand side of the objective is assigned thereby the objective  $\mathbf{f}(\mathbf{x})$  becomes a goal  $\mathbf{G}(\mathbf{x})$ , whose position is fixed in the feasible solution space. In  $\mathcal{F}$ ,

$$\mathcal{F} = \{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{g}_i(\mathbf{x}) \geq \mathbf{0}, i = 1, \dots, m, \mathbf{h}_j(\mathbf{x}) = \mathbf{0}, j = 1, \dots, \ell \} \quad \text{Equation 2. 17}$$

$\mathcal{F}$  is the feasible space bounded by all active constraints and bounds, in which, a point, or several points, or an area, that is/are on the goal set  $\mathbf{G}(\mathbf{x})$  or closest to it (using Euclidean distance in this chapter) are the *satisficing* solution(s)  $\mathbf{x}^s$ , see Equation 2.18 and 2.19 and Figure 2.18. Using deviation variables  $\mathbf{d} = (\mathbf{d}^-, \mathbf{d}^+)$  to measure the under-achievement and over-achievement of a goal versus its target and minimizing the deviation variables, the goal  $\mathbf{G}(\mathbf{x})$  becomes an equality constraint to be satisfied, the deviation variables  $\mathbf{d}$  become decision variables that form the new objective function  $\mathbf{z}(\mathbf{d})$ , and the original decision variables  $\mathbf{x}$  become auxiliary variables that do not show up in the objective.

$$\mathbf{G}(\mathbf{x}): \mathbf{f}(\mathbf{x}) + \mathbf{d}^- - \mathbf{d}^+ = \mathbf{t}, \text{ where } \mathbf{0} \leq \mathbf{d}^-, \mathbf{d}^+ \leq \mathbf{1}, \mathbf{d}^- \cdot \mathbf{d}^+ = \mathbf{0} \quad \text{Equation 2. 18}$$

$$\text{Minimize: } \mathbf{z}(\mathbf{d}) \quad \text{Equation 2. 19}$$

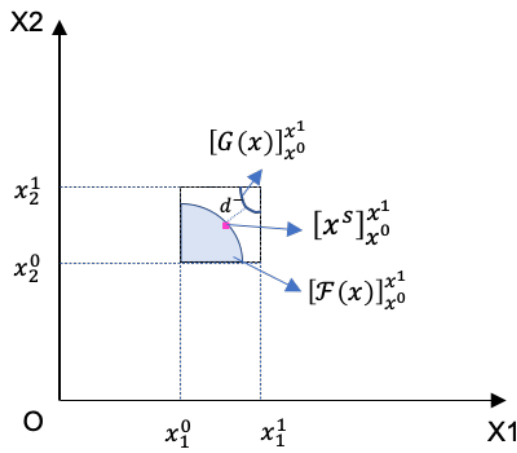


$G(x) \in \mathcal{F}(x)$   
 $d^- = 0, d^+ = 0$

$G(x) \cap \mathcal{F}(x) \neq \emptyset$   
 $d^- = 0, d^+ = 0$

(a)

(b)



$G(x) \cap \mathcal{F}(x) = \emptyset$   
 $d^- > 0, d^+ = 0$

(c)

**Figure 2. 18 Satisficing solutions in different cases**

Such a construct is known as the compromise Decision Support problem (cDSP) (Mistree, Hughes et al. 1993).

**Given:**  $\mathcal{P}, t$

**Find:**  $x, d$

**Satisfy:**

$$x \in \mathcal{F}: \{g(x) \geq 0, h(x) = 0, \text{lower bound} \leq x \leq \text{upper bound}\},$$

$$G(x), 0 \leq d^-, d^+ \leq 1, d^- \cdot d^+ = 0$$

**Minimize:**  $z(d)$

Using the cDSP construct, the nonlinear problem is linearized using the Adaptive Linear Programming (ALP) algorithm (Mistree and Kamal 1985, Mistree, Hughes et al. 1993), so that the linearized problem can be solved using Simplex algorithm and a vertex solution is obtained. The benefits of using a vertex solution versus an interior point solution are 1) a vertex solution is a good enough solution that guarantees the closest distance to the target, and 2) it does not require computing power to search for better interior point solutions.

There are other benefits of using cDSP to manage nonlinear, multi-goal problems, regarding the capacity to identify *satisficing* solutions that are relatively insensitive to uncertainties. Suppose there are parameters  $\boldsymbol{p}$  and deviation variables  $\boldsymbol{d}$  to formulate the merit function  $\boldsymbol{z}(\boldsymbol{p}, \boldsymbol{d})$ , therefore, we identify *satisficing* solution  $\boldsymbol{x}^s$  that obtains the minimal  $\boldsymbol{z}(\boldsymbol{p}, \boldsymbol{d})$ .  $\boldsymbol{x}^s$  is relative insensitive to uncertainties because the probability of any uncertainty breaking the Kuhn-Tucker conditions of a cDSP is much less (than that of an optimization problem without a right-hand side of the objective), as Equation 2.20 – 2.23 show. Usually, the combination form of  $\boldsymbol{d}$  is linear, because for typical engineering problems, we do not expect too complex a merit function unless any domain knowledge on goal(s) indicate the designer to do so, hence the first-order Lagrange equation only consists of  $\boldsymbol{p}$ , and the second-order Lagrange equation constantly equals to zero. So, this formulation significantly increases  $\mathbb{P}(\boldsymbol{x}^s | \boldsymbol{P})$ , the probability of maintaining  $\boldsymbol{x}^s$  as a *satisficing* solution under uncertainties.

$$\begin{aligned} \nabla_{\boldsymbol{d}} L(\boldsymbol{x}^s, \boldsymbol{d}, \boldsymbol{\mu}, \boldsymbol{\lambda}) &= \nabla_{\boldsymbol{d}} \boldsymbol{z}(\boldsymbol{d}) + \sum_{i=1}^m \boldsymbol{\mu}_i \nabla_{\boldsymbol{d}} \boldsymbol{g}_i(\boldsymbol{x}^s) - \sum_{j=1}^{\ell} \boldsymbol{\lambda}_j \nabla_{\boldsymbol{d}} \boldsymbol{h}_j(\boldsymbol{x}^s) - \boldsymbol{\lambda}_{\ell+1} \nabla_{\boldsymbol{d}} G(\boldsymbol{x}^s, \boldsymbol{d}) = \\ \nabla_{\boldsymbol{d}} \boldsymbol{z}(\boldsymbol{d}) + \mathbf{0} - \mathbf{0} \pm \mathbf{1} &= \nabla_{\boldsymbol{d}} \boldsymbol{z}(\boldsymbol{d}) \pm \mathbf{1} = \boldsymbol{y}(\boldsymbol{p}) \pm \mathbf{1} \end{aligned} \quad \text{Equation 2. 20}$$

$$\nabla_{\boldsymbol{d}\boldsymbol{d}}^2 L(\boldsymbol{x}^s, \boldsymbol{d}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \nabla_{\boldsymbol{d}\boldsymbol{d}}^2 \boldsymbol{z}(\boldsymbol{d}) \equiv \mathbf{0} \quad \text{Equation 2. 21}$$

$$\mathbb{P}(\boldsymbol{x}^s | \boldsymbol{P}) \approx \prod_{r=1}^R [1 - \boldsymbol{p}(\widetilde{\boldsymbol{p}}_r | \boldsymbol{P})] \quad \text{Equation 2. 22}$$

$$\mathbb{P}(\boldsymbol{x}^* | \boldsymbol{P}) \ll \mathbb{P}(\boldsymbol{x}^s | \boldsymbol{P}) \quad \text{Equation 2. 23}$$

For a n-dimension, K-goal problem, by adding deviation variables, we increase the dimensionality of a design problem, from  $[x_1, x_2, \dots, x_n]^T$  to  $[x_1, x_2, \dots, x_n, d_1^-, d_1^+, d_2^-, d_2^+, \dots, d_K^-, d_K^+]^T$ , thus make

it possible to absorb the risk of uncertainty that breaks the second-order sufficient conditions. This results in a robust solution, a solution that is relatively insensitive to uncertainties. By returning solutions consisting only of the original decision variables,  $x = [x_1, x_2, \dots, x_n]^T$ , we decrease the dimensionality. Such “dimension expansion and reduction” ensures a solution that is relatively insensitive to the uncertainties embodied in the modeling of an optimization problem. Therefore, we use compromise Decision Support Problems (cDSP) to formulate many-goal problems. For additional information see (Mistree, Patel et al. 1994).

For a many-goal problem, the formulation of the merit function reflects how the goals are managed, especially their priorities. The ways of combining the goals impact the design performance. Hereafter, we focus on exploring the ways of combining the goals in detail in Chapter 6.

### ***2.2.8 Toy Problem IV (TP-IV)***

***– Problem with multiple, nonlinear, and non-convex objectives (goals), the objectives (goals) are with various scales, and the target of the goals with various levels of achievability (the aspiration of one goal is not in feasible space)***

To check out that, if it is the formulation construct, or the solution algorithm, or both, that make the solutions different, we formulate the problem into a cDSP, use the optimizing algorithms and the satisficing algorithm to solve the cDSP, and compare the results. It means that, if the results are the same, then it is the formulation format (objective versus goal) that leads to the difference; otherwise, it is the solution algorithm that leads to the difference.

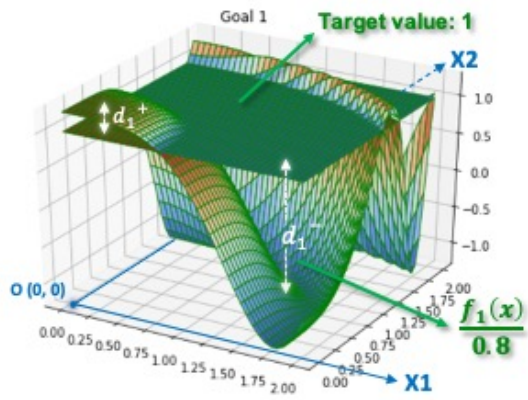
In TP-IV, one more complexity is managed – the targets of the goals are with various levels of achievability, which often takes place in engineering-design problems, so the aspiration of one

goal is not in feasible space. This means the upper limit of some deviation variables (which is 1) must be violated.

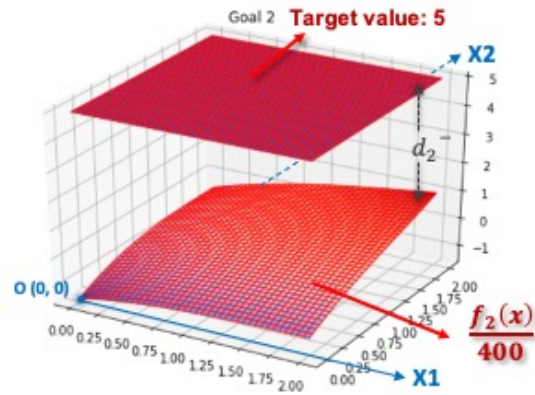
The cDSP formulation, visualization, solutions, and visualization of solutions on objective space and on X-Plane of TP-IV are given in Table 2.10, Figure 2.19, Table 2.11, Figure 2.20, and Figure 2.21, respectively.

**Table 2. 10 The Compromise DSP of the TP-IV**

TP	The Compromise DSP
IV	<p><u>Given</u></p> $x1, x2, d1^{\pm}, d2^{\pm}$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><u>Find</u></p> $x1, x2, d_1^{\mp}, d_2^{\mp}$ <p><u>Satisfy</u></p> <p><u>Goals:</u></p> $\frac{f_1(x)}{1.2} + d1^- = 1$ $\frac{f_2(x)}{400} + d2^- = 5$ <p><u>Constraints:</u></p> $x1 \cdot x2 \leq 1$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><u>Bounds:</u></p> $0 \leq x1, x2 \leq 2$ $0 \leq d1^{\pm}, d2^{\pm} \leq 1$ <p><u>Minimize</u></p> $Z = \sum_{i=1}^2 w_i \cdot (di^- + di^+)$



(a)



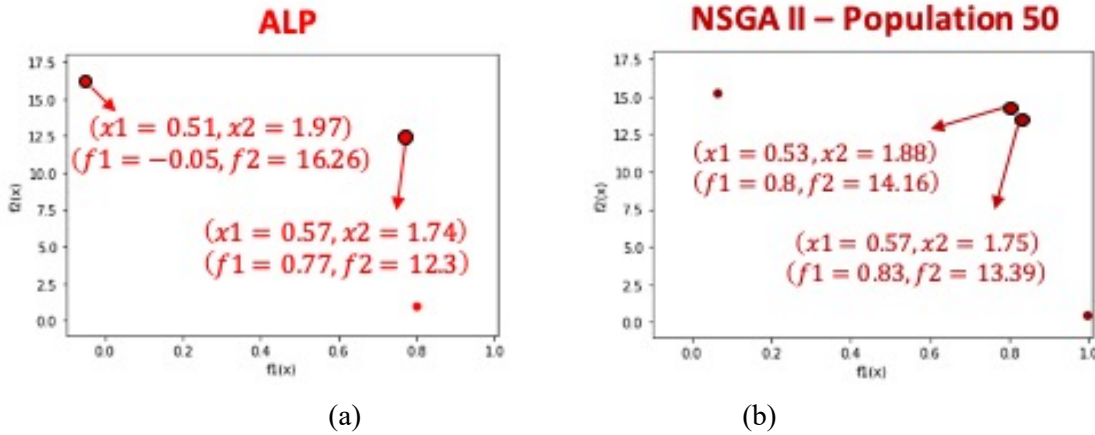
(b)

**Figure 2. 19 The Left-Hand Side (Objective Function) and the Right-Hand Side (Target) of the Two Goals of TP-IV on the X-f(X) Space**

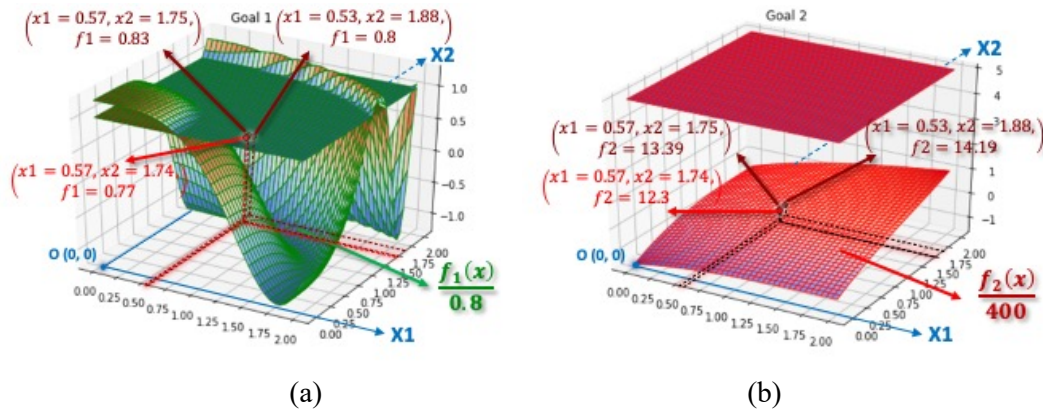


**Table 2. 11 Solutions to TP-IV (*close-to-nondominated solutions or good-enough solutions*)**

Weight	Starting point	COBYLA		Trust-constr		SLSQP		ALP		NSGA II/III – Population (P)=50																					
		Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$	Solution	$\sum_{i=1}^2 w_i \cdot f_i(x)$																				
(1, 0)	(0.5, 1)	Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints	(0.53, 1.75)	0.8	<u>(0.51, 1.88)</u>	<u>0.799</u>	(0, 0)	(2, 0.5)																					
	(0, 0)																														
	(2, 0.5)																														
(0, 1)	(0.5, 1)								Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints	<u>(0.51, 1.97)</u>	<u>15.36</u>	(0.51, 1.96)	14.81	(0, 0)	(2, 0.5)														
	(0, 0)																														
	(2, 0.5)																														
(0.5, 0.5)	(0.5, 1)															Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints	<u>(0.57, 1.74)</u>	<u>7.29</u>	(0.53, 1.88)	7.21	(0, 0)	(2, 0.5)							
	(0, 0)																														
	(2, 0.5)																														
(0.7, 0.3)	(0.5, 1)																						Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints	<u>(0.57, 1.74)</u>	<u>4.69</u>	(0.57, 1.75)	4.6	(0, 0)	(2, 0.5)
	(0, 0)																														
	(2, 0.5)																														
(0.3, 0.7)	(0.5, 1)	Cannot manage nonconvex equations with bounds	All solutions violate one or more constraints	All solutions violate one or more constraints	<u>(0.57, 1.74)</u>	<u>9.88</u>	(0.53, 1.88)	10.54																						(0, 0)	(2, 0.5)
	(0, 0)																														
	(2, 0.5)																														



**Figure 2. 20 The Solution Points to TP-IV on the Objective Space Using Two Algorithms – NSGA II finds more nondominated solutions, whereas ALP finds solutions close to nondominated solutions but with better weighted combined goal-achieved value**



**Figure 2. 21 The Solution Points to TP-IV on the X-f(X) Space – Little performance differences between ALP and NSGA II**

Observation: cDSP and ALP are designed to formulate and explore engineering-design problems with various completabilities of the goals:

$$\frac{f_i(x)}{T_i} \gg \frac{f_j(x)}{T_j} \quad \text{Equation 2. 24}$$

Using the ALP, good enough solutions (comparing with the solutions from NSGA II) can be obtained, whereas using optimizing algorithms, no feasible solutions are returned.

In Section 2.2.9, we address “*Why using ALP allows designers to get good enough solutions to a cDSP with various completabilities of the goals, but using optimizing algorithms (other than NSGA II) does not?*”

**2.2.9 Method Requirement 5: Allowing Some Violations of Soft Requirements, such as the Bounds of Deviation Variables**

- *Why using ALP allows designers to get good enough solutions to a cDSP with various completabilities of the goals, but using optimizing algorithms (other than NSGA II) does not?*
- When  $\frac{f_i(x)}{T_i} \gg \frac{f_j(x)}{T_j}$ , especially when  $d_j^-$  cannot meet its upper limit, that is this bound must be violated:

$$d_j^- \leq 1$$

**Equation 2. 25**

- Because optimization algorithms (such as Trust- constr and SLSQP) treat all constraints and bounds priorities equally, if at least one of the constraints and bounds must be violated, the problem is considered as infeasible. No feasible solutions can be returned using optimizing algorithms.
- Unlike optimizing algorithms, using the ALP, the constraints and the bounds of the system variables are the first priority, and the bounds of the deviation variables are the second priority. If the violation of any deviation bound allows a point  $X^S$  in the feasible area bounded by the constraints and the bounds of system variables to be found, then  $X^S$  is returned as a solution. So, for an n-dimension, m-constraint (with p inequality constraints and m-p equality constraints), k-goal cDSP:

$X^S$  is a *satisficing* solution,

***if and only if***

$$\frac{f_i(X^S)}{T_i} + d_i^- - d_i^+ = 1, \forall i = 1, \dots, k // \text{Goal functions hold at } X^S$$

***and***

$$g_i(X^S) \geq 0, \forall i = 1, \dots, p // \text{Inequality constraints are satisfied at } X^S$$

***and***

$$g_i(X^S) = 0, \forall i = p + 1, \dots, m // \text{Equality constraints are satisfied at } X^S$$

***and***

$$\text{lower bound} \leq X^S \leq \text{upper bound}$$

***and***

$$\min(\text{lower bound} - d_i^{\bar{+}}), \forall i = 1, \dots, k // \text{Minimize the violation of deviation bounds}$$

***and***

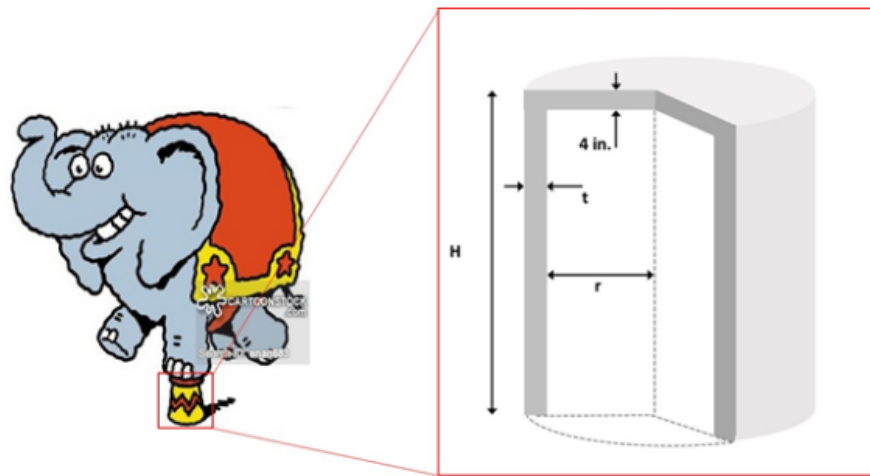
$$\min(d_i^{\bar{-}} - \text{upper bound}), \forall i = 1, \dots, k // \text{Minimize the violation of deviation bounds}$$

### ***2.2.10 Toy Problem V (TP-V)***

***– Problem with three nonlinear objectives (goals), non-convex functions, the objectives (goals) are with various scales, and the target of the goals with various levels of achievability***

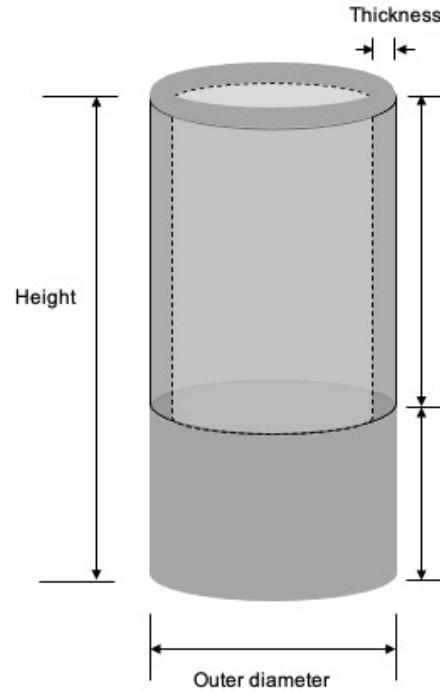
Problem statement of TP-V: in the late 1800s, Ringling Bros and Barnum and Baily Circus was looking to establish dimensions of a new pedestal for their circus elephant Jumbo; see Figure 2.22.

They would play a trick that involved a support pedestal where Jumbo would perform a one-legged handstand. The cost of manufacturing must be minimized, which depends on its thickness, width and the amount of material it would consume. And it must be as tall as possible for a wow factor. And finally, the pedestal must be wide enough to ensure Jumbo has enough room to safely stand on one foot. This means the goals of the design are to minimize the manufacturing cost, maximize the height, and maximize the outer radius. A material of 2024 Aluminum with a modulus of 10600 ksi and a density of  $0.1 \text{ lb}/(\text{in})^3$  has been selected. Jumbo's foot is approximately 25" in diameter so the pedestal must also be greater than 25". Jumbo weighs 13,560lb and stands 13.5ft tall. Use a factor of safety of 1.5.



**Figure 2. 22 The Functionality of an Elephant Stand (TP-V)**

Given a certain type of material, design a cylinder (the “elephant stand”) as it is shown in Figure 2.23. The cylinder has two parts joined together. The upper half is a tube. The designer’s interest is to determine its thickness, radius, and height that best satisfy the goals identified. The lower half is a 4-inch-height solid base. The goals identified by the designer include: Minimizing the manufacturing cost, maximizing the height and maximizing the outer diameter. Requirements include: the upper and lower limit of the parameters that the cylinder can physically reach.



**Figure 2. 23 The Dimension of an Elephant Stand (TP-V)**

The word formulation and mathematical formulation of the cDSP of TP-V are in Table 2.12. There are intermediate variables in TP-V that make the problem a high-dimensional problem hard to be visualized using three-dimensional coordinate system, so we skip the visualization and show the solutions in Table 2.13. Because the optimizing algorithms other than NSGA II cannot deal with TP-V due to its features (three nonlinear goals, non-convex functions, the goals are with various scales, and the target of the goals with various levels of achievability), we only show the results by using ALP and NSGA II. Since both ALP and NSGA II are insensitive to starting point, we do not try different starting points for TP-V. To explore the different priorities of the goals, we use 13 weight scenarios. To simplify the format of the table, we only show the goal achieved value  $f_i(x)$ . For three weight scenarios, (1, 0, 0), (0, 1, 0), and (0.2, 0.6, 0.2), the NSGA II solutions dominate the ALP solutions, that is for every goal, the achieved value is better. For the other ten weight scenarios, NSGA II solutions and ALP solutions do not dominate each other.

**Table 2. 12 The Word-Form and Math-Form of the Compromise DSP of TP-V**

TP	The Word Form	The Math Form
V	<p><b>Given</b>  System parameters  Material – 2014 Aluminum  Elastic modulus for the material  Safety factor  Yield stress for the material  Density of the material  Load (elephant’s weight)  Moment of inertia for the cylindrical section  Maximum normal stress  Maximum buckling stress for a fixed free column  Cost target  Height target  OR target</p> <p><b>Find</b>  System variables  Radius  Thickness  Height</p> <p><u>Deviation variables</u>  <b>Satisfy</b>  System constraints  Reaching the minimum outer diameter  Not exceeding a certain height-to-width ratio  Reaching the stress requirement  Not exceeding the maximum weight  Not exceeding the maximum load in stand.</p> <p><u>System goals</u>  Goal 1: reaching minimum cost target  Goal 2: reaching maximum height target  Goal 3: reaching maximum outer radius target</p> <p><u>Bounds</u>  The upper and lower limit of Radius  Thickness  Height</p> <p><b>Minimize</b>  The deviation function</p>	<p><b>Given</b>  <math>E = 10600000</math>  <math>OR=R+T</math>  <math>SF = 1.5</math>  <math>SIGY = 11000</math>  <math>P = 12000</math>  <math>PI = 2*ACOS(0.0)</math>  <math>RHO = 0.1</math></p> $I = \frac{\pi}{4} * [(R + T)^4 - R^4]$ $W1 = \pi * [(R + T)^2 - R^2] * (H - 4)$ $W2 = \pi * (R + T)^2 * 4$ $W = (W1 + W2) * RHO$ $STR = \frac{P}{\pi * [(R + T)^2 - R^2]}$ $PCR = \frac{\pi^2 * E * I}{4 * H^2}$ $COST = e^{\frac{2.5}{T}} * e^{\frac{3}{R}} * W$ <p>Cost target = 5000  Height target = 180  OR target = 15</p> <p><b>Find</b>  <u>System variables</u>  R: Radius  T: Thickness  H: Height</p> <p><u>Deviation variables</u>  <math>d_i^+</math>: over achievement of Goal i, where <math>i=1,2,3</math>  <math>d_i^-</math>: under-achievement of Goal i, where <math>i=1,2,3</math></p> <p><b>Satisfy</b>  <u>System constraints</u>  minOD: minimum outer diameter: <math>2 * R + 2 * T \geq 6</math> (CO1)  heiwid: height to width ratio: <math>R + T - 0.03 * H \geq 0</math> (CO2)  stress: minimum stress in stand: <math>\frac{SIGY}{SF} - STR \geq 0</math> (CO3)  weight: maximum weight: <math>1000 - W \geq 0</math> (CO4)  buckle: maximum load in stand: <math>\frac{PCR}{F} - P \geq 0</math> (CO5)</p> <p><u>System goals</u>  Goal 1: minimum cost: <math>\frac{cost}{cost\ target} + d_1^- - d_1^+ = 1</math> (G1)  Goal 2: maximum height: <math>\frac{height}{height\ target} + d_2^- - d_2^+ = 1</math> (G2)  Goal 3: maximum outer radius: <math>\frac{OR}{OR\ target} + d_3^- - d_3^+ = 0</math> (G3)</p>

		<u>Bounds</u> $3 \leq R \leq 45$ $0.5 \leq T \leq 2.5$ $100 \leq H \leq 120$ <u>Minimize</u> The deviation function $Z = \sum_{i=1}^3 w_i \cdot (di^- + di^+), \sum_{i=1}^3 w_i = 1$
--	--	--

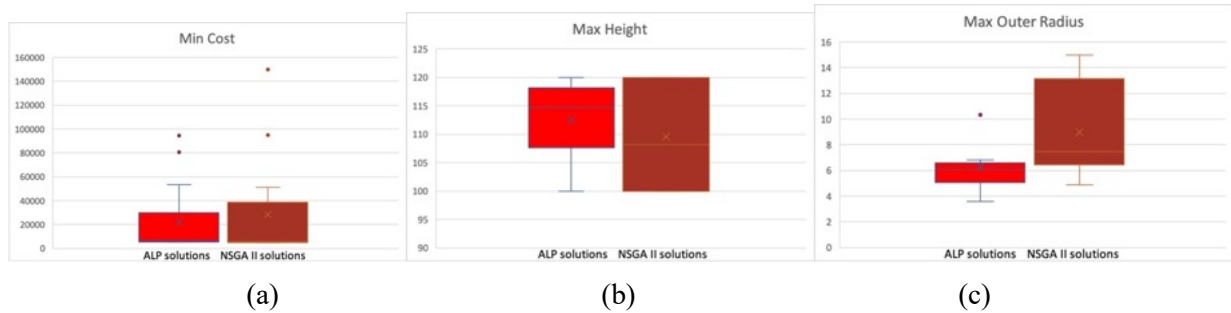
**Table 2. 13 Solutions to TP-V – the nondominated solution of each scenario is highlighted, the solution that gives the better achieved value of a goal but is not a nondominated solution is underlined**

Weight Scenario	Weight			ALP			NSGA II/III – Population (P)=50		
	$w_1$	$w_2$	$w_3$	$\min f_1(x)$	$\max f_2(x)$	$\max f_3(x)$	$\min f_1(x)$	$\max f_2(x)$	$\max f_3(x)$
1	1	0	0	5507.8	100	4.38	<b>5000</b>	<b>102.56</b>	<b>4.98</b>
2	0	1	0	53530.4	118.8	3.59	<b>25768</b>	<b>120</b>	<b>4.89</b>
3	0	0	1	<u>80580.1</u>	100	10.44	94863	<u>114.156</u>	<u>15</u>
4	0.6	0.2	0.2	5340.06	<u>110.6</u>	5.09	<u>5000</u>	100	<u>7.44</u>
5	0.2	0.6	0.2	6079.2	117.5	6.08	<b>5892.7</b>	<b>120</b>	<b>7.05</b>
6	0.2	0.2	0.6	<u>6547.6</u>	114.7	6.82	51197	<u>120</u>	<u>15</u>
7	0.5	0.35	0.15	6079.2	<u>117.5</u>	6.08	<u>5000</u>	100	<u>7.44</u>
8	0.15	0.5	0.35	<u>6114.2</u>	<u>119.96</u>	6.12	7848.7	108.15	<u>11.29</u>
9	0.35	0.15	0.5	5564.9	<u>106.1</u>	6.33	<u>5208.9</u>	100	<u>8.02</u>
10	0.7	0	0.3	5269.2	<u>109.2</u>	5.13	<u>5000</u>	100	<u>7.43</u>
11	0.3	0.7	0	5821.1	<u>120</u>	5.06	5691.7	<u>120</u>	<u>5.81</u>
12	0	0.3	0.7	<u>94608.4</u>	110	10.33	149720	<u>120</u>	<u>15</u>
13	0.33	0.33	0.33	6079.2	<u>117.5</u>	6.08	<u>5060.5</u>	100	<u>7.66</u>

If we visualize the achieved values of each goal under thirteen weight scenarios separately using box charts (Figure 2.24), the distribution of the goal-achieved values from using ALP and NSGA II vary. The achieved value of Goal 1 (Minimizing Cost) and Goal 2 (Maximizing Height) by using ALP is slightly better more stable than that of using NSGA II; see Figure 2.24 (a) and (b). However, the achieved value of Goal 3 (Maximizing Outer Radius) by using NSGA II is much

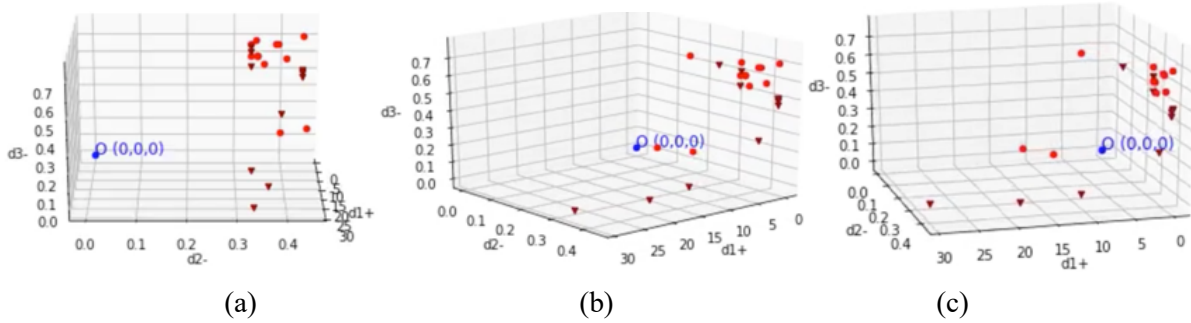


better than that of using ALP; see Figure 2.24 (c). Therefore, even using the same weights to combine the goals, solutions and goal-achieved value are different.



**Figure 2. 24 The Box Chart Solution Points to TP-V on the Objective Space Using Two Algorithms**

We visualize the solution points on the “deviation space” in Figure 2.25 from different angles. The three axes are  $d_1^+$ ,  $d_2^-$ , and  $d_3^-$ . The ideal solution point (or the Utopian point) which is not in the feasible space is the origin O (0, 0, 0) of this 3D coordinate system because at this point all three goals reach the target. So, the solution points close to O are more desired because they get better goal-achieved values. The round dots are solutions using the ALP. The triangle dots are solutions using NSGA II. From Figure 2.25, we cannot conclude that one algorithm performs obviously better than the other, but for Goal 3, NSGA II solutions are much closer to O. However, the solutions with low  $d_3^-$  have much higher  $d_1^+$ , which means to obtain a better outer diameter of the elephant stand, an extremely high cost is required. So, we can conclude that using NSGA II allows designers to identify solutions to better achieve Goal 3 by sacrificing Goal 1.



**Figure 2. 25 The Solution Points to TP-V on the Deviation Space Using Two Algorithms – Round dots are solutions using the ALP and Triangle dots are solutions using NSGA II**

Observation: for an engineering-design problem with three nonlinear goals, non-convex functions, the goals are with various scales, and the target of the goals with various levels of achievability, using ALP allows designers to get good enough solutions that are close to the nondominated solutions obtained by using NSGA II. However, as we give the drawbacks of using NSGA II earlier in this section, designers cannot get insight or knowledge on the problem formulation and improvement, we conclude that the formulation of cDSP and the use of ALP can give *satisficing* solutions and knowledge on problem formulation and system behavior within a relatively low computational complexity.

## 2.3 Summary of Differences between Optimizing and *Satisficing* Strategy

### 2.3.1 Differences between *Optimizing* and *Satisficing* Strategy

Why do cDSP, ALP, and DSIDES work for nonlinear, non-convex, multi-objective, multi-unit, and evolving-target Problems?

In summary, there are four advantages using *satisficing* strategy, which in this dissertation, is realized by using cDSP, ALP and DSIDES. The advantages include (Table 2.1):

***The advantage in formulation:***

- Using Goals and Minimizing Deviation Variables Instead of Objectives.
  - The benefits are: At a solution point, only the necessary Kuhn-Tucker conditions are met, whereas the sufficient Kuhn-Tucker conditions do not have to be met.
  - Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and unparameterizable uncertainties.

***The advantages in approximation:***

- Using second-order sequential linearization
  - The benefit is: Designers can have a balance between linearization accuracy and computational complexity.
- Using accumulated linearization
  - The benefit is: Designers can manage nonconvex problems in a way, and deal with highly convex, nonlinear problems relatively more accurately.

***The advantage in exploration:***

- Combining interior-point searching and vertex searching:
  - The benefit is: Designers can avoid being stuck into local optimum to some extent and identify satisficing solutions relatively insensitive to starting points changing.

***The advantage in evaluation:***

- Allowing some violations of soft requirements, such as the bounds of deviation variables.
  - The benefits are: Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility.
  - As a result, goals and constraints with different scale can be managed.

### ***2.3.2 Summary of Differences among cDSP, Goal Programming, and Mathematical Programming***

#### ***- Why We Choose cDSP?***

Based on the previous discussion – the features each method can or cannot manage in Section 1.22, Table 1.3, the mathematical explanation regarding meeting the Kuhn-Tucker conditions in Section 2.1 and using different methods to solve five toy problems and get different solutions in Section 2.2, we summarize the differences among cDSP, Goal Programming, and Mathematical Programming seeking optimal solutions as follows.

**Stage 1: Formulation.** *First*, the cDSP is a hybrid between mathematical programming (seeking optimal solutions) and goal programming. In a cDSP there are both constraints and goals. In goal programming, there are no constraints<sup>9</sup>. In a cDSP, the constraints are requirements (demands) that cannot be violated, whereas the goals are soft requirements (wishes) whose targets may not be reached but we want to minimize the distance between the targets and our results. The constraints and goals can be linear or non-linear (convex or non-convex) or both, and equality or inequalities or both. The benefit of being able to model both demands and wishes in one formulation is attractive in design. Due to the complexity of the supply chain with mass customization, the resulting cDSP usually entails dealing with non-convex and convex constraints and goals. The algorithm for solving the cDSP is documented in (Mistree, Hughes et al. 1993).

---

<sup>9</sup> Although in later publications, the formulation of Goal Programming allows managing constraints. By the time Mistree and the coauthors published their work on cDSP and ALP (Mistree, Hughes et al. 1993), it was generally accepted that in Goal Programming, there are only “soft requirements” as goals but no “rigid requirements” as constraints.

Second, in a cDSP, we use deviation variables to assess the extent by which we over-achieve or under-achieve a goal. By adding the deviation variables, we ensure that the solutions to a cDSP satisfies the necessary Kuhn-Tucker condition. The solutions do meet the test of sufficiency to guarantee a “true” or “global” optimum. A *satisficing* solution to a cDSP is the mapping of an optimal solution to a lower-dimensional space. The dimensions being reduced consist of the deviation variables  $D = [d_1^-, d_1^+, d_2^-, d_2^+, \dots, d_k^-, d_k^+]^T$ . By adding deviation variables, we increase the dimensionality of a design problem, from  $[x_1, x_2, \dots, x_n]^T$  to  $[x_1, x_2, \dots, x_n, d_1^-, d_1^+, d_2^-, d_2^+, \dots, d_k^-, d_k^+]^T$ , thus making it possible to absorb the risk of uncertainty at the constraint boundary. This results in a robust solution, that is, one that is relatively insensitive to uncertainties. By returning solutions consisting only of system variables, as is the case in solving an optimization problem,  $X = [x_1, x_2, \dots, x_n]^T$ , we decrease the dimensionality. Such “dimensionality reduction” does not result in a solution that is relatively insensitive to the uncertainties embodied in the modeling of the constraints of an optimization problem. For additional information see (Mistree, Patel et al. 1994).

Third, although a cDSP has similarities with the auxiliary problem of a linear programming problem when using the two-phase method<sup>10</sup>, there are differences. For a linear programming problem (P), when we relax the  $m$  equality constraints  $A \cdot X = b$  to  $m$  inequality constraints  $A \cdot X + U = b$ , by adding slack variables (or artificial variables)  $U = [u_1, u_2, \dots, u_m]^T$ , and change the objective function from  $\min C^T \cdot X$  to  $\min \sum_{i=1}^m u_i$ , an auxiliary problem (A) of the original linear

---

<sup>10</sup> The introduction of the auxiliary problem is given at “<http://www.math.uwaterloo.ca/~hwolkowi/henry/teaching/f05/350.f05/L18.pdf>”

programming problem (P) is created. If a solution  $[x_1, x_2, \dots, x_n, u_1, u_2, \dots, u_m]^T$  is optimal for (A) with  $u_i = 0, i = 1, 2, \dots, m$ , then the solution  $[x_1, x_2, \dots, x_n]^T$  is feasible for (P).

There are similarities between a cDSP and an auxiliary problem (A). The slack variables  $U$  in the auxiliary problem are similar to the deviation variables  $D$  in a cDSP, if we treat the equality constraints  $A \cdot X + U = b$  of (A) as the goals of a cDSP  $\frac{Goal_i(X)}{Target_i} + d_i^- - d_i^+ = 1$ . The objective of (A) is minimizing the sum of the slack variables  $U$ , similarly, the merit function of a cDSP is minimizing the linear combination of the deviation variables  $D$ .

However, there are differences between a cDSP and an auxiliary problem (A) of a linear programming problem (P). An auxiliary problem is a linear problem, whereas a cDSP can be nonlinear – both constraints and goals. When solving an auxiliary problem (A), one can only obtain the feasibility of the its original problem (P) but a *satisficing* solution (a good enough solution) is not guaranteed because the original objective function  $\min C^T \cdot X$  is not incorporated in (A). On the contrary, in a cDSP, goals are satisfied as equality constraints in a corresponding optimization problem, thus, a *satisficing* solution that is close to achieve the goals is identified. In addition, in an auxiliary problem, for any constraint, we only minimize either its under-achievement or over-achievement, whereas in a cDSP, we minimize both under-achievement and over-achievement of each goal. Furthermore, in an auxiliary problem, we treat all constraints equally by simply adding the slack variables  $U$ , whilst in a cDSP, we use weights to linearly combine the deviation variables so that we may assign different priority to each goal.

In summary, a cDSP is different from goal programming, optimization, or an auxiliary problem in linear programming. Further a *satisficing* solution to a cDSP is not only feasible, but also good enough with respect to the achievement of the goals.

**Stage 2: Exploration of the solution space.** In the second stage, the solution space is explored to find *satisficing* solutions associated with each design preference (scenario), in different phases in the product life cycle. *Type I* and *Type II* uncertainty are managed in the exploration of the solution space (ESS). In this dissertation, when we refer to design preferences, we particularly focus on the importance of the different goals.

**Weight sensitivity analysis – exploration of the design preferences.** We use weight sensitivity analysis to explore how the assessing different weights to the goals affect the system performance, that is, identifying *satisficing* solutions that are relatively insensitive to uncertainties.

**System capacity analysis – identification and management of the sensitive segment and bottleneck.** To overcome the capacity limitation of constraints or bounds, we propose system capacity analysis to identify the sensitive segment and bottleneck. If an inequality constraint has zero or tiny surplus or slack compared with its right-hand-side value, we define it as an active constraint. The solution is on or close to the boundary of the active constraint, so the solution is sensitive to the uncertainty of the active constraint. If the shadow price of an active constraint is lower than other active constraints, by relaxing this active constraint, we may not get much improvement in achieving the goals, and we define such a constraint as a “sensitive segment.” We then move the solution away from the sensitive segment by restricting the active constraint, that is, by adding a buffer to the constraint to prevent the solution from reaching the boundary defined by the constraint. If the shadow price of an active constraint has the largest value in comparison with that of other constraints, relaxing the constraint can result in the greatest improvement of the achievement of the goals. We define such an active constraint as a “bottleneck.” Also, it is important to find ways of relaxing the bottleneck in the physical system to boost the system potential. Once there is no longer the potential of physically relaxing the constraint, we move the

solution away from the newly relaxed boundary by restricting the constraint by adding a buffer to the defined boundary. Thus, we balance the need for robustness of the solution with our desire to obtain the best satisficing solution.

## **2.4 Research Questions (RQ1-RQ4)**

Based on the design construct we choose as the model evolution construct, cDSP, and the different types of uncertainty we need to manage through model evolution, we pose the primary research question in this section. First, what are the different types of uncertainty? How can we justify the primary research question with respect to managing the four types of uncertainty or realizing the four types of robust design?

### ***2.4.1 Justification of the Primary Research Question regarding Requirements***

#### ***- Four Types of Robust Design***

There are four types of robust design (Allen, Seepersad et al. 2006) with respect to managing the four types of uncertainty in engineering design. The first three types of robust design area are conceptually illustrated in Figure 2.26. Type IV uncertainty is the combination of Type I, II, and III uncertainty, so it is not represented as an individual area.

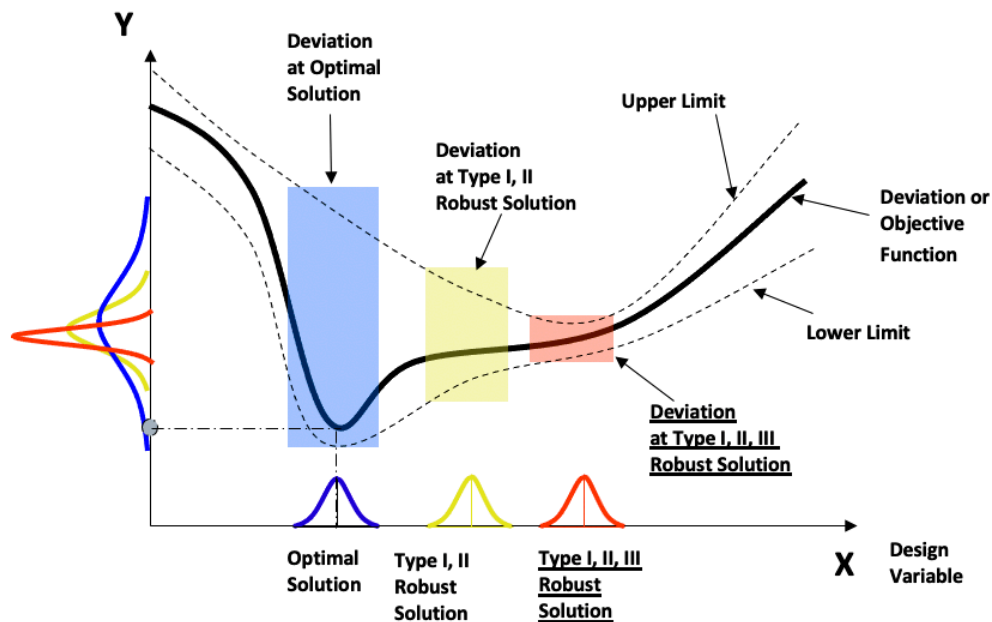
In **Type I** robust design, design variable values are identified to satisfy a set of performance requirement targets regardless of noise factors. *Noise factors* are not under a designer's control.

In **Type II** robust design, design variable values are determined that satisfy a set of performance requirement targets regardless of anticipated variations in those *design variables*.



In **Type III** robust design, design variable values are determined which satisfy a set of performance requirements regardless of variations in the *mathematical models* used to describe that performance.

In **Type IV (Combination of Types I, II, and III)** robust design, design variable values are determined which satisfy a set of performance requirements in spite of variability introduced by a *hierarchical, multiscale or multidisciplinary formulation of the product*.



**Figure 2. 26 Four Types of Robust Solution**

Based on the requirement of managing four types of uncertainty in engineering design, we justify the primary research question into three sub-research questions in Table 2.14.

- **“Primary Research Question – How can designers realize model evolution using satisficing strategy so that we can manage chaos in the physical world, reduce the risk of losing an optimal solution, and discover domain-independent knowledge to update metaheuristics?”**

The research questions regarding the realization of Type I and Type II Robust Design is marked as RDI-II. The research questions regarding the realization of Type III and Type IV Robust Design are marked as RDIII and RDIV, respectively. Hypotheses are proposed in Chapter 3.

**Table 2. 14 Justified Research Questions regarding Four Types of Robust Design**

Chapter	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8	Ch9
<b>Actions</b>	RG H	<i>RDI-II: What are the mechanisms and procedures that enable the exploration of the solutions relatively insensitive to Type I and Type II uncertainty?</i>	T V e M	E s V e S Q T A Q	E V e S Q T A Q			C Q E V a	T E
		<i>RDIII: What is the mathematics in the design method that allow the exploration of the solutions relatively insensitive to Type III uncertainty?</i>	T V e M		E V e S Q T A Q	E V e S Q T A Q			
		<i>RDIV: What is the method that allows sensing and managing Type IV uncertainty?</i>	T V e M			E V e S Q T A Q	E V e S Q T A Q		
<b>Nomenclature</b>	RG – give research gaps H – give hypotheses RD – robust design TVe – theoretically verify hypotheses M – introduce methods EVe – empirically verify hypotheses SQT – specify research questions in the context of test problems AQ – answer research questions CQ – closure the answers to research questions EVa – empirically validate hypotheses TE – theoretically extend the research								

#### 2.4.2 Justified Research Questions regarding Tasks

##### - In the Context of Four Stages of the Design Loop

Establishing connections among the four stages of the model evolution cycle – formulation, approximation, exploration, and exploration, is a way to allow information passing through different stages so as to realize robust design.

In this dissertation, we identify four connections among the stages; see Figure 2.27: connections between formulation and exploration, connections among approximation, exploration, and

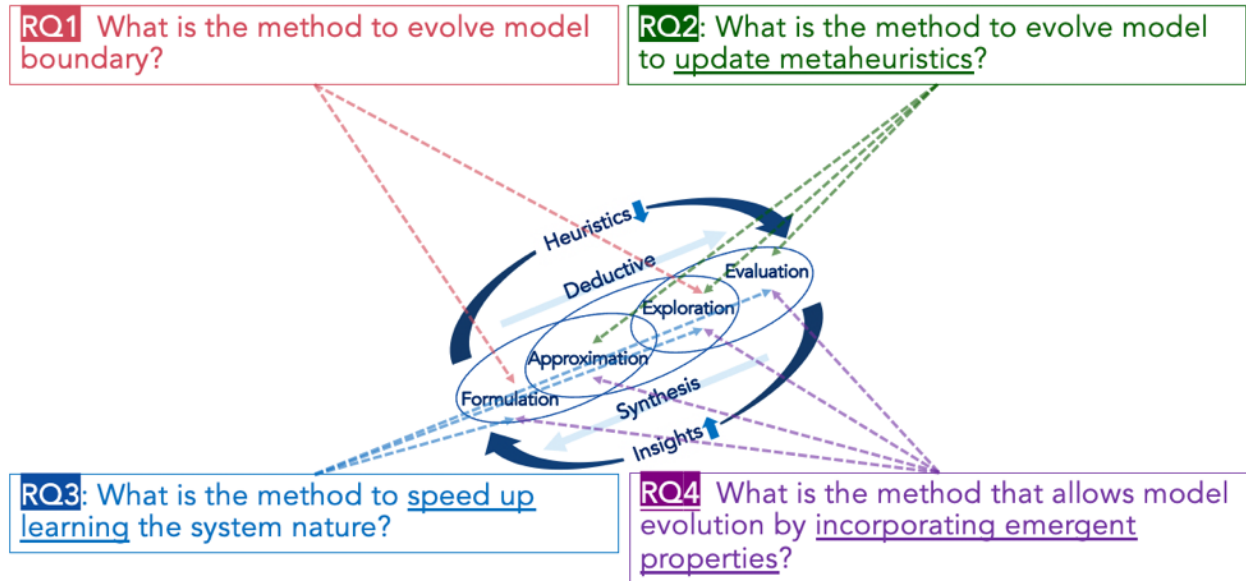
evaluation, connections among formulation, exploration, and evaluation, and connections among all four stages. We explain what are expected to happen or the potential contribution of the connections, hereafter.

*Formulation-exploration:* through exploring the solution space, designers can identify the segments in the model that can be improved to make solutions more insensitive to uncertainties. Based on such findings, designers can improve the model formulation

*Approximation-exploration-evaluation:* through exploration and evaluation, the solution quality is learned and defined, and its association with the metaheuristics used in the approximation can be identified and quantified, based on which, designers can update the metaheuristics in approximation and realize the design relatively insensitive to the uncertainties in the approximation.

*Formulation-exploration-evaluation:* through exploration and evaluation, the solution quality associated with different assumptions and rules in formulation are learned, based on which, designers can update the formulation rules and make the design relatively insensitive to the assumptions, errors, uncertainties, and inaccuracies in the formulation. As the process goes on, designers can speed up the learning process by selecting the most representative scenarios in the formulation.

*Formulation-approximation-exploration-evaluation:* by connecting all four stages, it is possible for designers to identify emergent properties of the system and incorporate them into the model by adjusting any stage.



**Figure 2. 27 Justified Research Questions RQ1-RQ4 and Their Connections with the Design Loop**

Model evolution includes but are not limited to establishing the aforementioned connections, based on which, we justify the primary research question into four research questions regarding establishing connections among stages in engineering design in Table 2.15.

The four research questions regarding the establishment of the four multi-stage connections are marked as RQ1, RQ2, RQ3, and RQ4, respectively. Hypotheses are proposed in Chapter 3.

**Table 2. 15 Connection between Research Questions (RQs) and Chapters (Ch)**

Chapter	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8	Ch9
Actions	RG H	<i>RQ1: What is the method to evolve model boundary?</i>	TVe M	EVe SQT AQ				CQ EVa	TE
		<i>RQ2: What is the method to evolve model to update metaheuristics?</i>	TVe M		EVe SQT AQ				
		<i>RQ3: What is the method to speed up learning the system nature?</i>	TVe M			EVe SQT AQ			
		<i>RQ4: What is the method that allows passing the information through multiple scales of a system?</i>	TVe M				EVe SQT AQ		

<b>Nomenclature</b>	RG – give research gaps H – give hypotheses RQ – research question TVe – theoretically verify hypotheses M – introduce methods EVe – empirically verify hypotheses SQT – specify research questions in the context of test problems AQ – answer research questions CQ – closure the answers to research questions EVa – empirically validate hypotheses TE – theoretically extend the research
---------------------	--

## 2.5 Specification of Hypotheses (SH1-SH4)

In Chapter 1, to fill the research gaps, it is hypothesized that *by connecting the multiple stages of design and passing information through them, designers can improve their decision models in iterations.*

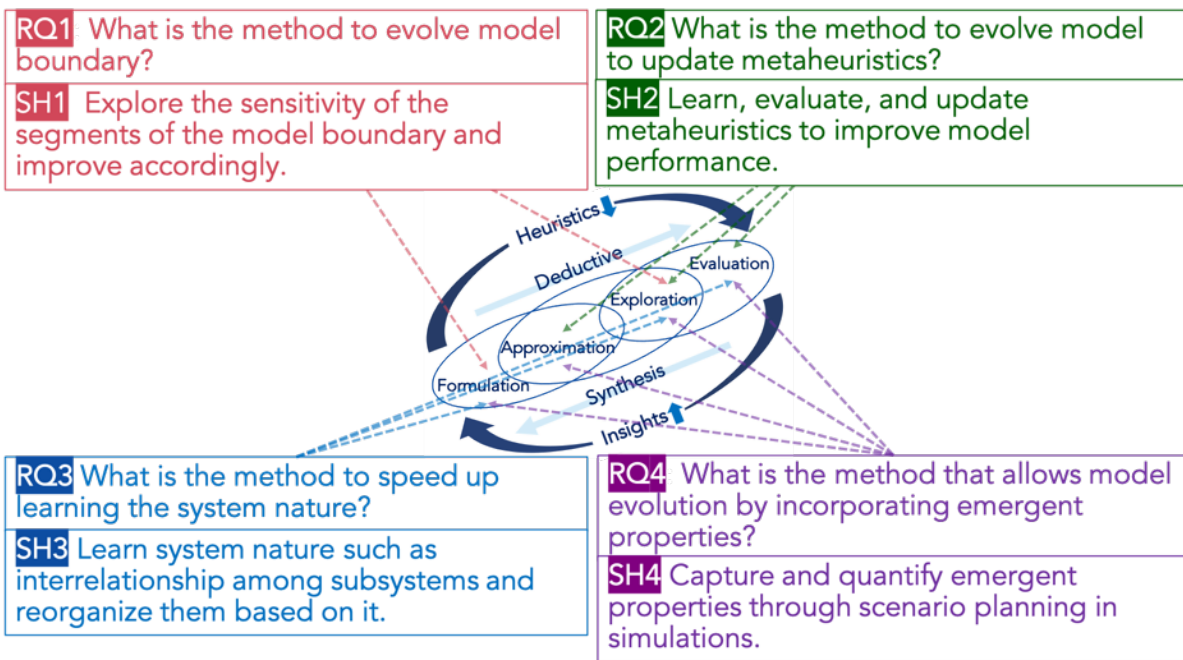
Given the four tasks and four research questions posed in Section 2.4, the hypotheses are specified into four hypotheses, SH1, SH2, SH3, and SH4, shown in Table 2.16, by testifying which, the research questions can be answered.

**Table 2. 16 Specification of Hypotheses for Answering the Research Questions**

Chapter	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Ch7	Ch8	Ch9
<b>Actions</b>	RG H	RQ1: What is the method to evolve model boundary?	<i>SH1: Explore the sensitivity of the segments of the model boundary and improve accordingly.</i>	TVe M	EVe SQT AQ			CQ EVa	TE
		RQ2: What is the method to evolve model to update metaheuristics ?	<i>SH2: Learn, evaluate, and update metaheuristics to improve model performance.</i>	TVe M		EVe SQT AQ			
		RQ3: What is the method to speed up learning the system nature?	<i>SH3: Learn system nature such as interrelationship among subsystems and reorganize them based on it.</i>	TVe M			EVe SQT AQ		
		RQ4: What is the method that allows	<i>SH4: Capture and quantify emergent properties through</i>	TVe M					

	passing the information through multiple scales of a system?	<i>scenario planning in simulation.</i>							
<b>Nomenclature</b>	RG – give research gaps H – give hypotheses RQ – research question SH – specify hypotheses TVe – theoretically verify hypotheses M – introduce methods EVe – empirically verify hypotheses SQT – specify research questions in the context of test problems AQ – answer research questions CQ – closure the answers to research questions EVa – empirically validate hypotheses TE – theoretically extend the research								

In Figure 2.28, the research questions RQ1-RQ4 and the specified hypotheses SH1-SH4 are visualized as the connections of different stages of the design cycle.



**Figure 2. 28 Research Questions RQ1-RQ4 and Specified Hypotheses SH1-SH4**

## 2.6 Role of Chapter 2 in this Dissertation

In Chapter 2, we justify the primary research question into an “answerable” level. We answer the “how question” – “how can we realize the model evolution using satisficing strategy?” We further

explain why we realize model evolution in a certain way – why we choose satisficing strategy, and specifically cDSP, ALP, and DSIDES, as the foundational method to process the tasks in the model evolution.

In Chapter 2, mathematical explanations on optimizing and satisficing strategy in engineering design regarding Kuhn-Tucker conditions are discussed. The differences between the two strategy is summarized and why in this dissertation, we choose cDSP, ALP and DSIDES to realize *satisficing* strategy for model evolution is stated. Then, given the requirements on the robust design and the elements of connections among different stages of model evolution, we justify the primary research question into research questions in two contexts – the robust design (RDI-II, RDIII, and RDIV) and model evolution (RQ1, RQ2, RQ3, and RQ4) and specify the hypotheses (SH1, SH2, SH3, and SH4) to answer the ; see Table 2.17. The organization of the later chapters regarding proposing the hypotheses for answering the justified research questions, specifying the research questions in the context of the test problems, and answering the research questions using the test problems, and extending the research questions in the way forward work is described.

**Table 2. 17 Plan of Addressing the Research Questions in Each Chapter**

Chapter	Ch1	Ch2		Ch3	Ch4	Ch5	Ch6	Ch7	Ch8	Ch9
Actions	RG H	RDI-II: What are the mechanisms and procedures that enable the exploration of the solutions relatively insensitive to Type I and Type II uncertainty?	RQ1: What is the method to evolve model boundary?	SH1: Explore the sensitivity of the segments of the model boundary and improve accordingly.	TVe M	EVe SQT AQ			CQ EVa	TE
			RQ2: What is the method to evolve model to update metaheuristics ?	SH2: Explore the sensitivity of the segments of the model boundary and improve accordingly.	TVe M	EVe SQT AQ				
		RDIII: What is the mathematics in the design								

	method that allow the exploration of the solutions relatively insensitive to Type III uncertainty?	RQ3: What is the method to speed up learning the system nature?	SH3: Learn system nature such as interrelationship among subsystems and reorganize them based on it.	TVe M			EVe SQT AQ		
	RDIV: What is the method that allows sensing and managing Type IV uncertainty?	RQ4: What is the method that allows passing the information through multiple scales of a system?	SH4: Capture and quantify emergent properties through scenario planning in simulation.	TVe M			EVe SQT AQ		
<b>Nomenclature</b>	RG – give research gaps H – give hypotheses RD – robust design RQ – research question SH – specify hypotheses TVe – theoretically verify hypotheses M – introduce methods EVe – empirically verify hypotheses SQT – specify research questions in the context of test problems AQ – answer research questions CQ – closure the answers to research questions EVa – empirically validate hypotheses TE – theoretically extend the research								

In Chapter 3, the feasibility of the hypotheses is theoretically verified, and the overview of the proposed methods and algorithms are introduced. In Chapter 4, 5, 6, and 7, the research questions are specified in different domains, test problems in those domains are used to prove the hypotheses and verify the proposed methods and algorithms. In Chapter 8, the answers to the research questions are summarized and the closure is given. In Chapter 9, the research way forward is narrated, with the extension of the research questions in the domain of way forward.



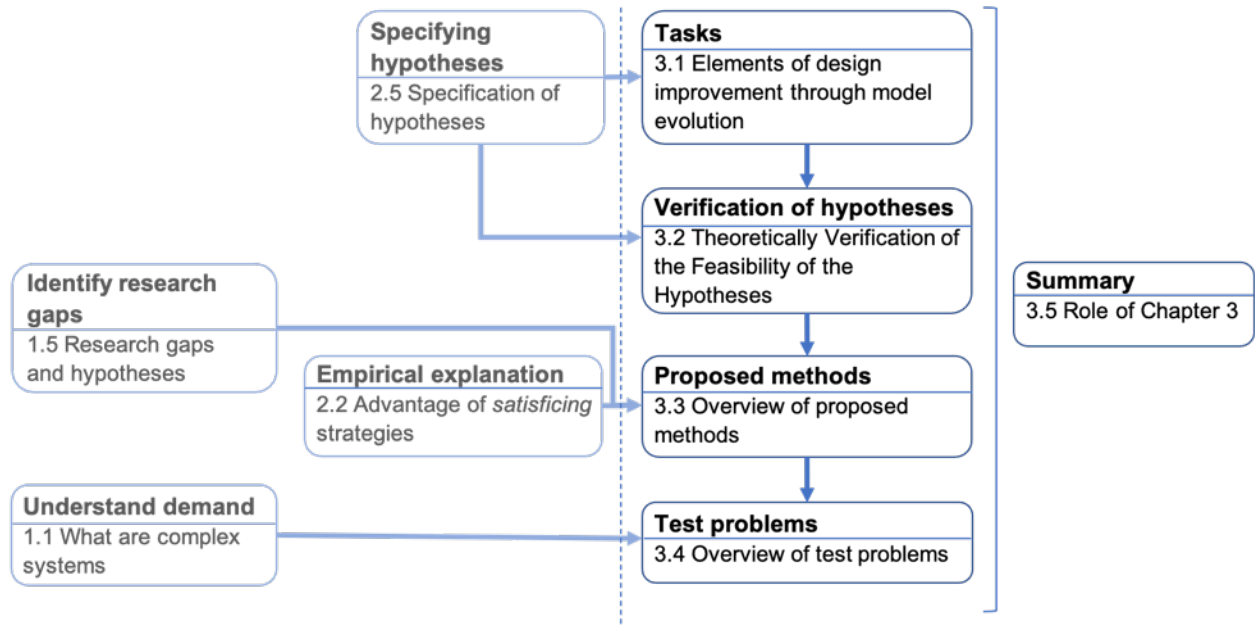
## **CHAPTER 3 PROPOSED METHODS – THE DESIGN EVOLUTION LOOP**

### ***– FOUNDATIONS FOR MODEL EVOLUTION***

*In Chapter 3, the hypotheses for answering the justified research question are stated. The “what question” is answered – “what tasks should designers finish to realize the model evolution?”*

*The methods using satisficing strategy are proposed to finish the tasks and testify the hypotheses.*

In Chapter 3, see Figure 3.1, in Section 3.1, introduced the elements of model evolution, which is an extension of the tasks in the design evolution cycle (in Section 2.4.2); in Section 3.2, stated the hypotheses, which are the tentative proposals for answering the justified research questions (in Section 2.4); based on the hypotheses, in Section 3.3, given an overview of the proposed methods which leverage *satisficing* strategy (discussed in Section 2.2) to fill research gaps (in Section 1.4); then, in Section 3.4, briefly introduced the test problems for testing the proposed methods, which are representative problems that contain typical characteristics of complex systems (in Section 1.1); finally, in Section 3.5, summarized the role of Chapter 3. The plan of addressing the research questions in Chapter 3 by theoretically verifying the feasibility of the specified hypotheses and proposing methods is shown in Table 3.1 and illustrated in Figure 3.2.



**Figure 3. 1 Organization of Chapter 3**

**Table 3. 1 Plan of Theoretically Verifying the Specified Hypotheses and Demonstrating the Proposed Methods in Each Chapter**

Chapter	Ch1	Ch2	Ch3		Ch4	Ch5	Ch6	Ch7	Ch8	Ch9
Actions	RG H	RDI-II	RQ1 SH1	<i>TVe1: Explore the boundary of the system by connecting formulation with exploration.</i>	<i>M1: Formulation-Exploration framework</i>	EVe SQT AQ				CQ EVa TE
			RQ2 SH2	<i>TVe2: Improve the approximation by connecting approximation, exploration, and evaluation.</i>	<i>M2: Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL)</i>		EVe SQT AQ			
		RDIII	RQ3 SH3	<i>TVe3: Speed up learning system nature by connecting formulation, exploration, and evaluation.</i>	<i>M3: Adaptive Leveling-Weighting-Clustering algorithm (ALWC)</i>			EVe SQT AQ		
			RDIV	RQ4 SH4	<i>TVe4: Learn emergent property by connecting formulation, approximation,</i>	<i>M4: Scenario planning in agent-based modeling</i>				

			<i>exploration, and evaluation.</i>							
<b>Nomenclature</b>	RG – give research gaps H – give hypotheses RD – tie to robust design RQ – pose research question SH – specify hypotheses TVe – theoretically verify hypotheses M – introduce methods EVe – empirically verify hypotheses SQT – specify research questions in the context of test problems AQ – answer research questions CQ – closure the answers to research questions EVa – empirically validate hypotheses TE – theoretically extend the research									

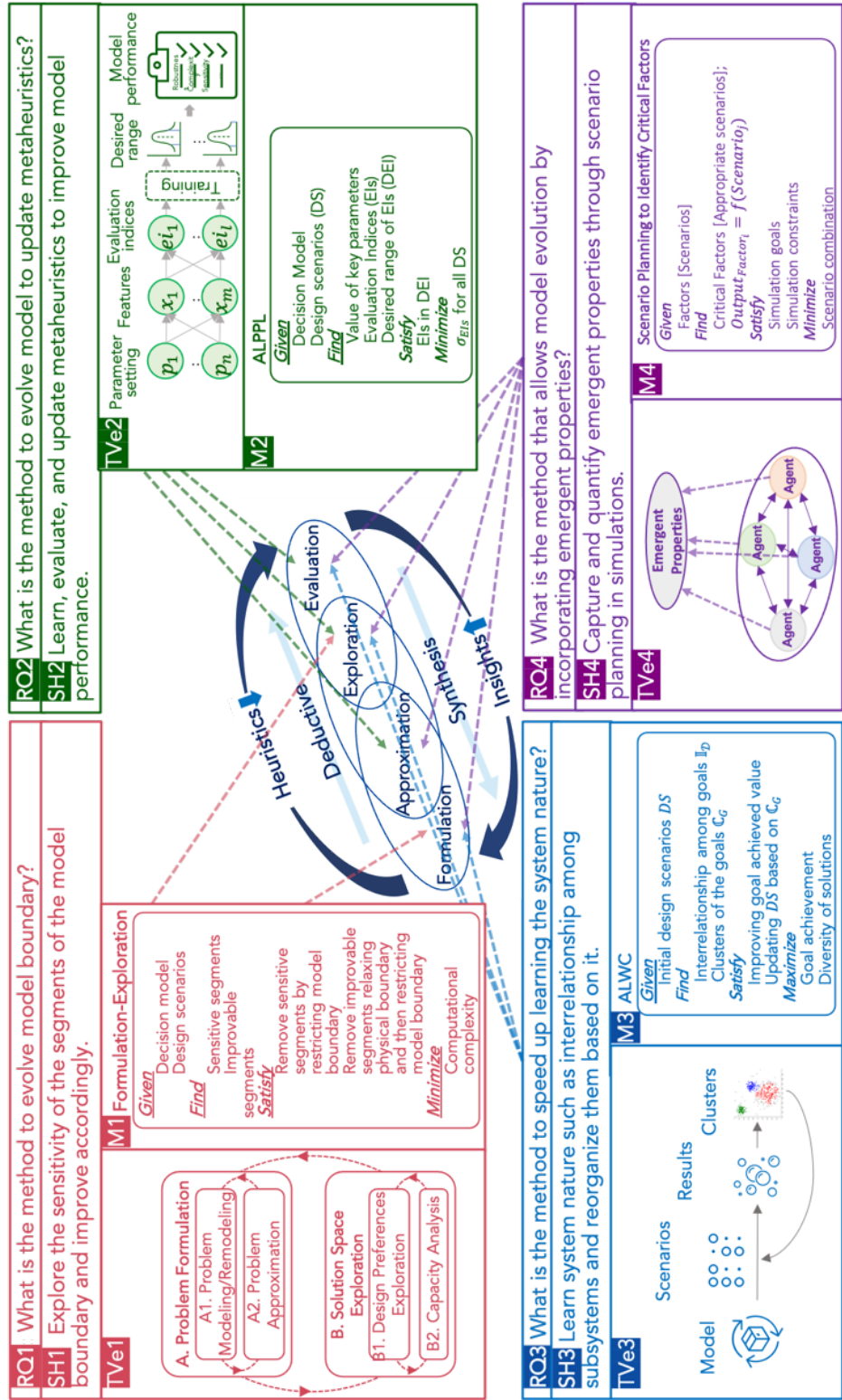


Figure 3. 2 Illustration of Research Questions (RQ1-RQ4), Specified Hypotheses (SH1-SH4), Theoretical Verification of Specified Hypotheses (TVe1-TVe4), and Methods (M1-M4) in the Context of Design Evolution Cycle

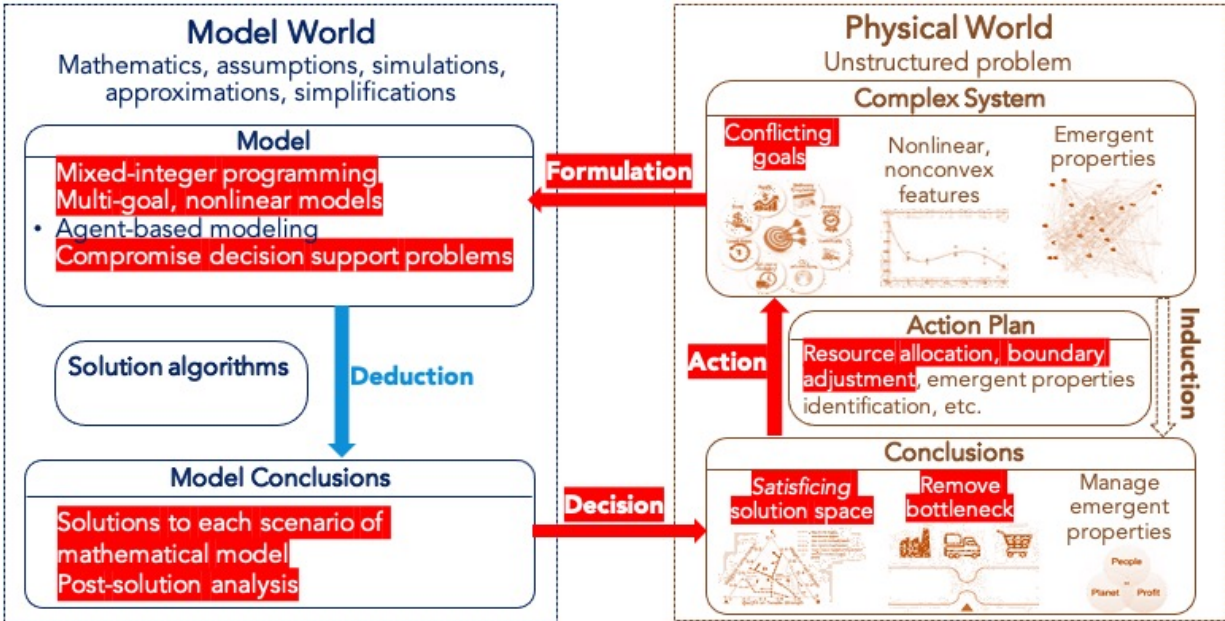
### **3.1 Elements of Design Improvement through Model Evolution (Task 1-4)**

The interactions between multiple stages in the evolution cycle of the realization of model-based complex systems are the elements in exploration of the solution space (ESS). In this dissertation, the elements being explored are described as follows.

#### ***3.1.1 Task 1: Formulation-Exploration***

As the boundary formed by the constraints and bounds may have variations due to 1) variation in parameters, 2) variation in decision variables, and 3) variation in model structure, the exploration of the boundary and the accordance reformulation allow the decision maker to improve the model accuracy and robustness.

Through establishing connections between model formulation and exploration, we expect to explore the model performance and pass the information to reformulate the model to improve the design accuracy and robustness. In Figure 3.3, we highlight the procedures and methods that are involved in getting the connections work.



**Figure 3. 3 The Methods and Procedures Involved in Formulation-Exploration – Realizing the model evolution through the items highlighted in red**

For example, in a dam network, the reservoir behind each dam has a lower bound – water level of the inactive pool, and an upper bound – water level of the flood pool, but there are dams with more strict bounds than the other, and those strict bounds are bottleneck of the whole dam network. To make the dam-network system more robust, the boundary should be explored so that the bottleneck can be pinpointed, thus buffer can be added to the mathematical model, therefore, the solution to the mathematical model can be away from the boundary of the physical system, which ensures the system robustness. The hypothesis on how to connect formulation with exploration is given in details in Section 3.2.1. The detailed description of the test problems and the method are in Section 4.2 and 4.3.

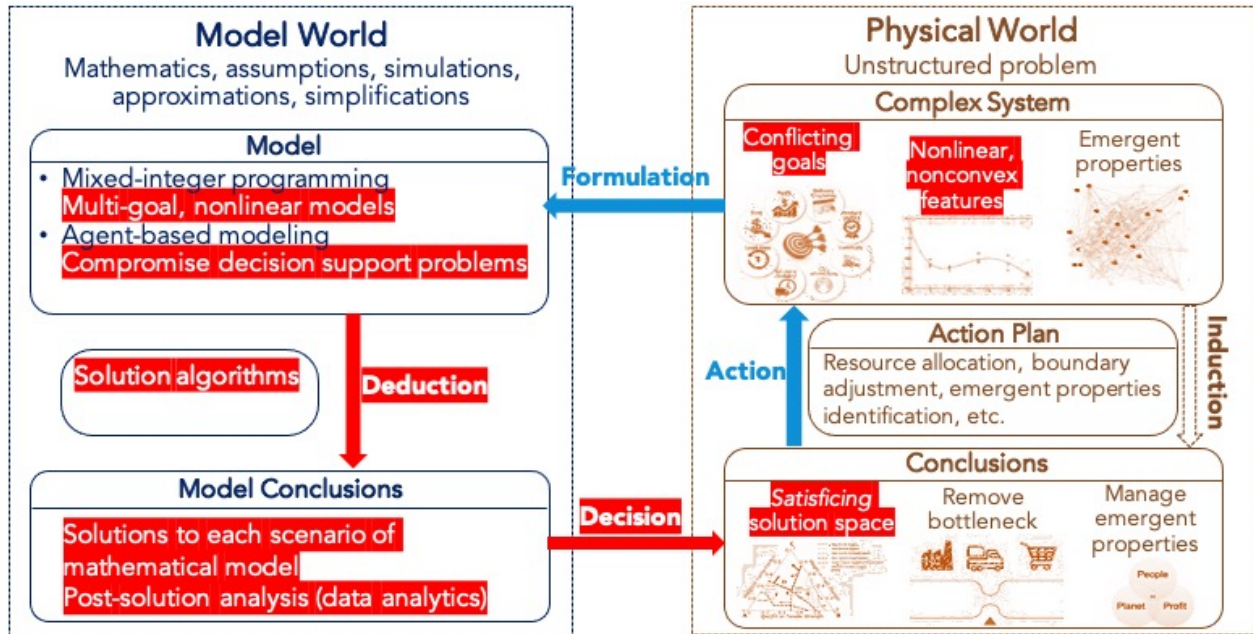
### **3.1.2 Task 2: Approximation-Exploration-Evaluation**

As we discuss in Section 1.2, design methods and solution algorithms for dealing with complex problems fall into two categories: formulate a complex problem exactly and solve it approximately

or approximate a complex problem and solve it exactly. For both methods, designers need to approximate the physical problem, what matters to the robustness of the result is when and how the approximation takes place. In this dissertation, we believe that any approximation relies on heuristics or metaheuristics to make rules. A heuristic is a mental shortcut that allows people to solve problems quickly and efficiently. A metaheuristic is a higher-level procedure to find a heuristic that may provide a sufficiently good solution to a problem. There are no perfect heuristics or metaheuristics but given the evolving knowledge and demand of a problem, designers can use the knowledge to update the metaheuristics or heuristics so as to improve the robustness of approximation and hence improve the design.

The utility of *evaluation* is processing data analytics to learn patterns in the data generated by adopting a large number of different scenarios. Based on the evaluation, predictions on model performance regarding more scenarios in heuristics updates can be done, hence, designers can evolve the model through improving the approximation.

In Figure 3.4, we highlight the procedures and methods that are involved in getting the connections work. Through establishing connections among model approximation, exploration, and evaluation, we expect to explore the model performance associated with the metaheuristics used in model approximation and pass the information to update the metaheuristics to improve the design to be *satisficing* and relatively less insensitive to metaheuristics.



**Figure 3. 4 The Methods and Procedures Involved in Approximation-Exploration-Evaluation – Realizing the model evolution through the items highlighted in red**

As the approximation of the model, for example, the linearization, may have impact on the model accuracy and the solution quality, capturing the connections between the critical parameters in the approximation algorithm and the solution quality is critical to the model evolution. Heuristics are used in the determination of the critical parameters of the approximation algorithm, such as the reduced move coefficient (RMC) in the ALP. If the quality of the solution can be clearly defined, and the association between the quality of the solution and the RMC value can be established, then the solution quality as well as the robustness of the model can be improved by adjusting the RMC. The hypothesis on how to connect approximation with exploration and evaluation is in Section 3.2.2. The detailed description of the test problem and the method are in Chapter 5.

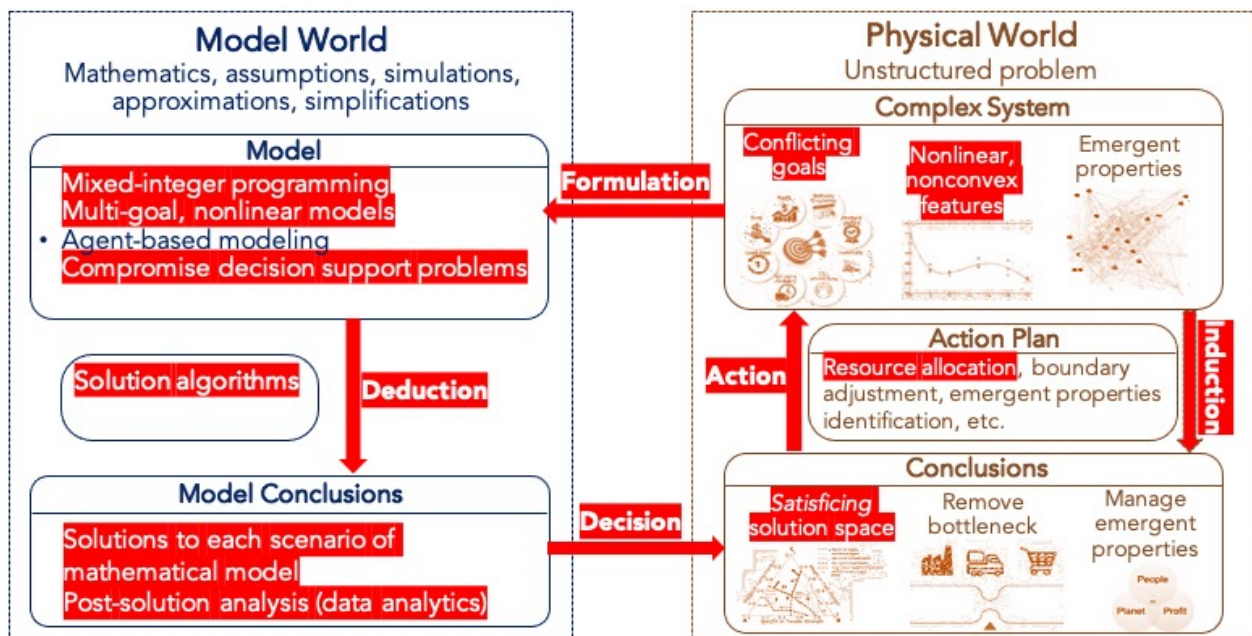
### **3.1.3 Task 3: Formulation-Exploration-Evaluation**

The structure of the model or any heuristics, assumptions, or simplifications applied consciously or subconsciously by the designers in model formulation can be evaluated and updated to improve



the design robustness, if a connection among formulation, exploration, and evaluation is established and information can be interpreted into knowledge. Designers need to identify what knowledge obtained from the exploration and evaluation procedure is useful. The knowledge on the improvement of the model formulation that can benefit the accuracy and robustness of the model. The improvement of model formulation is a huge topic, including the setting of parameters, variables, equations between parameters and variables, the formulation of the goals, the priority and combination of the multiple goals, etc. A method of passing through the information from the solution space back to the design space regarding the reformulation of the model is required.

In Figure 3.5, we highlight the procedures and methods that are involved in getting the connections work. Through establishing connections among model formulation, exploration, and evaluation, we expect to explore the better model formulations regarding the information learned from post-solution analysis, such as the interrelationship among subsystems.



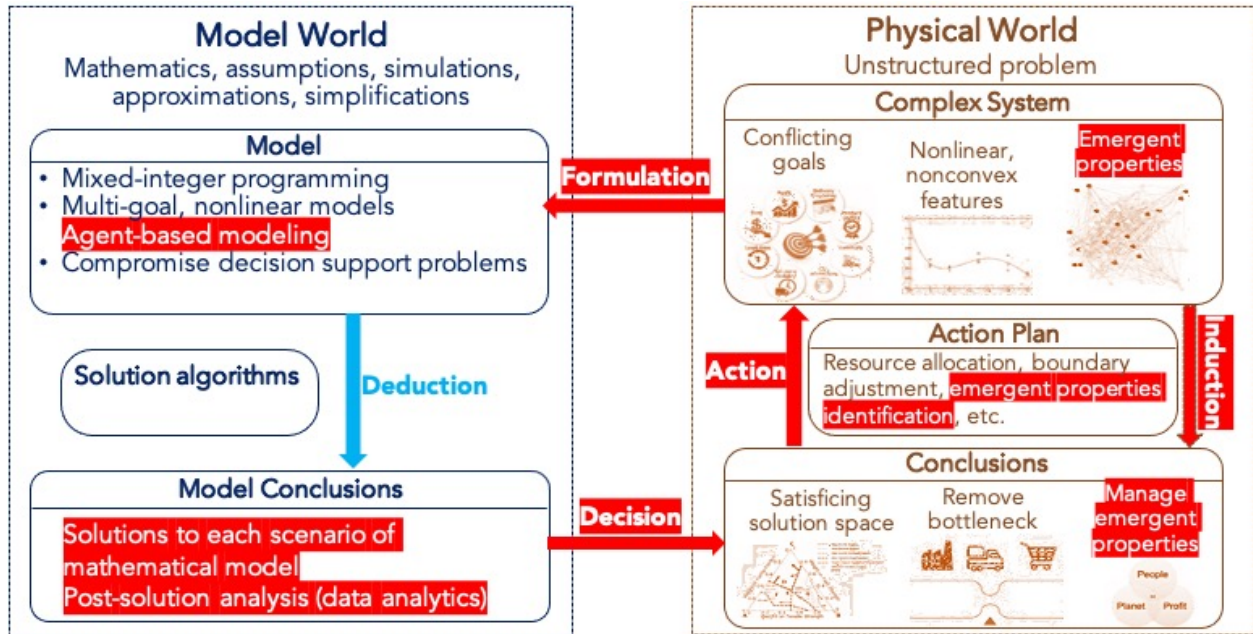
**Figure 3. 5 The Methods and Procedures Involved in Formulation-Exploration-Evaluation – Realizing the model evolution through the items highlighted in red**

For example, in a concurrent design problem with multiple subsystems, especially when the correlations and interactions among the subsystems are initially unknown to the designers, through analyzing the solutions generated by varying scenarios, interrelationships among the subsystems can be learned, so the model formulation can be adjusted based on such knowledge. The hypothesis on how to connect formulation with exploration and evaluation is given in details in Section 3.2.3. The detailed description of the test problem and the method are in Chapter 6.

#### ***3.1.4 Task 4: Formulation-Approximation-Exploration-Evaluation***

To manage the emergent properties of complex systems, which is to take into account “time” as a dimension in the design and update all four stages of the design along with the time. Setting the behavior rules of each individual or component of a complex system may not result in an expected system behavior. Therefore, to establish mechanisms of capturing emergent properties and incorporate them into the procedures of the evolution cycle is an element of the model evolution.

In Figure 3.6, we highlight the procedures and methods that are involved in getting the connections work. Through establishing connections among model approximation, exploration, and evaluation, we expect to capture the emergent properties of complex systems and incorporate them into the model.



**Figure 3. 6 The Methods and Procedures Involved in Approximation-Formulation-Exploration-Evaluation – Realizing the model evolution through the items highlighted in red**

For example, in a social network, how can designers capture emergent properties in human behavior and promote a new technology by leveraging critical and sensitive factors? The hypothesis on how to connect all four stages in the model evolution is given in details in Section 3.2.4. The detailed description of the test problem and the method are in Chapter 7.

### ***3.1.5 Model Evolution Cycle is an Open and Extendable Framework***

There can be other elements in the model evolution. More methods and examples can enrich the topic. The conception and procedure of model evolution is open and can be expanded to other relevant activities. In this dissertation, the focus is on the elements mentioned on the above. Based on the conception and elements of the evolution cycle, hypotheses are proposed in Section 3.2.

## **3.2 Theoretically Verification of the Feasibility of the Specified Hypotheses (TVe1-TVe4)**

To realize the four elements of the model evolution as well as answer the four research questions (RQ1-RQ4), the hypotheses are specified into four hypotheses (SH1-SH4). The theoretical verification of the feasibility of the four specified hypotheses are given hereafter.

### ***3.2.1 Theoretical Verification of Specified Hypothesis 1 (TVe1)***

**Research Questions 1 (RQ1)** – *What is the method to evolve model boundary?*

**Specified Hypothesis 1 (SH1)** – *Explore the sensitivity of the segments of the model boundary and improve accordingly.*

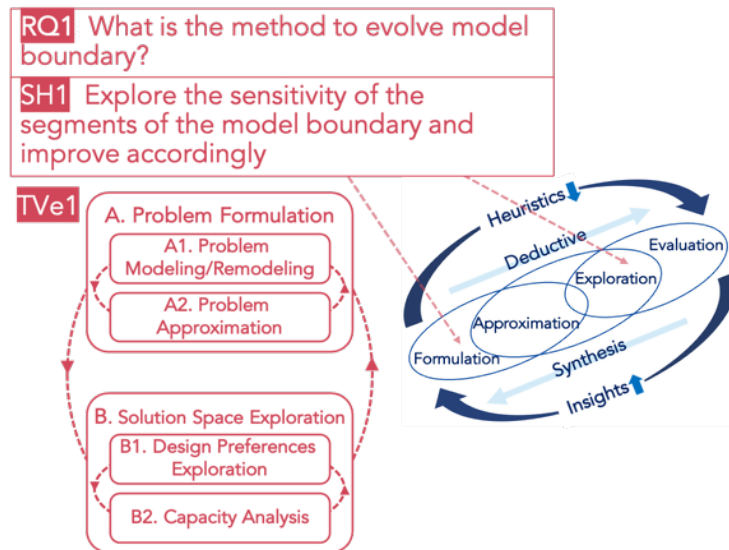
It is observed that the model formulation especially for engineering-design problems is usually based on designers' domain expertise, experience, some established conventions, assumptions, simplification, or initial settings, which can be wrong or evolving over time. To refine the boundary and make the solution space relatively insensitive to the errors or uncertainties embodied in or affected the model boundary, we hypothesize to explore the sensitivity of the segments of the model boundary to multiple versions of extreme uncertainties; see Figure 3.7.

It is *hypothesized* that through the outer cycle, which is to establish connections between problem formulation and solution space exploration, the forward information (on the next scenarios for solution space exploration) and backward information (on the next action plan for model improvement) can help evolve the model boundary and improve the design performance regarding the robustness, accuracy, and computational complexity. In this dissertation, the design robustness means the capacity of the solutions to be insensitive to the model errors and variations, the design accuracy means the representativeness of the scenarios of the design, and the computational complexity means the complexity of the calculations to obtain *satisficing* results.

It is *hypothesized* that through the inner cycles, which is to establish connections between different procedures in problem formulation or solution space exploration, the two-way information transmission mechanisms can also make each procedure gradually improve during iterations.

The rule-based connection mechanism is based on heuristics, but in Hypothesis 1, we did not evaluate the mechanisms and update the heuristics. In Specified Hypothesis 2 and 3, we incorporate evaluation in the multi-stage connections and manage the heuristics updating.

Given the hypothesis, we propose the method, Formulation-Exploration framework, which is briefly introduced in Section 3.3.1 and further introduced and testified using two test problems in Chapter 4.



**Figure 3. 7 Theoretical Verification of Specified Hypothesis I (TVe1) – Exploring the sensitivity of the segments of the model boundary and improve accordingly**

### 3.2.2 Theoretical Verification of Specified Hypothesis 2 (TVe2)

**Research Questions 2 (RO2)** – *What is the method to evolve model to update metaheuristics?*

**Specified Hypothesis 2 (SH2)** – *Learn, evaluate, and update metaheuristics to improve model performance.*

Model approximation especially for nonlinear, nonconvex, and multi-goal, engineering-design problems is usually based on rules which are made through designers' metaheuristics and experience, which can be wrong, inaccurate, or over-simplified. To improve the rule-based approximation regarding its accuracy, efficiency, and robustness relatively insensitive to metaheuristics, we hypothesize to learn, evaluate, and update metaheuristics to improve the model performance; see Figure 3.8.

It is *hypothesized* that by identifying the metaheuristics that have critical impact on the results in terms of solution robustness, such as (but are not limited to) settings of parameters, designers can learn the features that represent the performance of the approximation. From the features, evaluation indices can be created. Designers need to train the metaheuristics to bring the evaluation indices into desired range which ensure the approximation performance and the insensitivity of the solution space with respect to model errors, uncertainties, and metaheuristics changing.

Given the hypothesis, we propose the method, using parameter learning to improve approximation and solution algorithms, which is briefly introduced in Section 3.3.2 and further discussed and demonstrated using a test problem in Chapter 5.

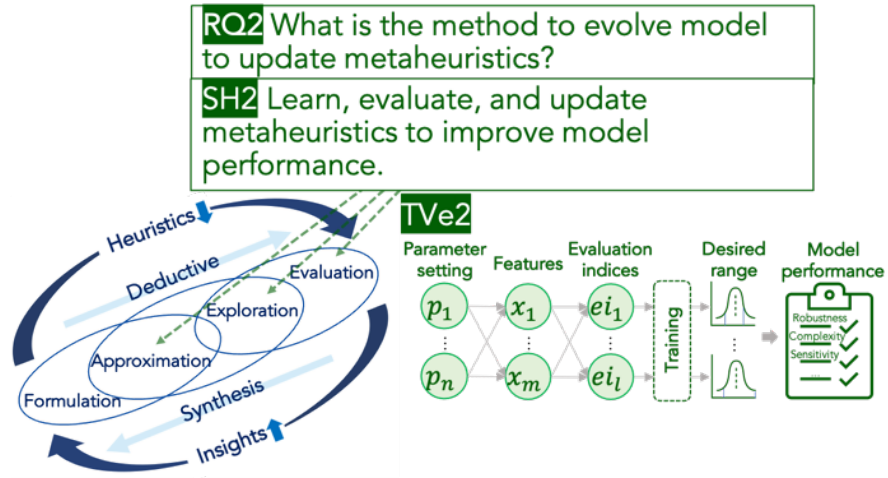


Figure 3. 8 Theoretical Verification of Specified Hypothesis 2 (TVe2) – Learn, evaluate, and update metaheuristics to improve model approximation

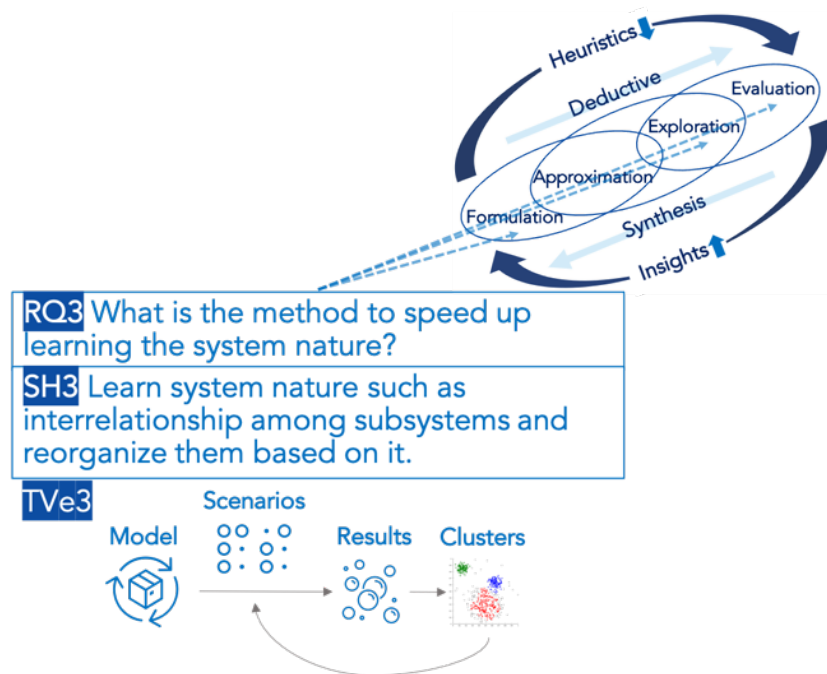
### 3.2.3 Theoretical Verification of Specified Hypothesis 3 (TVe3)

**Research Questions 3 (RO3)** – *What is the method to speed up learning the system nature?*

**Specified Hypothesis 3 (SH3)** – *Learn system nature such as interrelationship among subsystems and reorganize them based on it.*

For multi-stage, concurrent design problems, initially, designers may have no idea on the profound knowledge about the system nature, for example, the awareness and the interrelationship of subsystems. For some problems, designers even do not aware subsystems exist and may help smooth the design if they can be organized appropriately. If there is not sufficient domain expertise, or worse, if designers thought they have domain knowledge or experience that helps them proceed with the structuring of subsystems, but their so-called knowledge or experience is wrong or unpractical, It is ***hypothesized*** that there is information in the solution space that can be learned through post-solution analysis and can be interpreted into insight on how to select the most representative scenarios that better identify and organize the subsystems. It is ***hypothesized*** that

such an analyzing-interpreting-selecting process can help speed up designers' learning the system's nature.; see Figure 3.9.



**Figure 3.9 Theoretical Verification of Specified Hypothesis 3 (TVe3) – Learn system nature such as interrelationship among subsystems and reorganize them based on it**

The utility of adding an "evaluation" step to the formulation-exploration (Hypothesis 1) is extracting hidden information through learning from time-series data generated along iterating the model evolution cycle. We propose the method, using unsupervised learning to cluster the system into subsystems, learn their interrelationship, and update the design scenario based on it, which is briefly introduced in Section 3.3.3 and further discussed and demonstrated using a test problem in Chapter 6.

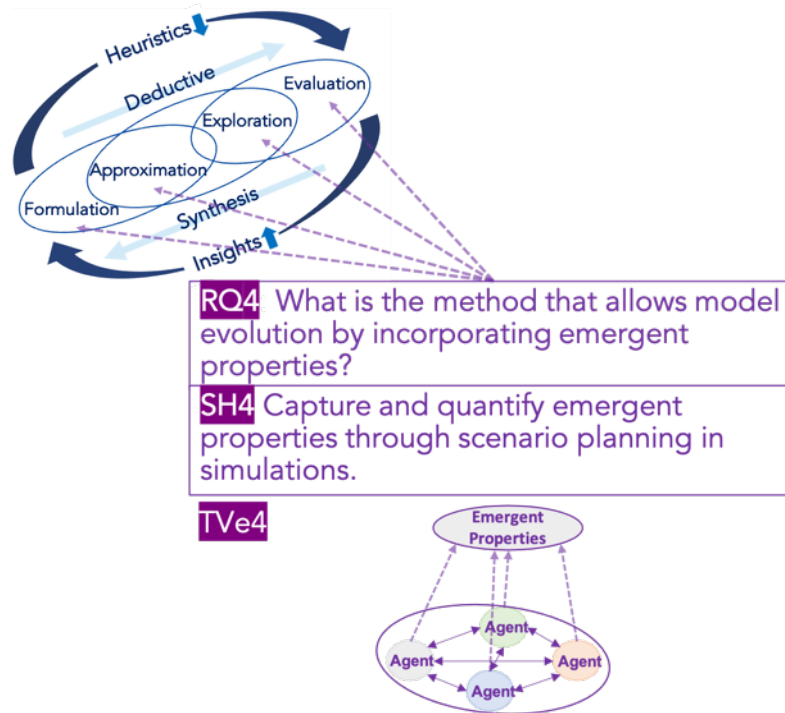
### 3.2.4 Theoretical Verification of Specified Hypothesis 4 (TVe4)

**Research Questions 4 (RQ4) – What is the method that allows model evolution by incorporating emergent properties?**



***Specified Hypothesis 4 (SH4) – Capture and quantify emergent properties through scenario planning in simulation.***

“In planning and policy, a wicked problem is a problem that is difficult or impossible to solve because of incomplete, contradictory, and changing requirements that are often difficult to recognize.”<sup>11</sup> It is *hypothesized* that by capturing emergent properties and incorporate them into decision models as an element of model evolution, designers can convert wicked problems into regular complex-system design problems. It is *hypothesized* using simulation tools such as agent-based modeling, emergent properties like collective behavior which is nonlinear with individual behaviors can be observed and quantified and then predicted; see Figure 3.10.



**Figure 3. 10 Theoretical Verification of Specified Hypothesis 4 (TVe4) – Capture and quantify emergent properties through scenario planning in simulations**

<sup>11</sup> This definition of “wicked problem” is from Wikipedia: [https://en.wikipedia.org/wiki/Wicked\\_problem](https://en.wikipedia.org/wiki/Wicked_problem)

It is *hypothesized* that learning and adding emergent properties to decision models can connect and affect all four stages of model evolution and result in design improvement. We propose the method, using simulation tools to generate synthetic data under a variety of design scenarios for a wicked problem, capture cluster the system into subsystems, learn their interrelationship, and update the design scenario based on it, which is briefly introduced in Section 3.3.4 and further discussed and demonstrated using a test problem in Chapter 7.

### **3.3 Overview of Proposed Methods (M1-M4)**

To testify the four specified hypotheses (SH1-SH4), four methods (M1-M4) are proposed. They are briefly introduced in this section one-by-one. Empirical demonstrations are done using test problems in Chapter 4, 5, 6, and 7.

#### ***3.3.1 M1: Exploration of the Boundary using Formulation-Exploration Framework***

To realize and test Hypothesis 1, *explore the sensitivity of the segments of the model boundary and improve accordingly*, we propose the Formulation-Exploration framework to establish connections between the two stages, the formulation and exploration. The conception of the method in a word-form cDSP is written as follows. Given the decision model and design scenarios, designers need to find the sensitive segments and improvable segments of the decision model, which are to be removed, meanwhile minimizing the computational complexity.

#### **Establishing connections between Formulation and Exploration**

##### **Given**

Decision model

Design scenarios

### **Find**

Sensitive segments

Improvable segments

### **Satisfy**

Remove sensitive segments by restricting model boundary

Remove improvable segments relaxing physical boundary and then restricting model boundary

### **Minimize**

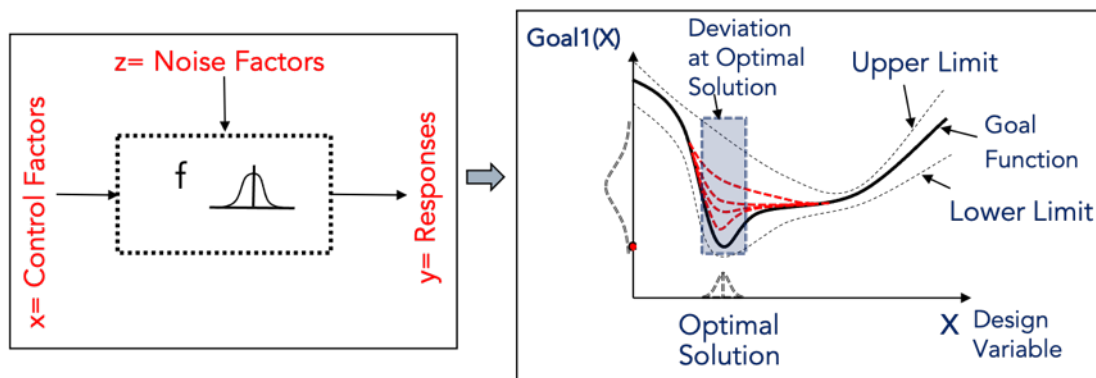
Computational complexity

***Sensitive segments*** mean the parameters or equations (response surface) that may embody Type I and Type II uncertainty, that are variation in parameters or unparameterizable factors (noise factors) or variables (control factors), or when any Type I or Type II uncertainty takes place, these segments change and lose a feasible solution; see Figure 3.11. For example, if an inequality constraint  $g_i(x) \geq 0$  becomes active at a solution point  $\mathbf{x}^o$ ,  $g_i(\mathbf{x}^o) = 0$ , or in other words, when plugging in a solution into an inequality constraint, the slack or surplus of the constraint is zero, then such a constraint is a sensitive segment of the decision model under the design scenario when  $\mathbf{x}^o$  is a solution point. By changing design scenarios, designers obtain a set of solution points,  $X$ , and if  $g_i(\mathbf{x}^o) = 0, \forall \mathbf{x}^o \in X$ , then constraint  $g_i(x) \geq 0$  is a sensitive segment.

***Control factors*** are the factors that under designers' direct control. It is assumed that a control factor is accurately quantifiable and controllable, so they can only be decision variables of the

decision model. *Noise factors*, on the contrary, cannot be controlled by designers directly. Sometimes noise factors are not even quantifiable or parameterizable, and they may not be incorporated into the decision model. However, initially, designers' understanding of control factors and noise factors is insufficient to make decisions regarding Type I and Type II uncertainty, actions to explore solution space, such as design preferences exploration and capacity analysis, can help identify the sensitive segments.

Likewise, *improvable segments* are the parameters or equations that can be modified, and when modify them in certain ways, the goal(s) can be achieved better. For example, a constraint with a large dual price (or shadow price) is an improvable segment. The dual price is the amount that the goal (objective) would be improved (achieved more completely) as the right-hand side of the constraint is relaxed by one unit.

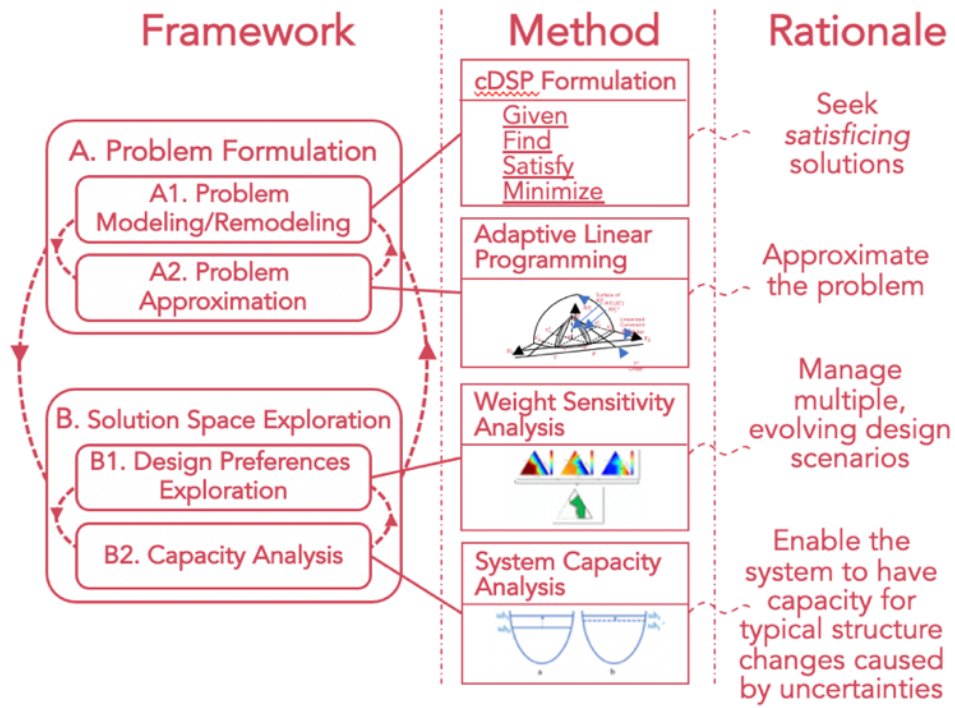


**Figure 3. 11 The Control Factors and Noise Factors Bring Variation in Goal Function**

The method of each step in the framework and the rationale of each method is illustrated in Figure 3.13. To realize *satisficing* strategy in engineering design, we choose the cDSP as the construct to formulate design problems and the ALP algorithm to approximate the nonlinear, non-convex problems. The reason why using the cDSP and the ALP can identify *satisficing* solution space is in Section 2.1. In solution space exploration, there can be more activities. In this dissertation, we

talk about weight sensitivity analysis, which is exploring weight vectors to combine the multiple goals, and capacity analysis, which is to identify the constraints with limited capacity.

In Chapter 4, we use two test problems – a continuous problem and a discrete problem to demonstrate how to use the formulation-exploration framework to explore the boundary of a problem and make the design relatively insensitive to boundary variation.



**Figure 3. 12 Formulation-Exploration Framework**

### 3.3.2 M2: Improving Algorithm Robustness using Parameter Learning

To realize and test Hypothesis 2, *learn, evaluate, and update metaheuristics to improve model performance*, we propose to enable information passing through among approximation, exploration, and evaluation, by adopting the Adaptive Linear Programming algorithm with Parameter Learning (ALPPL). The conception of the method in a word-form cDSP is written as follows. Given the decision model and design scenarios (DS), designers need to find the

appropriate value of key parameters through making the evaluation indices (EIs) fall into desired range (DEI), meanwhile minimizing the standard deviation of the EIs under all DS. The standard deviation is one of the statistics chosen to represent the robustness of the solution with respect to the DS changing for the test problem, which is to improve the ALP algorithm by updating heuristics in parameter setting.

**Given**

Decision Model

Design scenarios (DS)

**Find**

Value of key parameters

Evaluation Indices (EIs)

Desired range of EIs (DEI)

**Satisfy**

EIs in DEI

**Minimize**

$\sigma_{EIs}$  for all DS

***Evaluation indices*** (EIs) are the indices that can effectively reflect the approximation performance of a design regarding the criteria that designers desire, such as the approximation accuracy, the robustness of a solution with respect to the approximation accuracy, and the computational complexity of the approximation. In this dissertation, it is emphasized that evaluation indices can

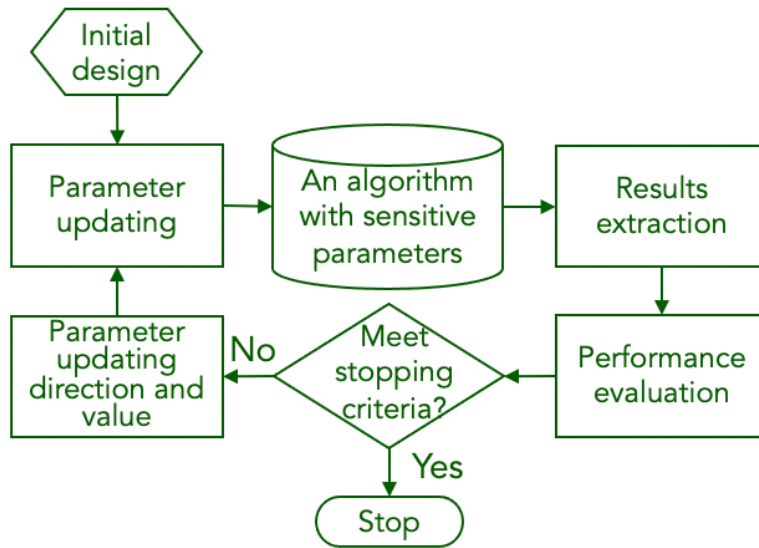
be customized to each problem. Evaluation indices can be developed by designers based on their domain knowledge or experience, and they can also be trained through learning the data generated through exploring the solution space. The difference between the two ways of obtaining evaluation indices is whether incorporating the evaluation procedure into the design cycle. In this dissertation, “evaluation” is defined as the procedure that requires using technologies in data analytics to analyze the data and gain knowledge among different factors belong to multiple stages in the design cycle.

***Desired range of EIs*** (DEI) is a range that when the value of the evaluation index falls in the range, the approximation performance can be guaranteed to an acceptable level. By identifying the desired range of each evaluation index, designers can obtain knowledge on the association between the evaluation indices and the approximation performance. By bringing the evaluation indices into their desired ranges, the designers can ensure that the approximation of the design is acceptable.

While changing design scenarios and parameter values, designers obtain results whose evaluation indices follow some distribution. Statistics are used to control the robustness and computational complexity of the solutions whilst choosing appropriate design scenarios.

The steps added to the ALP are illustrated in Figure 3.13. After initializing the decision model and the algorithm with certain parameter setting, designers can run the algorithm and obtain results. By extracting the results and interpreting them to evaluate the performance of approximation, insight on how to further update the parameter is obtained and used in the next iteration.

In Chapter 5, we use a test problem on designing the cooling stage of the hot rod rolling chain to demonstrate how to use parameter learning to improve an approximation and solution algorithm.



**Figure 3. 13 Learn and Update Metaheuristics in an Algorithm Using Parameter Learning**

### 3.3.3 M3: Exploring Interrelationships among Subsystems using Unsupervised Learning

To realize and test Hypothesis 3, *Learn system nature such as interrelationship among subsystems and reorganize them based on it*, we propose the Adaptive-Leveling-Weighting-Clustering (ALWC) algorithm to establish connections among formulation, exploration, and evaluation. The conception of the method in a word-form cDSP is written as follows. Given the decision model and initial design scenarios, designers need to find the interrelationship among goals, based on which, designers can update the combination format of the goals. Through the process, it is expected the achieved value of the goals are improved and the most representative design scenarios can be identified and explored rather than enumerating all scenarios. Meanwhile, designers can maximize the goal achievement and the diversity of the solutions.

#### **Given**

Decision model

Initial design scenarios (DS)



### **Find**

Interrelationship among goals ( $\mathbb{I}_D$ )

Clusters of the goals  $\mathbb{C}_G$

### **Satisfy**

Improving goal achieved value

Updating DS based on  $\mathbb{C}_G$

### **Maximize**

Goal achievement

Diversity of solutions

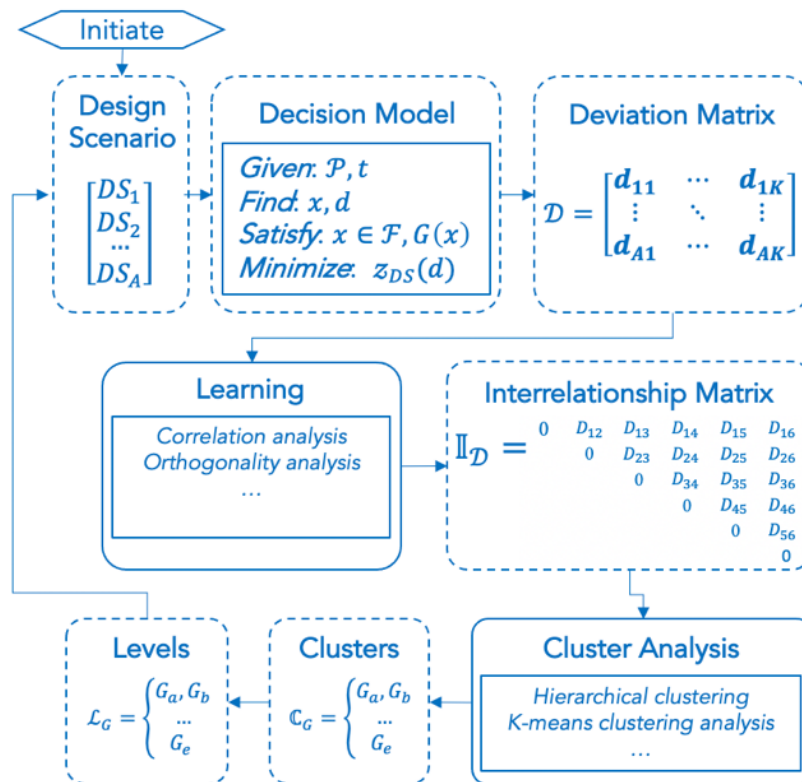
For this method, we assume that each goal represents the interest of a subsystem, or certain formats of the combination of several goals may represent the interest of a subsystem, so, exploring the interrelationship among the goals is a way to learn the categorization of subsystems of the whole system and learn the interrelationship among the subsystems.

The *interrelationship* of the goals can be one or more measurement, such as correlation among the achieved values of the goals under multiple design scenarios, or orthogonality among the deviation of the goals under multiple design scenarios. Whether to choose a measurement depends on the nature of the problem or designers' preferences.

The *diversity of solutions* is defined as how different the solutions are from one another. The diversity of the solutions can reflect whether the solution space is explored sufficiently and whether there are adequate alternative solutions for designers to select in various situations.

The procedures of the ALWC algorithm are given in Figure 3.14. After initializing the design scenarios on the organization of subsystems, designers adopting the design scenarios and obtain a deviation matrix. A column of the deviation matrix is the deviation of all goals under one design scenario. The deviation is the distance between the achieved value of a goal and the target of the goal. By learning the interrelationship among the rows of the deviation matrix, designers obtain the correlation, or orthogonality, or other types of interrelationship among the goals. Based on such interrelationship, the goals can be grouped into clustered, based on which, the goals can be reorganized in the next iteration, by using the most representative design scenarios.

In Chapter 6, we use a test problem of concurrent engineering designing to demonstrate how to use the proposed method to learn and speed up learning the system nature.



**Figure 3. 14 Learn and Speed up the Learning of Systems using Machine Learning**

### ***3.3.4 M4: Exploring Critical Factors through Scenario Planning in Agent-Based Modeling***

To realize and test Hypothesis 4, *Capture and quantify emergent properties through scenario planning in simulation*, we propose a framework to learn critical factors in simulations. The conception of the method in a word-form cDSP is written as follows. Given all the factors that may or may not affect the simulation results and the scenarios of each factor, find the critical factors that severely impact the simulation results and the appropriate scenarios of those factor that result in desired output, meanwhile satisfying simulation goals and constraints and minimizing the number of the combinations of scenarios.

#### **Given**

Factors [Scenarios]

#### **Find**

Critical Factors [Appropriate scenarios]

$\text{Output}_{\text{Factor}_i} = f(\text{Scenario}_j)$

#### **Satisfy**

Simulation goals

Simulation constraints

#### **Minimize**

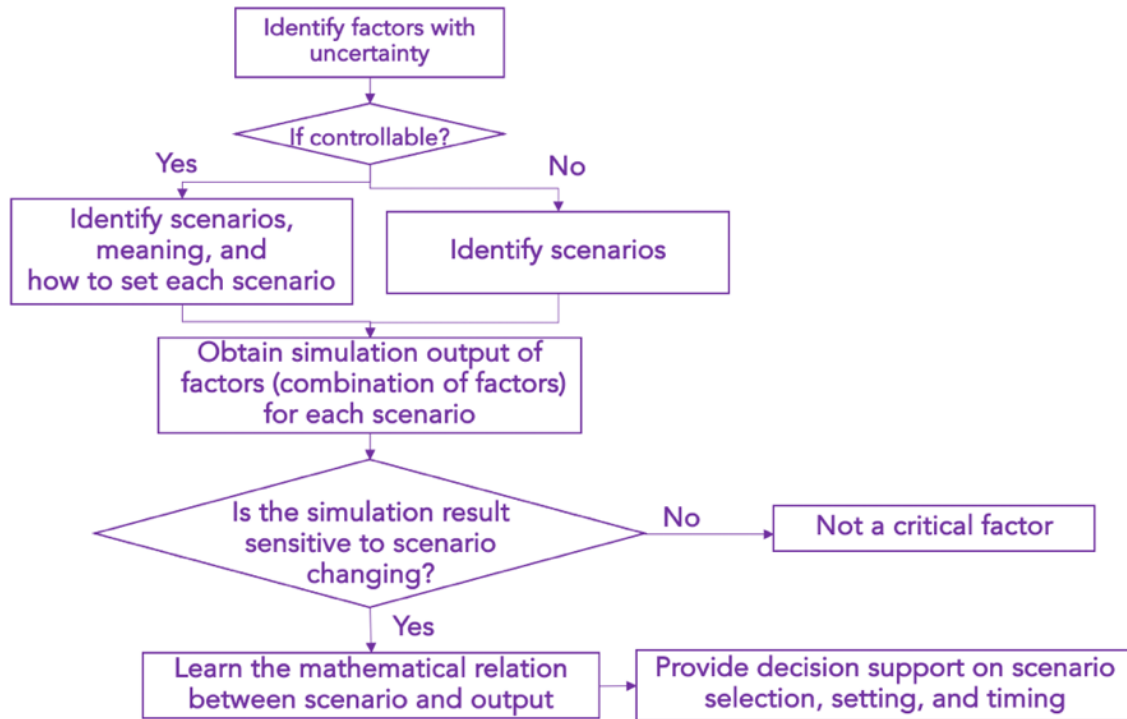
Scenario combination

*Factors* are variables or parameters with a range or multiple scenarios, or with uncertainties, and designers lack knowledge of critical factor setting and the corresponding simulation results, especially the simulation output under the influence of the combination of multiple scenarios of different factors. Among all the factors, some have a more significant impact on the simulation results when changing scenarios, while others do not.

*Critical factors* are the factors that have relatively significant impact on the simulation results when applying different scenarios. The aim of learning the emergent properties is to identify the critical factors, quantify their impacts on the simulation results, and find the certain scenarios that lead to desired results.

The procedures of learning critical factors in a simulation such as agent-based modeling are shown in Figure 3.16. After identifying the factors with uncertainties, designers need to make a judgment whether they are controllable or not. For the controllable factors, designers need to take them as decision variables, identify the possible scenarios of each factor, the physical meaning of each scenario, and the ways of setting each scenario. For the uncontrollable factors, designers need to set them as parameters with uncertainties or unparameterizable noise factors and identify their possible scenarios as well. Then, by exploring the scenarios and the combination of scenarios, they obtain simulation output of each scenario or combination of scenarios. For the factor that the simulation results do not vary significantly whilst changing its scenarios, designers can treat it as a noncritical factor. Otherwise, it is a critical factor, so designers can learn the mathematical relation between the scenarios and the simulation outputs, based on which decision support on scenario selection, setting, and timing is acquired.

In Chapter 7, we use a test problem of identifying the critical factors in intervening a social system to demonstrate how to use the proposed method to learn and leverage critical factors in a simulation.



**Figure 3. 15 The Process of Learning Critical Factors in a Simulation**

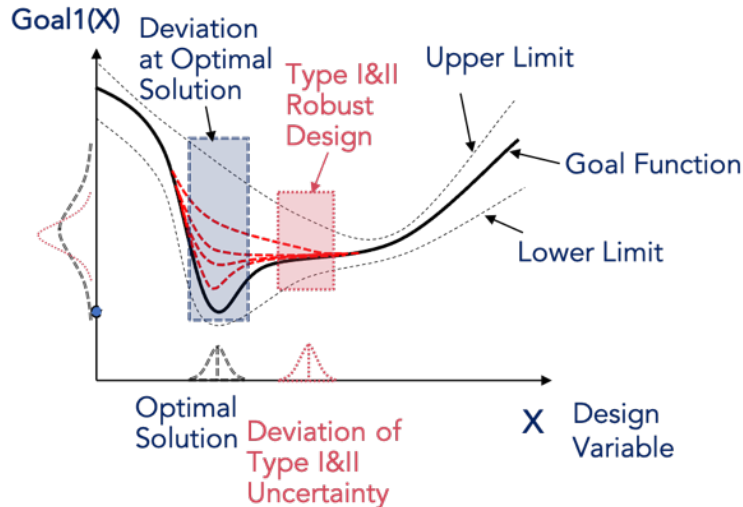
### 3.4 Overview of Test Problems

In this section, the connections and transitions between the theoretical verification and the empirical verification of the methods are given, that is, the logic flow between the Quadrant 1 (Q1) and Quadrant 2 (Q2) of the validation square (Figure 1.11 in Section 1.6).

#### 3.4.1 Required Characteristics of the Test Problems

To realize the RDI-II (robust design Type I and Type II) to identify *satisficing* solution space that is relatively insensitive to the variation in parameters and variables cause segments of the boundary sensitive, we need to use test problems that encounter uncertainty in parameters and variables. As it is shown in Figure 3.16, as the Type I and Type II uncertainty may impact the boundary of the physical system, the designers should refine the mathematical model accordingly, as the bold dotted red lines in Figure 3.16. By refining the model formulation, the designers expect to identify

the deviation at the optimal solution and *identify satisfying* solutions that are relatively insensitive to the Type I and Type II uncertainty, which is the Type I&II Robust Design.



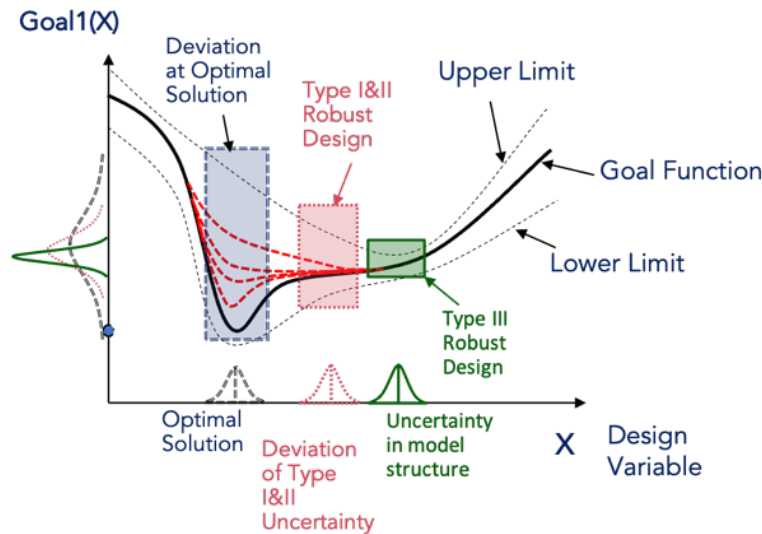
**Figure 3. 16 Test Problems for RDI-II – Refining the model formulation and identifying *satisficing* solutions relatively insensitive to the variation in parameters and decision variables**

Therefore, the features of the test problem(s) for realizing RDI-II are

- Fewer factors to control but a lot of requirements, or in other words, the number of decision variables are fewer than the number of constraints. The coefficient matrix of the model has more rows than columns. What does it mean in the physical world? It means that the variables that the designers can control are limited, whereas there are a lot of requirements that they must meet by controlling the limited variables.
- Multiple goals but lacking knowledge on the ways of combining the goals associated with design preferences. When changing the way of combining the multiple goals to represent different design preferences, the solution may change as well. So, the solutions to the problem are usually sensitive to design preferences. In some cases, designers may not even have the knowledge on the combination ways associated with the design preferences.

- Variation in parameters and decision variables can make the optimal solution infeasible.
- Parameter setting significantly impacts on the design results, but the designers rely on metaheuristics to set parameters and there is no mechanism to evaluate and update the metaheuristics.

To realize the RDIII (robust design Type III) to identify *satisficing* solution space that is relatively insensitive to the variation in mathematical model, we need to apply the proposed methods to test problems with uncertainty in model structure. As it is shown in Figure 3.17, as the Type III uncertainty impact the structure of the model, the designers should learn how the source of the uncertainty disturb the model and make decisions based on the learned information. These decisions lead to a range in the solution space that has relatively narrow variation in model structure solution, which is Type III Robust Design.



**Figure 3. 17 Test Problems for RDI-II – Refining the model formulation and identifying *satisficing* solutions relatively insensitive to the variation in parameters and decision variables**

Therefore, the features of the test problem(s) for realizing RDIII are

- Designers rely on domain expertise to make decisions on the approximation or simplification of the model formulation and solving, but domain-independent knowledge can be acquired through post-solution analyses, applying which, the design can be improved.
- Designers rely on metaheuristics to make rules in one or more stages of the design – formulation, approximation, exploration, and evaluation, but there is no effective mechanism to update metaheuristics.
- Through post-solution analyses, the evaluation of the performance of the metaheuristics can be done and used to improve the metaheuristics updating.

To realize the RDIV (robust design Type IV) to capture emergent properties of the system and incorporate them into the decision model, we need to deal with test problems with emergent properties, learn and leverage critical factors to identify *satisficing* solution space that is relatively insensitive to the uncertainties caused by the actions of managing the first three types of uncertainty.

There are different sources of emergent property. First, some functional relationships, especially cause-and-effect relations, are hidden under the system's appearance, which the designers may ignore in the initial formulation. When gradually recognize those functional relationships during the solution and post-solution analyses, designers may take them as emergent properties and incorporated them into the decision model during the model improvement process. By doing this, the decision model can become more accurate and complete with the model evolution.

Second, as designers attempt to manage Type I, II, and III uncertainty, their actions may bring emergent properties to the system. If they are not aware of and cope with those emergent



properties, the solutions may not be *satisficing* anymore. Therefore, Type IV Robust Design allows designers to capture, quantify, and model the emergent properties. By doing this, the decision model can become more robust to multiple types of uncertainty with the model evolution.

It is expected through managing the emergent properties, designers can identify the solution space that is relatively insensitive to all four types of uncertainty, and this is a dynamic process. How can designers manage emergent properties? By identifying the critical factors that can be controlled as inputs and that they can significantly affect the system's output, identifying the mathematical relations between the critical factors and the output, and leveraging the input to get desired output.

Therefore, the features of the test problem(s) for realizing RDIV are

- Designers' domain knowledge and experience are insufficient to make the model's completeness, accuracy, and fidelity to an acceptable level. This level allows decision support to be somehow correct and useful. For example, designers' knowledge does not support them to design reasonable dimensions and recognize all requirements of the system.
- There are controllable or uncontrollable factors that may or may not impact the output of the system. Designers have no idea of what those factors are, how to control them, and how they affect the output. However, they can learn them using empirical methods such as simulations and scenario planning.
- When designers attempt to manage Type I, or II, or III uncertainty, their actions may cause new uncertainties.

Given the required features of the test problems, in Section 3.4.2, we briefly introduce the test problem – why they are appropriate for this dissertation, how they can be used to testify the proposed methods, what output is expected by dealing with the test problems?

### 3.4.2 Brief Introduction of Each Test Problem

In Table 3.2, it is summarized the five test problems, what robust design types they are used to illustrate, what methods they are used to test, what uncertainties are contained in each problem, and what types of uncertainty they are.

**Table 3. 2 Summary of Test Problems – The robust design type, testified methods, and uncertainties of each test problems. The uncertainties underlined in italic are managed in this dissertation.**

RD Type	RDI-II		RDIII		RDIV
Method	M1: Formulation-Exploration Framework		M2: Adaptive Linear Programming with Parameter Learning (ALPPL)	M3: Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)	M4: Scenario Planning in Agent-Based Modeling
Chapter	Ch 4		Ch 5	Ch 6	Ch 7
Uncertainty / Test Problem	T1.1: Dam network	T1.2: Supply chain	T2: Hot rolling process chain	T3: Thermal system	T4: Promoting second-season farming
Type I	<i>Uncertainty in timing and amount of inflow (precipitation) and outflow demand (user demand)</i>	<i>Uncertainty in demand side (order fluctuation)</i>	<i>Uncertainty in hyper parameter setting (Parameters in approximation algorithm)</i>	<i>Uncertainty in parameter setting in solution algorithm (Starting point of searching)</i>	<i>Uncertainty in price (Price of agriculture products)</i>
Type II	<i>Uncertainty in outflow (water release)</i>	<i>Uncertainty in supply side (productivity fluctuation)</i>	<i>Uncertainty in user preferences</i>		<i>Promotion effort and timing</i>

Type III			<i>Uncertainty in model approximation due to heuristics in approximation</i>	<i>Uncertainty in model approximation (ways of combining multiple goals)</i>	
Type IV				<i>Uncertainty in using domain knowledge to simplify the model (fixing decision variables and selecting design scenarios)</i>	<i>Interventions that change the mathematical relation among promotion and result (developing local market)</i>

RD – robust design

M – method

EVe – empirical verification of the method

T – test problem

***Test Problem 1.1 (T1.1)*** – designing a dam network by controlling the water outflow. The only control factor of this problem is how much water each reservoir should release to its downstream each month, but multiple goals should be satisfied, and uncertainty in precipitation and the priority of the user groups needs to be managed. In this dissertation, the uncertainty in water inflow is managed by exploring and refining the boundary of the system using the formulation-exploration framework. The detailed introduction of T1.1 and the empirical verification of M1 is in Section 4.2.

***Test Problems 1.2 (T1.2)*** – designing a supply chain by positioning the customer order decoupling point (CODP). Unlike the dam-network design problem, this problem has discrete variables, the CODP, so it is a coupled decision problem, which includes two types of decisions, selecting one scenario from several alternatives and compromising multiple goals by determining the value of continuous decision variables. In this problem, uncertainties in the supply side and demand side are managed, multiple goals in different phases of the product life cycle are managed based on evolving preferences. The detailed introduction of T1.2 and the empirical verification of M1 is in Section 4.3.

**Test Problem 1.2 (T1.2)** – improving the parameter setting (setting the reduced move coefficient or RMC for short) of approximation algorithm (the Adaptive Linear Programming algorithm or ALP for short) to make the design (the cooling stage of hot rod process chain) relatively insensitive to the approximation. In the cooling stage of the hot rod process chain problem, the value of the RMC can seriously influence the size and robustness of the solution space, especially the range of the weights that return solutions satisfy different design preferences. Uncertainties caused by using heuristics in parameter settings are managed by using parameter learning. The detailed introduction of T2 and the empirical verification of M2 is in Chapter 5.

**Test Problem 3 (T3)** – improving the understanding of the interrelationship among subsystems of a concurrent engineering design problem, the thermal system, by using unsupervised learning to analyze the correlation or orthogonality among the deviation matrix of the goals in multiple design scenarios. Subsystem awareness and model reformulation regarding the organization of the goals can be helpful in helping designers select the most representative design scenarios and boost the system performance. The detailed introduction of T3 and the empirical verification of M3 is in Chapter 6.

**Test Problem 4 (T4)** – identifying controllable critical factors in a promotion through scenario planning in simulations and leveraging the critical factors to reach promotion goals. In a social system such as a rural community, to promote a new lifestyle based on new technology, even each individual's preferences and behaviors are known, it is difficult to predict the collective behaviors. Simulations using agent-based modeling can help designers understand collective behaviors and what interventions can be done to drive the system behavior towards the direction they desire at the right timing. The detailed introduction of T4 and the empirical verification of M4 is in Chapter 7.

### 3.5 Role of Chapter 3 in this Dissertation

In this chapter, the elements of design improvement through model evolution, which are the possible ways and tasks of answering research questions are discussed in Section 3.1. Theoretical verification of the feasibility of demonstrating the hypotheses by carrying out the tasks are done in Section 3.2. The corresponding methods for finishing the tasks are proposed in Section 3.3. The test problems used to test the hypotheses and methods are briefly introduced in Section 3.4. In summary, this chapter is a foundation for Chapters 4-7, in which different test problems with desired characteristics are used to demonstrate the effectiveness of the proposed methods.

From Chapter 1 to Chapter 3, Quadrant 1 of the Research Questions are addressed; see Figure 3.20. The theoretical structural validity of the research questions is answered by identifying conditions for the design evolution loop under high complexity and uncertainty.

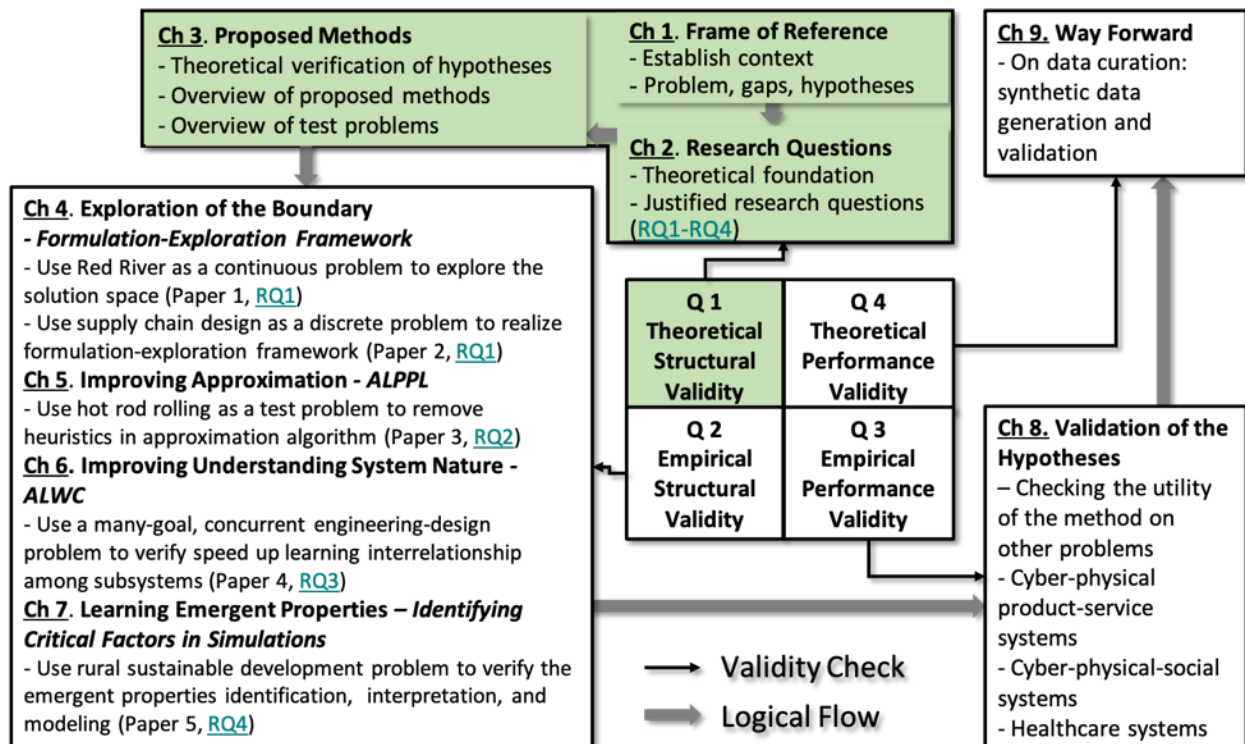


Figure 3. 18 Finishing Theoretical Structural Validity in Chapter 1, 2, and 3

# CHAPTER 4 TYPE I & II ROBUST DESIGN THROUGH FORMULATION-EXPLORATION FRAMEWORK

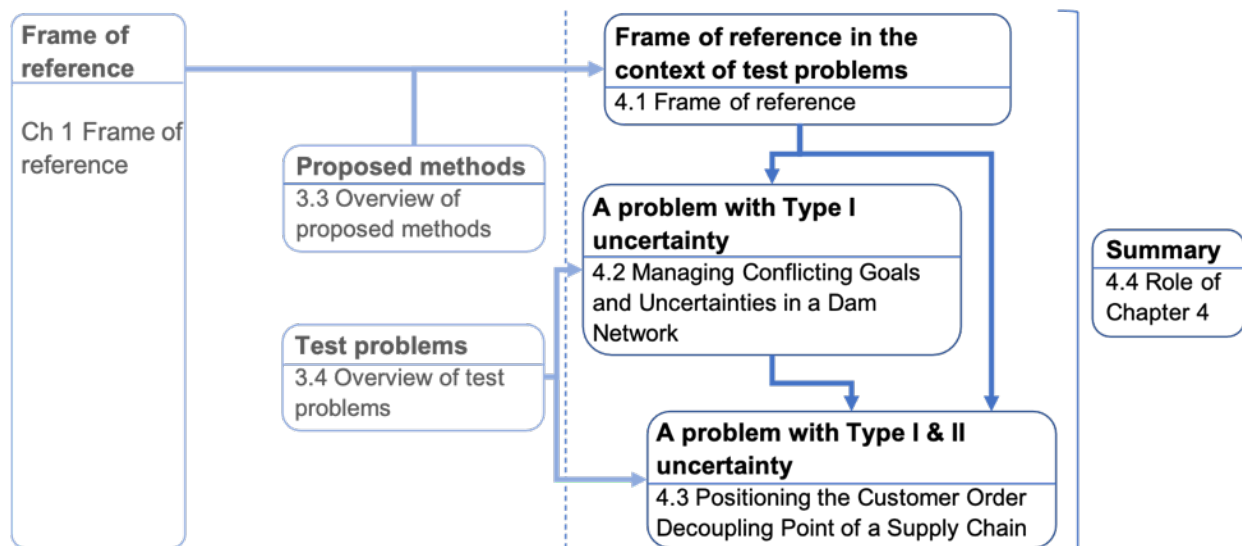
## – EXPLORATION OF THE BOUNDARY OF THE SYSTEM

### *The new knowledge in Chapter 4:*

*A method that allows to identify the sensitive elements of the model and improve the model accordingly – the Three-Step Exploration Method (Figure 4.9).*

*A framework that incorporates the Three-Step Exploration Method that allows to identify the sensitive elements of a mixed-variable, coupled decision model and improve the model accordingly – the Formulation-Exploration Framework (Figure 4.22).*

In Chapter 4, see Figure 4.1: in Section 4.1, the reference is framed in the context of the test problems; in Section 4.2, the dam-network design problem with uncertainty in water inflow is used to empirically verify the Three-Step Exploration method, which is an example of the implementation of the Formulation-Exploration framework; in Section 4.3, the supply chain design problem with discrete variables and uncertainty from supply side and demand side is used to empirically verify the Formulation-Exploration framework; in Section 4.4, summarized the role of Chapter 4.



**Figure 4. 1 Organization of Chapter 4**

The plan of specifying and answering Research Question 1 in the context of the test problems is shown in Table 4.1. In Chapter 4, the Proposed Method 1 (M1), Formulation-Exploration framework, is empirically verified (EVe1) using two test problems, designing a dam network by controlling the water outflow (T1.1) and designing a supply chain by positioning the customer order decoupling point (T1.2). Research Question 1 (RQ1) is specified into the context of the test problems (SQT1) and answered (AQ1) by testifying M1. The empirical validation and theoretical validation are in Chapters 8 and 9.

**Table 4. 1 Plan of Specifying Research Question 1 (RQ1) and Empirically Verifying the Formulation-Exploration Framework (M1)**

Chapter	Ch1	Ch 2	Ch 3	Ch 4-7		Ch 8	Ch 9	
				Ch 4	Ch 5-7			
Actions	RG H	RD RQ SH	TVe M	<p><i>EVe1: use two test problems with uncertainties in parameters and variables to verify SH1 and demonstrate M1.</i></p> <p><i>SQT1: specify RQ1 in the context of the test problems – designing a dam network and designing a supply chain:</i></p> <p><i>Q1.1 - What is needed for a designer to make a satisficing decision with respect to different design preferences, in related to the different requirements in a changing design environment?</i></p> <p><i>Q1.2 - What is needed for a designer to check flexibility of the design in face of errors of the model and variations of the environment?</i></p> <p><i>Q1.3 - What modification can be made to the cDSP to improve feasibility robustness?</i></p> <p><i>Q1.4 - How can designers improve the robustness (insensitivity) of the design of the customer order decoupling point without sacrificing the performance of the supply chain to facilitate mass customization?</i></p>		EVe SQT AQ	CQ EVa	TE
				<p><i>AQ in T1.1 dam-network design:</i></p> <p><i>AQ1.1 - With design scenarios representing different design preferences, designers identify the satisficing area of the weights of the multiple goals and provide their physical meanings.</i></p> <p><i>AQ1.2 - Using inflow scenarios considering different weather and climate conditions to identify the sensitive segments. Exploring the practicality of removing the sensitive segments by modifying the mathematical model and adjusting the physical system.</i></p> <p><i>AQ1.3 - Using the information of the practicality of modifying the model and changing the system, make improvements, including changing parameters' value to add buffers, and reallocating water to different pools to change the physical boundary. Then go through the three steps again. By running such a loop, improve the model to give satisficing solutions that are relatively insensitive to uncertainties.</i></p>	<p><i>AQ in T1.2 Supply Chain design:</i></p> <p><i>AQ1.4 - Using the Formulation-Exploration framework, designers can explore the solution space by using representative design scenarios and explore the potential of boosting the performance of the system by adopting physical means. By doing this in iterations, designers can exploit all means that lead to reaching the most desirable satisficing solution space given the resources on hand.</i></p>			

<b>Nomenclature</b>	RG – give research gaps H – give hypotheses RD – tie to roust design RQ – pose research questions SH – specify hypotheses TVe – theoretically verify hypotheses M – introduce methods EVe – empirically verify hypotheses SQT – specify research questions in the context of test problems AQ – answer research questions CQ – closure the answers to research questions EVa – empirically validate hypotheses TE – theoretically extend the research
---------------------	---

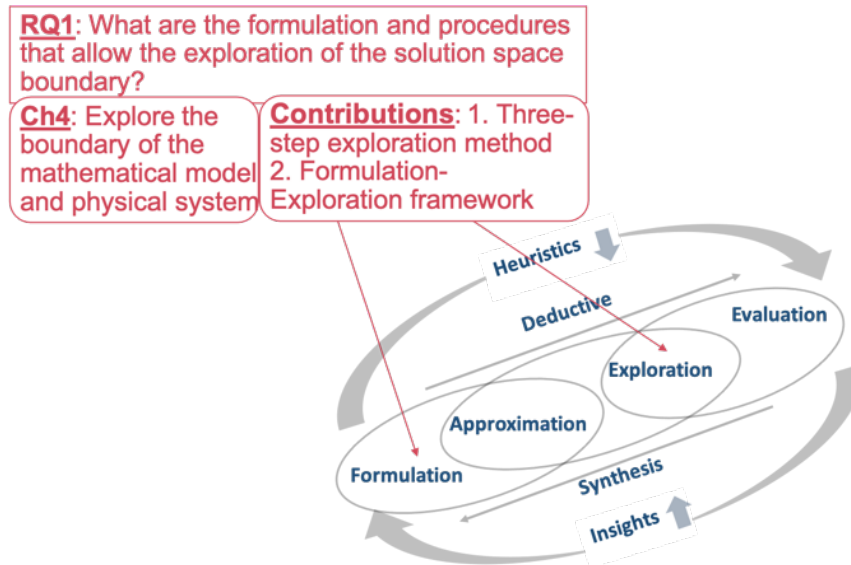
In this chapter, the method and framework of exploration of the solution space boundary are proposed and tested by using two examples in different fields. The Research Question 1 (RQ1) is answered.

***RQ1: What is the method to evolve model boundary?***

In other words, what are the formulation and procedures that allow the exploration of the solution space boundary?

To answer RQ1, formulation and exploration of a design problem should be studied and the interactions between the two procedures are built and expanded. See Figure 4.2.





**Figure 4. 2 Specified Research Question 1 and the Relevant Stages to be Connected in Design Evolution Cycle**

#### 4.1 Frame of References on Satisficing Strategy

George Box, a British mathematician and professor of statistics, wrote that “essentially, all models are wrong but some are useful” (Box and Draper 1987). In keeping with George Box’s observation, the decision maker must be able to work constructively with decision models that are typically incomplete and inaccurate (Simon 1996). The analysis embodied in a decision model does not represent the physical world completely and accurately, making it virtually impossible to predict the future state exactly (Norman 1990). A designer is able to work around this limitation by identifying solutions that are relatively insensitive to inaccuracies embodied in the analyses models; see (Triantaphyllou and Sánchez 1997).

In this chapter, a construct to exercise a decision model, the compromise Decision Support Problem (cDSP) is introduced. Based on the cDSP construct, the exploration of the solution space is presented. The three-step exploration method and the Formulation-Exploration framework are presented to excise the exploration. The method and the framework allow a designer to ascertain

to what extent the solution is insensitive to errors inherent in the modeling of the decision problem (Sabeghi, Smith et al. 2015), and answering to key questions (extended questions based on RQ1) such as:

*Q1.1 - What is needed for a designer to make a satisficing decision with respect to different design preferences, in related to the different requirements in a changing design environment?*

*Q1.2 - What is needed for a designer to check flexibility of the design in face of errors of the model and variations of the environment?*

*Q1.3 - What modification can be made to the cDSP to improve feasibility robustness?*

Moreover, if it is for a particular system, such as a supply chain with a customer order decoupling point, the research question can be further specified into the context of the problem:

*Q1.4 - How can designers improve the robustness (insensitivity) of the design of the customer order decoupling point without sacrificing the performance of the supply chain to facilitate mass customization?*

The compromise DSP is a multi-objective decision model (Mistree, Hughes et al. 1993) which enables a designer to determine values of design variables which satisfy a set of constraints to achieve a set of goals (Chen, Tsui et al. 1994). The objective is to minimize the deviations of different goals from target values using lexicographic minimization (Sabeghi, Shukla et al.).

In a design problem, different stakeholders have different perspectives and need to be accommodated (Aitken, Childerhouse et al.). To model decisions especially in goal programming, the major challenge is in the determination of the weights to assign to the deviations in the objective function (deviation function) (Neely, Sellers et al. 1980). Understanding the inherent

choices and risks within the context of a design lead to justifiable decisions (Smith, Milisavljevic et al. 2015).

In a multidiscipline system, managing conflicting goals is critical to making the system sustainable in the presence of considerable uncertainty. The priorities of multiple goals and availability of the resource vary external environment. Uncertainties such as variations in parameters and consequent variations in constraint boundary can intensify the discrepancies between the designers' desire (the target of the goals) and what designers achieve (the completeness of the goals). To reduce such discrepancies, we seek to *satisfice* the goals, considering typical uncertainties. It is observed that models are incomplete and inaccurate, which calls into question using a single point solution and suggests the need for solutions, which are robust to uncertainties. So, we explore *satisficing* solutions that are relatively insensitive to uncertainties, by incorporating different design preferences, identifying sensitive segments, and improving the design accordingly.

In Chapter 4, RQ1 are addressed using two test problems. The expansion of the literature review and the methods introduction on the two test problems are presented in Section 4.2 and 4.3, respectively.

## **4.2 Managing Conflicting Goals and Uncertainties in a Dam Network**

### ***– Test Problem 1.1 (T1.1): Apply ESS to a Continuous Problem***

#### ***The new knowledge from managing Test Problem 1.1 is***

*A method that allows to identify the sensitive elements of the model and improve the model accordingly – the Three-Step Exploration Method.*

In a multi-reservoir system, ensuring adequate water availability while managing conflicting goals is critical to making the social-ecological system sustainable in the presence of considerable uncertainty. The priorities of multiple user-groups and availability of the water resource vary with

time, weather and other factors. Uncertainties such as variations in precipitation can intensify the discrepancies between water supply and water demand. To reduce such discrepancies, *satisficing* conflicting goals is desired.

It is observed that models are incomplete and inaccurate, which calls into question using a single point solution and suggests the need for solutions which are robust to uncertainties. So, *satisficing* solutions are explored, so that the solutions that are relatively insensitive to uncertainties are identified, associated with different design preferences. Sensitive elements of the model are identified, and the model is improved accordingly. In this section, presented an example of the exploration of the solution space to enhance sustainability in multi-disciplinary systems, when goals conflict, preferences are evolving, and uncertainties add complexity. The proposed three-step method of exploration of the solution space can be applied in mechanical design.

#### **Nomenclature (mainly applied in Section 4.2)**

**DSP** – Decision Support Problem  
**cDSP** – Compromise Decision Support Problem  
**LHS** – Left-hand-side  
**RHS** – Right-hand-side  
**WS** – Weight Scenario  
**IS** – Inflow Scenario  
**LP** – Linear Programming  
**MILP** – Mixed Integer Linear Programming  
**SLP** – Stochastic Linear Programming  
**CCLP** – Chance Constraint Linear Programming  
**NFP** – Network Flow Programming  
**IP** – Interior Point  
**DP** – Dynamic Programming  
**GP** – Goal Programming

#### **Glossary (mainly applied in Section 4.2)**

**Key Nodes** – The nodes in the network that have more than one upstream node that are directly linked to them.

**Ordinary Nodes** – The nodes in the network that have no upstream node or only one upstream node. A node in a network must be either a key node or an ordinary node.

**Boundary of the solution space** – The constraints or bounds of the model that bound the feasible solution space.

**Slack or surplus** – The slack or surplus variable provides information to a designer about how close the current solution is to satisfying a constraint/bound as an equality. This value, on a less-than-or-equal-to ( $\leq$ ) constraint/bound, is referred to as slack, and on a greater-than-or-equal-to ( $\geq$ ) constraint/bound is referred to as surplus. If a constraint/bound exactly satisfies an equality, the slack or surplus value will be zero (LindoOnlineHelp).

**Dual price (shadow price)** – The dual price is the amount that the goal (objective) would be improved (achieved more completely) as the RHS of the constraint/bound is relaxed by one unit (LindoOnlineHelp).

**Active constraints/bounds** – The constraints or bounds that have zero or a very small slack or surplus are defined as active constraints/bounds. In this section, we define “very small” as less than 1% of the larger of the LHS and the RHS of the constraint/bound.

**Improvable constraints/bounds** – The constraints or bounds that have relatively large positive dual prices (shadow prices) are improvable constraints/bounds. In this section, any dual price that is greater than 0.1% of the achieved value of the goals is a relatively large positive dual price.

**Sensitive segments** – There are two types of sensitive segments: active constraints/bounds and improvable constraints/bounds.

**Restricting RHS** – On less-than-or-equal-to ( $\leq$ ) constraints/ bounds, restricting the RHS means decreasing the RHS. On the greater-than-or-equal-to ( $\geq$ ) constraints/bounds, restricting RHS means increasing the RHS.

**Relaxing RHS** – It is the opposite of restricting the RHS. On less-than-or-equal-to ( $\leq$ ) constraints/ bounds, relaxing the RHS means increasing the RHS. On the greater-than-or-equal-to ( $\geq$ ) constraints/bounds, relaxing RHS means decreasing the RHS.

**Satisficing** – “Satisficing is a decision-making strategy or a cognitive heuristic that entails searching through the available alternatives until an acceptability threshold is met” (Byron 1998).

**Acceptability threshold** – A value that is identified by a designer as the acceptable value of a goal and that can be achieved.

**Satisficing solutions** – Satisficing solutions are those solutions for which the acceptability threshold for all goals are met simultaneously.

**Satisficing weight range** – The area in a ternary plot in which the goal weights offer a designer a choice of *satisficing* solutions.

**Physical boundary** – The physical boundary of a system is a boundary that the system can be reached physically. It may be different from the mathematical boundary. To improve the mathematical model and obtain solutions that are relatively insensitive to uncertainties, we “add a buffer” to the physical boundary as the boundary of the mathematical model. Then the boundary of the mathematical model is more restrictive than the physical boundary.

**Insensitive solutions** – Insensitive solutions are solutions away from the physical boundary. Insensitive solutions are relatively insensitive to the uncertainty of the physical boundary.

**Improvement** – In this section, we define model improvement and system improvement as the improvement in insensitivity to the uncertainties.

#### **4.2.1 Problem Statement – Test Problem 1.1: Dam-Network Planning**

The Red River is a tributary of the Atchafalaya River, which is a distributary of the Mississippi River and flows separately into the Gulf of Mexico. We use a part of its dam-network system on the border of Oklahoma and Texas (Figure 4.3). In 2015 the Red River basin experienced a severe

drought followed by flooding, both of these events impact people, planet, and profit. Therefore, managing the supply and sensible distribution of fresh water to support human activity while sustaining vigorous, effective ecosystems is a major ecological challenge (Poff, Brown et al. 2016). We use data from a recent large-scale, comprehensive analysis of the hydrology, societal water usage, and water availability for the Red River by Xue (Xue, Zhang et al. 2015) and McPherson (McPherson 2016) with their coauthors.

*Goals:* We have three user-groups in the basin – people, fish in the reservoirs, and fish in the streams between reservoirs. To meet water demands, there are three goals:

- To reach the target for water storage in reservoirs.
- To meet people’s demand for water – including agricultural and municipal demand.
- To meet the water requirements for the fish in streams.

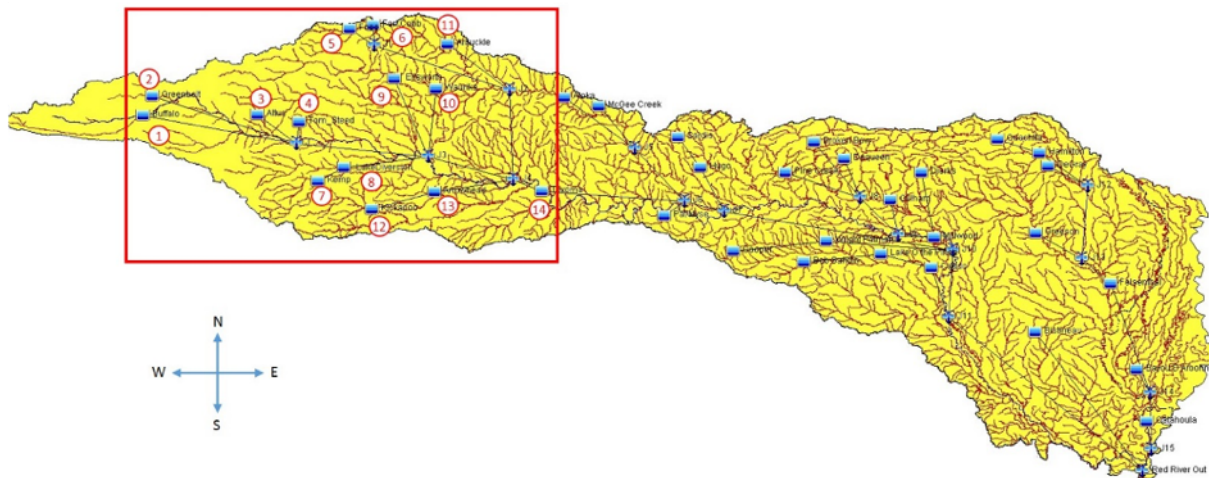
For each goal, we wish to minimize the difference between water supply and water demand. Water volume is measured in cubic feet.

*Problem Size:* we consider a 14-dam network, shown in the rectangle in Figure 4.3 and Figure 4.4. We select these fourteen dams for three reasons.

- 1) Independence. This 14-dam network is a relatively independent sub-network in the 38 dams in the Red River basin, and its interaction with other dams is by managing Dam 14, Texoma.
- 2) Representativeness. As a sub-network of the Red River basin, this 14-dam network is representative, because of its features such as a hierarchical upstream-downstream structure and key nodes which connect with ordinary nodes.

3) Direct interactions with other dams. This 14-dam network is used to release water directly to other dams (the dams not in the rectangle in Figure 4.3) on the Red River basin. This release goes through a single dam – Texoma. The structure of the 14-dam network is shown in Figure 3<sup>12</sup>. The physical features of the network are given in detail in Section 4.2.4.

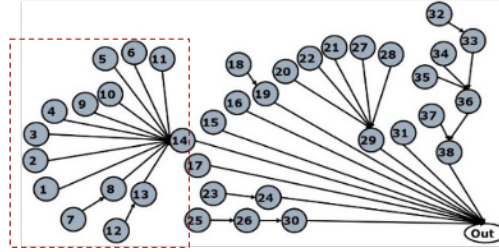
4) Avoiding repeated calculation and analysis. In this section, we focus on the method rather than the results. We use the 14-dam network to prove the utility of the proposed method. The size can be enlarged, and the time scale can be changed – we can apply our method in the 38-dam network, we can manage 12 months rather than 3 months, and we can use a smaller time period such as one week or one day to further reduce the risk, and we can incorporate uncertainties in outflows.



**Figure 4. 3 Dams along the Red River Basin**

---

<sup>12</sup> The position of the nodes and the length of the arches in Figure 4.4 are different from those in Figure 4.3. They are change for the convenience of the visualization. The structure of the network in the two figures are the same.



**Figure 4. 4 The 14-Dam Network**

Here we consider a planning horizon of three months, and the planning time unit is one month.

Each dam corresponds to a reservoir. Each node represents one dam and the reservoir behind it.

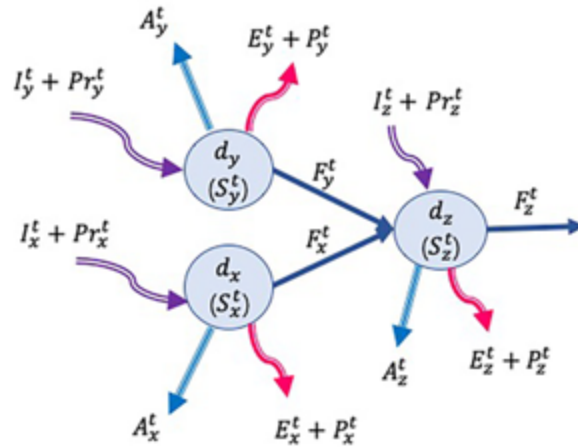
*Variables and Parameters:* The total water inflows of each reservoir have two parts:

- Water which is released from the direct upstream dams of Dam  $d$  ( $\hat{d} \in D^{(d)}$ ) in the previous month ( $t$ ),  $\sum_{\hat{d} \in D^{(d)}} F_{\hat{d}}^t$ . This water conserves fish in the streams and becomes the inflow of Reservoir  $d$ . These inflows are crucial for planning and is controlled by designers, so they are decision variables.
- Natural water received from the outside of Reservoir  $d$  in Month  $t$ , includes tributary inflow ( $I_d^t$ ) and precipitation ( $Pr_d^t$ ),  $I_d^t + Pr_d^t$ . These are crucial to the results but cannot be controlled so they are parameters with uncertainty.
- The total water outflows of each reservoir have three parts:
- Water released for people by Dam  $d$  in Month  $t$ ,  $A_d^t$ , includes for agricultural and municipal use. They are decision variables.
- Water released by Dam  $d$  in Month  $t$ ,  $F_d^t$ , for the fish living between Dam  $d$  and its downstream dams. They are critical and can be controlled so they are decision variables.
- Natural loss of water in Reservoir  $d$  in Month  $t$ , includes evaporation ( $E_d^t$ ) and seepage loss ( $P_d^t$ ),  $E_d^t + P_d^t$ . From the historical data, the natural loss of the 14 dams is only 2%



of the total water storage volume, and the variance of the natural loss in 12 months is only 11% of the variance of the inflows. So, the natural loss and its uncertainty is ignorable compared with the water storage, therefore, they are defined as constant parameters. The water stored in Reservoir d in Month t,  $S_d^t$ , is used to conserve the fish in the reservoir. They are important to the planning and can be controlled so they are decision variables.

In Figure 4.5 a part of the dam-network with three dams is illustrated – Dam x, y, and z, and we show how these decision variables and parameters affect the system.



**Figure 4. 5 A Small Part of the Dam-Network in the Red River Basin**

In addition to the variables and parameters of inflows and outflows, there are target value for each goal.

- The target value of water stored (S) in Reservoir d in Month t,  $ST_d^t$
- The target value of water released for people (A) by Dam d in Month t,  $AT_d^t$
- The target value of water released for fish (F) by Dam d in Month t,  $FT_d^t$

With the parameters and variables, here gives the mathematical form of the goals – Equations 4.1, 4.2 and 4.3. In order to avoid over-supply or under-supply, we use the sum of the squares of the difference between water supply and water demand in each month so that each goal indicates the achievement of water demand for one user-group.

$$\sum_{d \in D} \sum_{t \in T} \left(1 - \frac{S_d^t}{ST_d^t}\right)^2 + v_1^- - v_1^+ = 0 \quad \text{Equation 4. 1}$$

Goal for water stored in Reservoir d in Month t

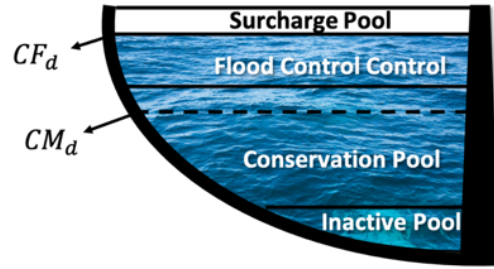
$$\sum_{d \in D} \sum_{t \in T} \left(1 - \frac{A_d^t}{AT_d^t}\right)^2 + v_2^- - v_2^+ = 0 \quad \text{Equation 4. 2}$$

Goal for water released to people by Dam d in Month t

$$\sum_{d \in D} \sum_{t \in T} \left(1 - \frac{F_d^t}{FT_d^t}\right)^2 + v_3^- - v_3^+ = 0 \quad \text{Equation 4. 3}$$

Goal for water released to fish by Dam d in Month t

*Bounds:* As it is shown in Figure 4.6, for each reservoir, there are several pools for different functionalities. The water level in the inactive pool never changes; the conservation pool is to conserve fish. Within the conservation pool, a certain volume is necessary to contain enough microorganisms and nutrients for the fish to live and reproduce healthily. The flood control pool is reserved for flood runoff and must be evacuated immediately to keep the space in readiness for the next flood. On the top of the reservoir, there is a surcharge pool. We do not plan to use the surcharge pool because it is the last backup space.



**Figure 4. 6 The Pools of a Reservoir**

For Reservoir d, there is a lower bound of water storage volume,  $CM_d$ , to guarantee that the amount of water in the conservation pool is acceptable and does not cause system failures. Its upper bound,  $CF_d$ , is the capacity including inactive pool, conservation pool, and flood control pool.

The lower bound of water release volume, both for fish ( $F_d^t$ ) and for people ( $A_d^t$ ), is zero, since the water outflows subtract backflows are always positive. We do not have an upper bound of  $F_d^t$  and  $A_d^t$ . Goal 2 and Goal 3 are set to restrain the over-achievement for water released for people and fish.

In summary, the bounds are the capacity of reservoirs, and the upper/lower limit of water release volume and water storage volume – Equation 4.4-4.7.

$$S_d^t \leq CF_d \quad \text{Equation 4. 4}$$

Upper bound of the water released by Dam d in Month t

$$S_d^t \geq CM_d \quad \text{Equation 4. 5}$$

Lower bound of the water released by Dam d in Month t

$$F_d^t \geq 0 \quad \text{Equation 4. 6}$$

Lower bound of the water released to fish by Dam d in Month t

$$A_d^t \geq 0$$

Equation 4. 7

Lower bound of the water released to people by Dam d in Month t

*Constraints:* to the water storage of Reservoir d at the beginning of Month t,  $S_d^t$ , the total water inflow is added,  $\sum_{\hat{a} \in D(d)} F_{\hat{a}}^t + I_d^t + Pr_d^t$ , and the total water outflow is subtracted,  $-F_d^t - A_d^t - E_d^t - P_d^t$ , so the reservoir's water storage at the beginning of the next month,  $S_d^{t+1}$ , is obtained. (Equation 4.8 and Figure 4.7.)

$$S_d^t + \sum_{\forall \hat{a} \in UD_d} F_{\hat{a}}^t + I_d^t + Pr_d^t - F_d^t - A_d^t - E_d^t - P_d^t = S_d^{t+1}, \text{ where } t = 1, 2, 3$$

Equation 4. 8

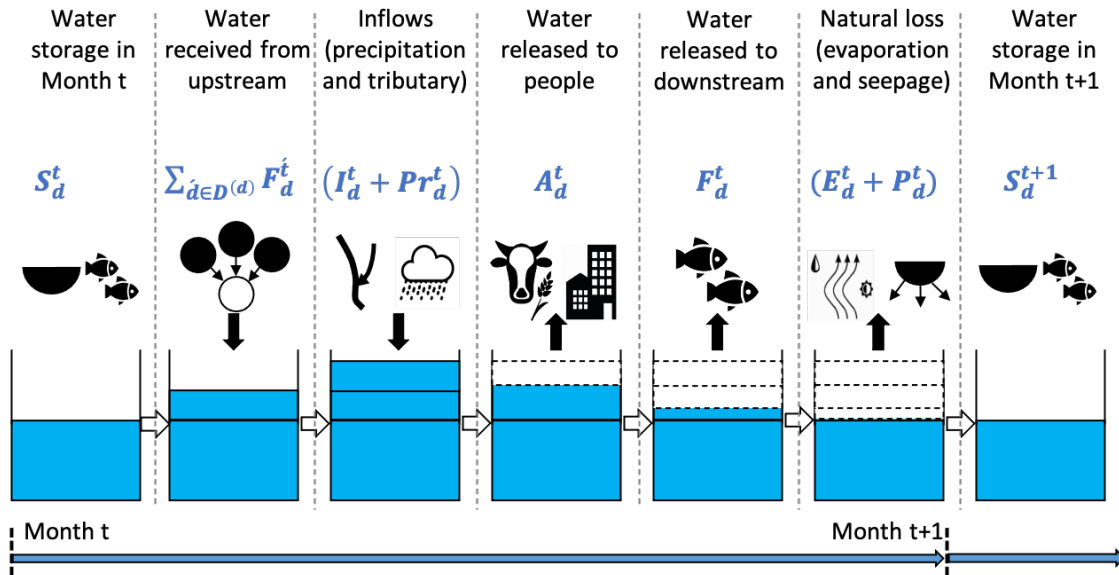


Figure 4. 7 Illustration of the Equality Constraints for Dam (Reservoir) d

In this model, we have water storage volume as a goal, meanwhile we have upper bounds and lower bounds of water storage volume as bounds. The meaning of the goal and bounds are different. The water stored in each reservoir ensures the healthiness of the conservation pool, to conserve the fish. The upper bound and lower bound of water storage are to guarantee operable conditions – making sure the reservoirs can be run smoothly with low risk of system breakdown. The water

demand of the fish in reservoirs, however, is more specific. In other words, the bounds of the water storage in a reservoir allow the operators to have a relatively larger range for operations, whereas the target for water volume in the reservoir is an ideal value that boosts the environmental condition.

In this section, only the quantity of the water is considered without considering the water quality. The flows for each month are planned and the daily differences are ignored.

*Uncertainty:* Based on our data, we simultaneously manage two uncertainties in water inflows – the variations in precipitation and tributary inflow under present-day conditions. The uncertainties are modeled by considering different inflow scenarios (ISs) of precipitation and tributary inflow. Each IS represents a typical weather and climate condition, from extreme drought to flood, and uneven case as drought followed by flood. We use ISs instead of stochastic variables to incorporate the uncertainty to avoid making assumptions about the uncertainty distribution. This is explained further in Section 4.2.3.

#### ***4.2.2 Critical Review of The Literature on Dam-Network Water Resource Management***

Challenges such as high dimensionality and computational complexity are encountered by dam-network designers (Reddy and Kumar 2007) – multiple user-groups compete for a limited resource, and designers must deal with complicated variables. Those variables include precipitation, tributary inflow, dam storage, irrigation, and municipal water demands (Rani and Moreira 2010), etc. They are complicated because they often bring uncertainties into the system. Once a dam-network is planned, operational plans and evaluations should make the system's performance meet the water demand of all user-groups as closely as possible.

To make the reservoirs serve people's goals, scholars have tried many methods to plan the dams. Popular methods of dam water planning include Linear Programming (LP), Mixed Integer Linear

Programming (MILP), Stochastic Linear Programming (SLP), Chance Constrained Linear Programming (CCLP), Network Flow Programming (NFP), Interior Point (IP), Nonlinear Programming (NLP), Dynamic Programming (DP), and Goal Programming (GP) (Wurbs). However, there are limitations with each of these methods. See Table 4.2.

**Table 4. 2. Gaps and limitations of the Methods in the Literature**

Methods in literature	Manage nonlinear/nonconvex features	Manage uncertainties in inflows/outflows	Manageable computational complexity	Relatively stable performances as size increases	Relatively stable performances as dimensionality changes	Gaps and Limitations
Deterministic Linear Program-ming (LP)			*	*	*	Cannot manage nonlinearity, nonconvexity, discontinuity, or variations in parameters.
Mixed Integer Linear Programming (MILP)	*					Computational complexity increases exponentially as the number of integer variables increases.
Stochastic Linear Programming (SLP)	*	*	*	*	*	The assumptions of the distribution of stochastic variables may be incorrect.
Chance Constrained Linear Programming (CCLP)	*	*	*	*	*	Can be used to decrease the frequency of system failures but cannot manage the severity of each system failure.
Network Flow Programming (NFP)			*	*		Cannot be used to evaluate the structure of the network and output improvement suggestion.
Interior Point Methods (IP)	*			*	*	IP methods are efficient only when the problem is a large-scale one and are relatively hard to be implemented.
Nonlinear Programming (NLP)	*	*				Not computational efficient for engineering design purposes and may trap the designer in a local optimum.
Dynamic Programming (DP)			*			Curse of dimensionality – adding extra dimensions in Euclidean space causes exponential increases in calculation volume.
Goal Programming (GP)			*	*	*	<i>Preemptive GP</i> : An error in the choice of a primary goal can prevent a possible large improvement in a secondary goal, yet setting hierarchy of the goals are based on the designers' domain knowledge and intuition rather than general quantified methods; <i>Weighted GP</i> : There is difficulty in finding the appropriate values of the goal weights and evaluating their rationality.

Using deterministic LP, Crawley and Dandy (Crawley and Dandy 1993) determine optimal operating policies using an LP model, Loucks (Loucks 2000) sizes reservoir capacities, Dahe, and Srivastava (Dahe and Srivastava 2002) apply yield models to design an eight-reservoir system,

Needham and coauthors (Needham, Watkins Jr et al. 2000) offer methods for flood control, and Vedula and coauthors (Vedula, Mujumdar et al. 2005) make an optimal conjunctive use policy for irrigation in a reservoir–canal–aquifer system. Nonlinear features or uncertainties in the system cannot be captured and managed in LP – for example, uncertainties in water inflow and outflow.

To manage the uncertainties, such as the variations in parameters, Stochastic Linear Programming (SLP) and Chance Constrained Linear Programming (CCLP) are widely used. Loucks and coauthors (Loucks, Stedinger et al. 1981) use an SLP to get optimal steady-state probabilities for dam outflows and storage assuming the inflows follow a single Markov chain. Sreenivasan and Vedula (Sreenivasan and Vedula 1996) apply a CCLP to determine the optimal hydropower production while satisfying irrigation demands at a specified level of reliability. Van Ackooij and coauthors (van Ackooij, Henrion et al. 2014) present a cascaded reservoir optimization problem with uncertainty of inflows in a joint CCLP setting, and present an iterative algorithm for solving similarly structured problems that require a Slater point and the computation of gradients. SLP has been used with assumptions about the distribution of stochastic variables, which may cause errors. Another problem with SLP and CCLP is that they only guarantee a small probability of the system failure but cannot ensure that for each failure the severity is within a manageable level.

Since the reservoirs work as a network, Network Flow Programming (NFP) is used to deal with multi-reservoir system planning and is considered to be a computationally efficient form of LP. Ford and Fulkerson (Ford and Fulkerson 1962) treat reservoir systems as general configurations of capacitated networks, and they maximize the flow while minimizing the cost. Kuczera and Diment (Kuczera and Diment 1988) discuss the principles of formulating network LPs, and apply these principles to develop a simulation model, WASP (Water Assignment Simulation Package). Hsu and Cheng (Hsu and Cheng 2002) apply a generalized NFP model for long-term supply-

demand analysis for basin-wide water resources planning. NFP allows us to manage the amount and timing of the flow in the network as a whole, but one of its limitations is that we cannot determine the potential for improvement of the network structure.

Interior Point (IP) Methods are efficient in solving large problems. Ponnambalam and coauthors (Ponnambalam, Vannelli et al. 1989) introduce Karmarkar's interior point LP (IPLP) approach to reservoir operation, showing that the IPLP algorithm is capable of solving large multi-reservoir operation problems faster than the simplex method. Seifi and Hipel (Seifi and Hipel 2001) apply an improved IP method to multi-reservoir operation planning and show how exploiting the problem structure enhances algorithm performance. Mousavi and coauthors (Mousavi, Moghaddam et al. 2004) use a primal-dual IP algorithm to resolve dimensionality of a multi-reservoir, multi-functional system and demonstrate the computational efficiency of the proposed method using historical data. However, IP only works better than the Simplex algorithm for large-scale problems, and many IP algorithms are relatively hard to implement.

Managing the multiple functions of reservoir systems is computationally difficult due to the nonlinearity in the complex relationships among physical and hydrological variables. This problem is often solved by approximation (linearization) or successive applications of LP, dynamic programming, or algorithms such as sequential quadratic programming (SQP) and generalized reduced gradients (GRG). Zhou and coauthors (Zhou, Zhang et al. 2007) use the combination of entropy and fuzzy optimization to improve reservoir operation. Teegavarapu and Simonovic (Teegavarapu and Simonovic 2000) use a mixed integer NLP formulation with binary variables to study daily hydropower operation of four cascading reservoirs. Barros and coauthors (Barros, Tsai et al. 2003) demonstrate that NLP is particularly fit for setting up guidelines for real-time operations using inflow prediction with frequent updating. A well-known drawback of NLP is its



computational complexity. Although the NLP are with good fidelity and suited for real-time operations, they are not computationally efficient for engineering design purposes. In addition, using NLP may trap a designer in a local optimum without sufficiently exploring the design space.

To obtain acceptable computational complexity while managing multi-period plans, Dynamic Programming (DP) is introduced (Rani and Moreira 2010). Yakowitz (Yakowitz 1982) reviews DP models and concludes that computational considerations impose a severe limitation on the scale of DP problems. Nandalal and Bogardi (Nandalal and Bogardi 2007) discuss the applicability and limits of DP methods, specifically for reservoir operation problems. The well-known “curse of dimensionality”, however, is the main drawback of DP (Bellman and Dreyfus 1962), because adding extra dimensions in Euclidean space may cause exponential increases in volume (Bellman 1957).

Goal Programming (GP) is an efficient method for managing multiple conflicting goals (Ignizio 1982). GP is a multi-objective optimization method that allows the flexible expression of policy constraints as objectives. Clayton and coauthors (Clayton, Weber et al. 1982) develop an approach of goal programming using a modified pattern search routine. Loganathan and Bhattacharya (Loganathan and Bhattacharya 1990) apply five GP schemes (preemptive, weighted, min-max, fuzzy, and interval) minimizing deviations from a set of preferred target storage and flow values, and find efficient alternative optima. Eschenbach and coauthors (Eschenbach, Magee et al. 2001) use preemptive GP and combined detailed system representations, policy expression flexibility, and computational speed for routine daily scheduling of large complex multi-objective reservoir systems. Changchit and Terrell (Changchit and Terrell 1993) present an application of chance-constrained GP (CCGP) to a system of multipurpose reservoirs for planning rather than real-time operation. The problem with GP is that designers cannot separate the goals into “rigid” goals and

“soft” goals to manage them in different ways. Moreover, the problem with preemptive GP is that even a tiny difficulty of a primary goal may block a large improvement in a secondary goal. The challenge with weighted GP is determining the appropriate weights to represent the goals priority.

The limitations of these methods are summarized in Table 4.1. Besides, there is another difficulty. All of these methods are used to find optimal solutions, which are boundary solutions. As models are incomplete and inaccurate, boundary solutions are sensitive to uncertainties in the constraints and bounds. According to George Box (Box and Draper 1987), “all models are wrong but some are useful.” In other words, models are typically incomplete, inaccurate and of different fidelities, hence the use of a multi-objective optimization model to obtain a single optimal solution is questionable. Especially in a dam-network with a variety of uncertainties, such as variations in precipitation, uncertain tributary inflow, and climate change.

Therefore, instead of searching for optimal solutions, we turn to searching for *satisficing* solutions, as described by Herbert A. Simon (Simon and Kadane 1975). *Satisficing* solutions are “good enough” solutions. A designer can work around this research gap by providing insensitive solutions – the solutions which are relatively insensitive to inaccuracies embodied in the models (Triantaphyllou and Sánchez 1997) and uncertainties that cannot be well captured in the models. The method descriptions are given in Section 4.2.3.

#### ***4.2.3 Proposed Methods – The Three-Step Exploration Method***

As the total water resource is limited, the designer may not meet all user-groups’ water demands at all times, hence we work to minimize the discrepancies between water supply and water demand for each user-group. We use a method which is a hybrid of mathematical programming and GP – the compromise Decision Support Problem (cDSP) because we can manage “rigid” goals (such as

the physical capacity of the reservoirs) as constraints and manage “soft” goals (such as the water demand of different user-groups) as goals. Mistree and coauthors (Mistree 1993) develop the cDSP. It has the following advantages.

1. Generic. Any optimization model has an equivalent representation in the cDSP.
2. Appropriate accuracy and manageable computational complexity. Not only does using the cDSP offer a method for an alternative representation, but it also provides a representation for effectively capturing the nature of real problems by incorporating nonlinear constraints and goals and mixed variables. After approximating the problem as a linear one, the accuracy of the approximated problem is maintained as a “good enough” level, and its computational complexity is reduced to a manageable level (Mistree, Hughes et al. 1981).
3. Providing *satisficing* solutions. Using the cDSP construct, designers can explore the solution space to *satisfice* different design preferences, manage multiple types of uncertainties in the system, and improve the model and the physical system.

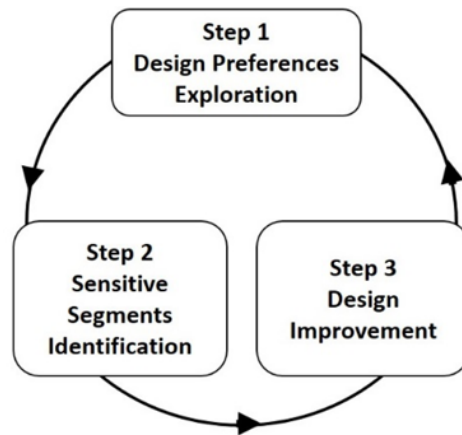
By exploring the solution space, a designer can identify a space containing *satisficing* solutions that are relatively insensitive to 1) errors that are anchored in incomplete and possibly inaccurate information embodied in the analysis models used to define the behavior of the dam-network, and 2) unpredictable uncertainties in parameters. Applying the method may allow a designer to answer three key questions in the context of dam-network planning:

*Q1.1-1. How can a designer manage multiple conflicting water resource goals with reasonable priorities under different precipitation scenarios?*

*Q1.2-1. How can a designer manage uncertainties in the inflows (precipitation and inflows from upstream) by making water flow plans that are relatively insensitive to these uncertainties?*

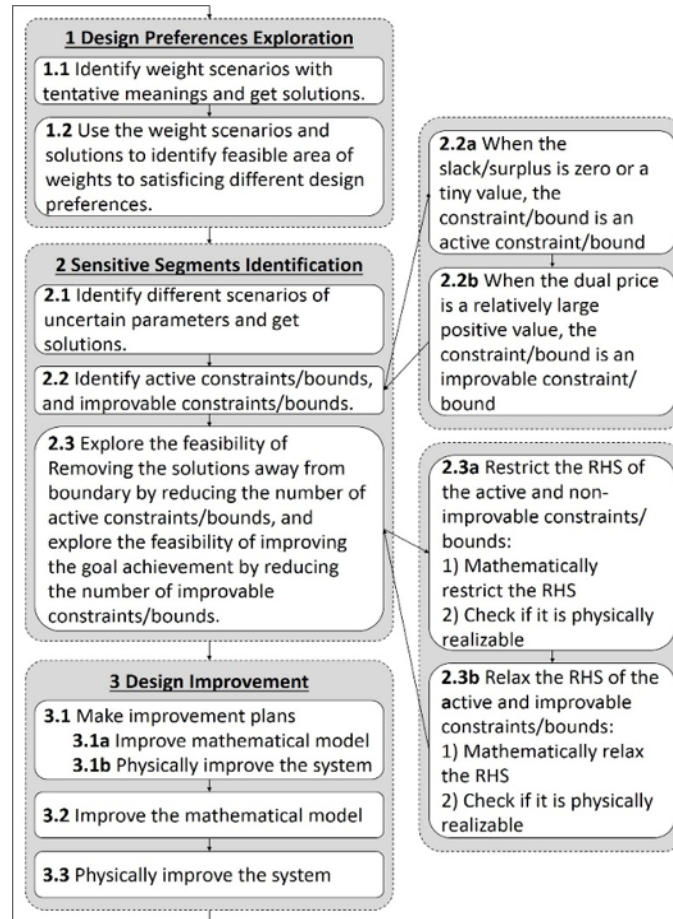
*Q1.3-1. How can a designer improve the mathematical model and give recommendations on the physical system improvement so that the system can be relatively insensitive to uncertainties?*

To answer the three key questions, we hypothesize that there are three steps to be accomplished – design preference exploration, sensitive segment identification, and design improvement, which are formed in a loop (Figure 4.8).



**Figure 4. 8 Three Steps for the Exploration of the Solution Space**

The detailed procedural steps for exploring the solution space are given in (Fok and Chopra 1986), with extensions and applications in (Sabeghi, Shukla et al. 2016). In this section, the method is extended (Figure 4.9).



**Figure 4. 9 Method for Exploration of the Solution Space**

**Step 1 (Figure 4.9):** Design preference exploration. In this step, the precipitation and tributary forecasts are treated as deterministic values, to obtain an overview of the feasible space for the goal weights ignoring any uncertainty.

**Step 1.1:** A group of weight scenarios (WSs) to represent different goal priorities are used to obtain solutions.

**Step 1.2:** Using ternary plots to identify the range of weights that *satisfice* different design preferences.

**Step 2 (Figure 4.9):** Sensitive segment identification. There are two types of sensitive segments.

***Active constraints/bounds:*** constraints or bounds with zero or very small slack/surplus. We define “very small slack/surplus” as “less than 1% of the larger of the RHS and LHS of a constraint/bound”.

***Improvable constraints/bounds:*** constraints and bounds with relatively large positive dual prices. We define a dual price greater than “0.1% of the achieved value of the goals” as a relatively large positive dual price.

Active constraints and improvable constraints are sensitive segments because:

The solutions are always on the boundary formed by active constraints/bounds, so they are relatively sensitive to the changes that take place to the active constraints/bounds. When their RHS values of change, we may lose the solutions. Therefore, we explore the practicality of moving the solutions away from the boundary, to make them relatively insensitive to uncertainties.

By relaxing the improvable constraints/bounds, the goals can be achieved more completely. Hence, we explore the practicality of relaxing the improvable constraints/bounds.

In this step, we take into account the uncertainties by using different parameter scenarios (inflow scenarios - ISs) that represent different weather and climate conditions. By analyzing the practicality of moving the boundary solutions and improving goal achievement, we suggest model modifications for system improvement.

**Step 2.1:** Take into account possible weather and climate conditions by using representative inflow scenarios (ISs) and obtain solutions of different inflow scenarios.

**Step 2.2:** With the solutions, identify the sensitive.

**Step 2.3:** Explore the practicality of bringing the solutions away from the boundary by reducing the number of active constraints/bounds and exploring the practicality of improving the goal achievement by reducing the number of improvable constraints/bounds.

**Step 3 (Figure 4.9):** Design improvement. With the outcomes from step 2.2 and 2.3, we improve the model mathematically or improve the system physically.

We improve the design either by modifying the mathematical model or by adjusting the physical system.

**Step 3.1:** Improvements are planned based on steps 2.2 and 2.3.

***Determine which improvement actions we shall apply.***

All the sensitive segments can be divided into two categories – they are either active and improvable constraints/bounds, or active and non-improvable constraints. In other words, all improvable constraints/bounds are active constraints/bounds, but not all active constraints/bounds are improvable constraints/bounds.

There are two characteristics of active and improvable constraints/bounds: 1) they are too restrictive, thus if they are relaxed the goals will be achieved more completely, and 2) the solutions are on the boundary, so they are sensitive to uncertainties. We need to deal with each type separately. For the first type, the system is adjusted physically (Step 3.1b) and for the second type the mathematical model is refined (Step 3.1a).

The active and non-improvable constraints/bounds are of the second type, so we modify the mathematical model (Step 3.1a).

**Step 3.2:** The mathematical model is improved by adjusting the relevant parameters.

### ***Options for modifying the mathematical model:***

Bring the solution away from the physical boundary by creating a more conservative mathematical boundary – since the solution is always on the boundary, we “add a buffer” by creating a mathematical boundary that is more conservative than the physical boundary, then the solution will be on the mathematical boundary while staying away from the physical boundary.

Manage the uncertainty by obtaining more accurate assessment of the parameters – the parameters that bring uncertainties make the solution infeasible or close to the boundary, so through parameters assessment, we can avoid some of such uncertainties.

**Step 3.3:** The physical system is improved, and the mathematical model is improved correspondingly.

We use the results of step 2.3 here – if the improvable constraints/bounds can be relaxed by making physical change to the system, then we make the physical change and then the corresponding mathematical change so that the goals can be achieved more completely; if we cannot do any physical change then we accept the current physical boundary.

With the proposed method, we formulate the model in Section 4.2.4.

#### ***4.2.4 Formulation of Compromise DSP (cDSP)***

In this section we demonstrate the use of Step 1.1 (Figure 4.9). Based on the network structure in Figure 4.4, we summarize the information of each dam in Table 4.3. For example, for Dam 14, Texoma, 11 upstream dams (1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 13) release water to it.



**Table 4. 3 Features of 14 Dams in Red River Basin**

Number	Dams	Set of Upstream Dams ( $UD_d$ )
1	Buffalo	
2	Greenbelt	
3	Altus	
4	Tom Steed	
5	Foss	
6	Fort Cobb	
7	Kemp	
8	Lake Diversion	7
9	Ellsworth	
10	Waurika	
11	Arbuckle	
12	Kickapoo	
13	Arrowhead	12
14	Texoma	1,2,3,4,5,6,8,9,10,11,13
<b>Out</b>	To the other part of the network	14

In this step, we use a deterministic value of precipitation based on historical data.

The cDSP for the 14-Dam network is as follows<sup>13</sup>.

**Given**

System parameters

$D = \{d\} = \{1,2, \dots, 14\}$  //The set of 14 dams (reservoirs)

$T = \{t\} = \{1,2,3\}$  //Planning period – three months

$UD_d = \{\hat{d}\}$  //The set of upstream dams (Table 2).

$ST_d^t$  //Target of water storage volume for Reservoir d at the beginning of Month t

$FT_d^t$  //Target of water release volume to downstream fish from Dam d in Month t

$AT_d^t$  //Target of water release volume to people from Dam d in Month t

<sup>13</sup> In this problem, to separate the Dam “d” from the deviation variable “d” and “d<sup>+</sup>”, the deviation variables are represented by “v<sup>-</sup>” and “v<sup>+</sup>”. For the other problems in this dissertation, the deviation variables are all represented using “d” and “d<sup>+</sup>”.

$(E_d^t + P_d^t)$  //Natural loss (evaporation and seepage) of Reservoir d in Month t  
 $CF_d^t$  //Flood capacity of Reservoir d  
 $CM_d^t$  //Minimum storage volume of Reservoir d  
 $(I_d^t + Pr_d^t)$  //Anticipated tributary inflow and precipitation for Reservoir d  
 $w_i$ , where  $i = 1,2,3$  //Weight of Goal i

### **Find**

System variables

$S_d^t$  //The volume of water stored in Reservoir d at the beginning of Month t

$A_d^t$  //The volume of water released from Dam d to people in Month t

$F_d^t$  //The volume of water released from Dam d to downstream fish in Month t

Deviation variables

$v_i^+$ ,  $v_i^-$ , where  $i = 1,2,3$  //Over-achievement and underachievement of Goal i

### **Satisfy**

System constraint

$S_d^t + \sum_{\forall d \in UD_d} F_d^t + I_d^t + Pr_d^t - F_d^t - A_d^t - E_d^t - P_d^t = S_d^{t+1}$ , where  $t = 1,2,3$

System goals

$$\sum_{d \in D} \sum_{t \in T} (1 - \frac{S_d^t}{ST_d^t})^2 + \sum_{d \in D} (1 - \frac{S_d^4}{ST_d^4})^2 + v_1^- - v_1^+ = 0$$

//Goal 1 – Reservoir: reach the target of water storage in each reservoir in the beginning of each month, and at the end of the last month.

$$\sum_{d \in D} \sum_{t \in T} (1 - \frac{A_d^t}{AT_d^t})^2 + v_2^- - v_2^+ = 0$$

//Goal 2 – People: reach the target of water released to people by each dam in each month.

$$\sum_{d \in D} \sum_{t \in T} (1 - \frac{F_d^t}{FT_d^t})^2 + v_3^- - v_3^+ = 0$$

//Goal 3 – Fish: reach the target of water released to fish in streams by each dam in each month.

Bounds

$$S_d^t \leq CF_d$$

$$S_d^t \geq CM_d$$

$$F_d^t \geq 0$$

$$A_d^t \geq 0 // \text{Bounds of system variables}$$

$$v_i^- \geq 0,$$

$$v_i^+ \geq 0,$$

$$v_i^- \cdot v_i^+ = 0$$

where  $i = 1,2,3$  //Bounds of deviation variables

### **Minimize**

The deviation function

$$z = \sum_{i=1}^3 w_i \cdot v_i^-, \text{ where } 0 \leq w_i \leq 1, \text{ and } \sum_{i=1}^3 w_i = 1$$

//The weighted sum of deviation variables

*Explanation of the Goal Formulation:* To scale the three goals, we use the variable divided by its target value as the achievement rate of each variable ( $\frac{S_d^t}{ST_d^t}$ ,  $\frac{A_d^t}{AT_d^t}$  and  $\frac{F_d^t}{FT_d^t}$ ), and then use “One subtracts the achievement rate” as the difference between supply and demand. In this way we scale the water demand of different user-groups. We calculate how much percentage we do not meet a user-group’s demand, instead of how much water in quantity (cubic feet). According to our data, the average target for water storage ( $ST_d^t$ ) is around 390000 cubic feet, whereas the average target for water release for people ( $AT_d^t$ ) and water release for fish ( $FT_d^t$ ) are around 30000 and 51000 cubic feet. If we do not scale them, the water storage goal will always dominate the other two goals.

*The Utility and Meaning of the Weights:* Sub-networks along the Red River basin serve different conditions and circumstances – different populations, crops, wild fish species, and water storage flexibilities, thus the priority of the goals may vary from dam to dam. Even for a single area, the priority for meeting different users’ demands may vary with seasons. To empower decision makers to have the flexibility of managing goals associated with different design preferences, we assign different weight scenarios (WSs) to the goals and make rules for them to follow. This process is defined as Design Preference Exploration (Figure 4.9, Step 1).

#### ***4.2.5 Water Resource Planning Results and Discussion***

**Step 1.1 Identify Weight Scenarios and Get Solutions:** By assigning equal weights to the three goals and solving the problem using Lingo 17.0, we obtain the results (Table 4.4). In this scenario, the three weights are 0.333, 0.333 and 0.333 respectively. The value under each goal is the value of deviation variable, the sum of the square of the discrepancies between supply and demand, hence we prefer small values.

**Table 4. 4 Results for Equal Weights on Preferences**

<b>W1</b>	<b>W2</b>	<b>W3</b>	<b>Goal 1 (<math>v_1^+</math>)</b>	<b>Goal 2 (<math>v_2^+</math>)</b>	<b>Goal 3 (<math>v_3^+</math>)</b>
0.333	0.333	0.333	0.49	0.53	0.31

We use more weight scenarios (WSs) to generate solutions and identify *satisficing* solutions. The WSs represent different considerations of the priority of the goals.

### **Design Preference Exploration**

#### **Step 1.2 Use Weight Scenarios and Solutions to Identify Feasible Area of Weights that Satisfice**

**Different design Preferences.** As we have three goals, we use ternary plots to visualize the achievement of each goal corresponding to different weight scenarios (WSs) and identify the meaning of each WS. By using ternary one can expand discrete results to the whole design space (weight space) and visualize the *satisficing* space.

We visualize the location of 8 WSs in a ternary (Figure 4.10) and give their tentative physical meanings (Table 4.5). We use the three axes (sides of the triangle) to represent the weights of the three goals, so each point in the ternary is a unique WS. The sum of the coordinates along the three axes of any point in the ternary is always one. To make WSs representative and cover a variety of design preferences, we work out the problem with 34 WSs and obtain results in Appendix A.

*Data Normalization:* we normalize the deviations of each goal within the range of [0, 1] for visualization. For each goal we convert the minimum deviation value to 0, and the maximum deviation value to 1, and other values between 0 and 1.

*Results Visualization:* we plot the results of each goal using color contours in a ternary (Figure 4.11, 4.12 and 4.13 for Goal 1, Goal 2 and Goal 3). The WS with the smallest deviation value, which is what we are seeking, is in dark blue, whereas the largest value is in dark red, and every

value in between has a color between dark blue and dark red. We can identify the value of each point by looking up the color bar on the right.

*Identifying the Feasible Area of the Weight for Each Goal:* For each goal, the aspiration may not be met, but with expertise in water resource management<sup>14</sup>, an acceptable value of the achievement of the goal is identified as a threshold, which is converted to normalized deviation values of 0.29, 0.32 and 0.3 for reservoir, people and fish, respectively. For each goal, the weights giving results that are less than or equal to the threshold are feasible weights, and the area containing all feasible weights is the feasible area of the weights. In each ternary plot, we illustrate the threshold with a line (the more WSs we use, the smoother the line is), and arrows show the region of feasibility for the weights.

*Identifying the Satisficing Area of Weights:* When we overlap the feasible area for the three weights, we obtain the superimposed area as the *satisficing* area of weights (shaded area in Figure 4.14).

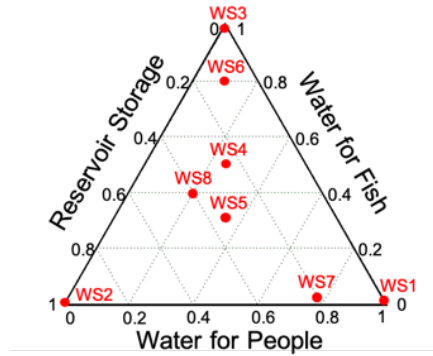
---

<sup>14</sup> The data and relevant knowledge are from

Xue et al. Xue, X., K. Zhang, Y. Hong, J. J. Gourley, W. Kellogg, R. A. McPherson, Z. Wan and B. N. Austin (2015). "New multisite cascading calibration approach for hydrological models: Case study in the red river basin using the VIC model." Journal of Hydrologic Engineering **21**(2): 05015019.

and

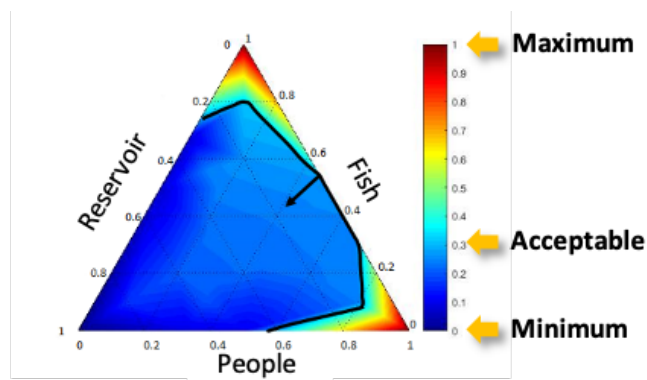
McPherson et al. McPherson, R., et al. (2016). "Impacts of climate change on flows in the Red River Basin." Final report to the South Central Climate Science Center.



**Figure 4. 10 Visualization of the Eight WSs in the Ternary Plot**

**Table 4. 5 Physical Meaning of the Eight WSs – Type II Uncertainty**

WS	W1	W2	W3	Tentative physical meaning
1	0	1	0	People is the only important goal
2	1	0	0	Reservoir is the only important goal
3	0	0	1	Stream fish is the only important goal
4	0.25	0.25	0.5	Reservoir and people have equal importance whereas stream fish is more important than the former two.
5	0.33	0.33	0.33	All three goals are equally important
6	0.1	0.1	0.8	Reservoir and people are not the priority whereas stream fish are much more important than the former two.
7	0.2	0.79	0.01	People is the most important, followed by reservoir, whereas stream fish has a much lower priority <sup>15</sup> .
8	0.4	0.2	0.4	Reservoir and stream fish have equal importance whereas people are less important than the former two.



**Figure 4. 11 Feasible Weight Area of Goal 1 – Reservoir**

<sup>15</sup> According to experts in water resource, WS1 and WS7 are often used in reality.

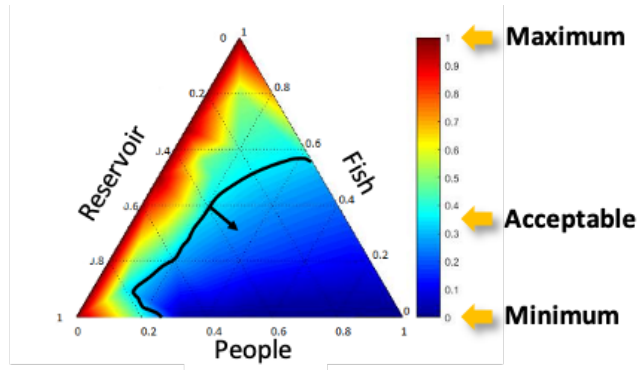


Figure 4. 12 Feasible Weight Area of Goal 2 – People

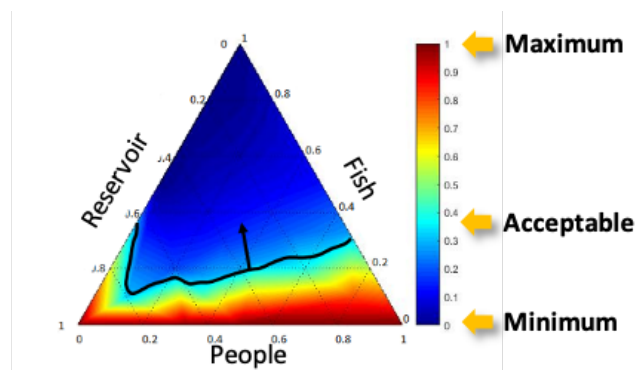


Figure 4. 13 Feasible Weight Area of Goal 3 – Fish

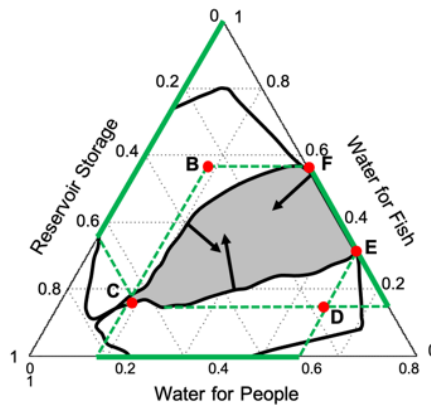


Figure 4. 14 Satisficing Weight Area for Three Goals<sup>16</sup>

<sup>16</sup> In Figure 12, the solid lines show the *satisficing* weight range of each goal, and the dashed lines are the auxiliary lines which facilitate us to identify the *satisficing* weight range. Every auxiliary line is parallel with one of the axes.

Identifying the Satisficing Range of Weights: we identify the range of the weights in the satisficing area, and show them in Table 4.6, using the ternary plot – we first identify the vertices of the area (C, F and E). Then we make lines parallel to the axes crossing the vertices. The range that is covered by the intersection of the parallel lines and the axes are the ranges of the satisficing weights. However, such ranges do not guarantee satisficing solutions, because we have enlarged the irregular satisficing area (the shaded area in Figure 4.16) to the Pentagon BCDEF. For example, Spot A is in the identified range of the weights (Table 5) but not in the satisficing area. So, when we select weights, we need to visualize them in a ternary plot to avoid taking a point not in the *satisficing* area as a *satisficing* weight. The ranges in Table 4.6 only give us a rough idea, but they do not ensure satisficing solutions.

**Table 4. 6 Range of Weights of the Satisficing Space**

<b>Weight</b>	<b>Range</b>
W1	0 – 0.65
W2	0.18 – 0.7
W3	0.15 – 0.56

*WS-Design Preference Look-Up Table:* Designers can use the WSs and results in Appendix A as a look-up table and meet different design preferences by setting appropriate weights.

### **Sensitive Segment Identification**

**Step 2.1 Identify Different Scenarios of Uncertain Parameters and Get Solutions.** We use ten inflow scenarios (ISs) to obtain solutions and identify the sensitive segments. The ISs are generated as follows.

*Using Inflow Scenarios:* To ensure that the frequency and severity of system failures within manageable levels, we test multiple scenarios of parameters with uncertainties (inflow scenarios – ISs).



In our ISs, we consider the extremely dry, flooding, and extremely uneven precipitation scenarios to study the model’s sensitivity. The ten ISs are listed in Table 4.7. The “No Rain” scenario means there is no inflow at all in any of the three months. The “Flood” scenario indicates in each of the three months, the inflow is four times the forecasted value.

We used the inflow scenarios (ISs) in Table 4.7 and weight scenarios (WSs) in Appendix A to obtain solutions, based on which we identify the sensitive segments.

**Table 4. 7 Inflow Scenarios (ISs) – Type I Uncertainty**

#	Inflow scenarios	The Percentage of the Inflow Based on the Forecast Value in Each Month		
		M1	M2	M3
1	No Rain	0%	0%	0%
2	Extremely Dry	20%	20%	20%
3	Dry	60%	60%	60%
4	Normal	100%	100%	100%
5	Rainy	150%	150%	150%
6	Extremely Rainy	200%	200%	200%
7	Flood	400%	400%	400%
8	Rain Unevenly I	0%	200%	100%
9	Rain Unevenly II	400%	0%	0%
10	Rain Unevenly III	400%	0%	400%

**Step 2.2 Identify Active Constraints/Bounds and Improvable Constraints/Bounds and Obtain**

**Solutions:** *Active Constraints/Bounds:* In Table 4.8, we give the active constraint/bounds and their active ISs and WSs, from which we observe that in all ISs, the lower bound of storage in Reservoir 7 has limited capacity, and in flood scenario, the upper bound of storage in Reservoir 6 has limited capacity.

**Table 4. 8 Active Bounds in Each IS and WS**

Active Bounds	Inflow Scenarios	Weight Scenarios	Physical Meaning of the Bounds
$CM_7^2$	1 – 10	All except 2, 4	The lower bound of storage volume in Reservoir 7 in Month 2 is relatively high
$CM_7^4$	1 – 10	All except 3	The lower bound of storage volume in Reservoir 7 in Month 4 is relatively high

$CF_6^3$	7	All except 2, 3, 4	The upper bound of storage volume in Reservoir 6 in Month 3 is relatively low
$CF_6^4$	7	All except 2, 3, 4	The upper bound of storage volume in Reservoir 6 in Month 4 is relatively low

*Improvable Constraints/Bounds:* The constraints and bounds with relatively large positive dual prices are identified and shown in Table 4.9. From Table 4.9, it is concluded that the upper bound of the storage volume in Reservoir 6 affects the achievement of the goals – if the upper bound is increased (relaxed), a better water flow plan can be obtained.

**Table 4. 9 Improvable Bounds in Each IS and WS**

<b>Improvable Bounds</b>	<b>Inflow Scenarios</b>	<b>Weight Scenarios</b>	<b>Physical Meaning of the Constraints or Bounds</b>
$CF_6^3$	1 – 10	1, 3, 26 – 28	The upper bound of storage volume in Reservoir 6 in Month 3 is relatively low
$CM_6^4$	1 – 10	1, 26, 28	The upper bound of storage volume in Reservoir 6 in Month 4 is relatively low

*Improvement Indication:* In conclusion, the lower bound of storage volume in Reservoir 7 is the active and non-improvable bound, and we can use the method in Step 2.3a to improve it; the upper bound of Reservoir 6 is the active and improvable bound, and we can use the method in Step 2.3b to improve it.

### **Model Improvement**

Using the improvement suggestions in Section 4.2, in this section, we carry out Step 2.3 and Step 3.

First, we explore the practicality of adding a buffer to the active and non-improvable constraints/bounds<sup>17</sup>.

---

<sup>17</sup> Obtaining better assessment of the precipitation to reduce uncertainty is one way of improving the

**Step 2.3a, 3.1a, and 3.2, Active and Non-Improvable Constraints/Bounds Improvement –**

**Restricting:** First, we explore the practicality of restricting the RHS of  $CM_7^t$ , which is to increase the lower bound of Reservoir 7 in the mathematical model without changing the physical level of the conservation pool. In this way we add a “buffer” to the active bound by making its mathematical model more restrictive than its physical condition. We illustrate this concept in Figure 4.17. Solution A is at the lower bound (lb) of Reservoir 7, which is sensitive to uncertainty in inflows. To bring it away from the boundary, we need to add a buffer to the RHS of the bound by changing the bound from lb to lb’ so that we can bring Solution A to Solution A’. Solution A’ is away from the physical bound lb.

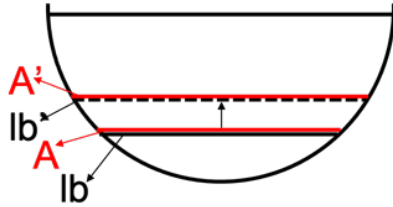
Then we explore the practicality of this change. By redesigning the capacity of each pool, we make the reservoir meet the requirements of multiple functions. To increase the lower bound of a reservoir means to increase the water level of the conservation pool. We can realize this by reallocating the water volume of the pools – we increase the water level requirement of the conservation pool by 1% and reduce the water level requirement of the flood pool to offset the change, because historical data and the physical condition of the reservoir inform us that such reduction is safe and attainable. As we make the physical change, we change the corresponding mathematical bounds as follows.

Original bound:  $S_7^t \geq CM_7^t$

Improved bound:  $S_7^t \geq CM_7^t * 1.01$ , Where  $t = 2, 4$

---

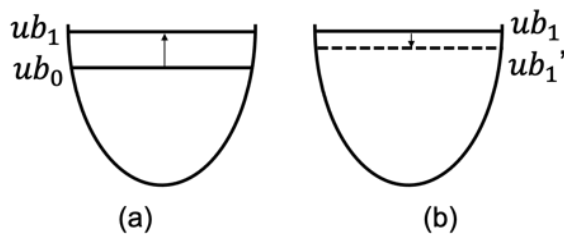
mathematical model, but as our precipitation data are provided by experts in water resource management, we do not tackle precipitation improvement in this case. Therefore, we focus on the capacity of the pools of the reservoirs.



**Figure 4. 15 Bring the Solution Away from the Boundary by Restricting the RHS**

**Step 2.3b, Step 3.1b and Step 3.3, Active and Improvable Constraints/Bounds Improvement –**

**Relaxing.** First, we explore the practicality of relaxing the upper bound of Reservoir 6, by increasing the upper bound of Reservoir 6. This action will benefit the achievement of the goals. We run this for several iterations until we get a solution with a dual price that is not relatively large, then we start treating it as an active and non-improvable constraint/bound and move the solution away from the boundary. However, if we continue iterating until we cannot physically relax the RHS any further but the constraint/bound still has a positive dual price, then we stop and deal with it as an active and non-improvable constraint/bound. This procedure is to explore the physical boundary of the system that gives the best achievement of the goals. Based on this physical boundary, we “add a buffer” to drive the solutions away from the physical boundary. In Figure 4.16-a, we illustrate this procedure. We explore the upper physical bound and bring the bound from  $ub_0$  to  $ub_1$  (Figure 4.16-a), in order to make the RHS of this bound reasonable and facilitate obtaining the best achievement of the goals. And then we add a buffer to  $ub_1$  by moving it to  $ub_1'$  as shown in Figure 4.18-b.



**Figure 4. 16 Applying the Physical Boundary by Relaxing RHS and Then Bring the Solution Away from the Physical Boundary by Restricting RHS**

Then, we explore the practicality of physically increasing the upper bound of the water storage volume in Reservoir 6 by 1%. Our data indicate that the maximum storage of Reservoir 6 is 75,573 cubic feet, which is 1.03% larger than the upper bound of 74,799 cubic feet, so we have the capacity to relax the RHS of this bound. Hence, we can increase the RHS of the upper bound of the Reservoir 6 by 1%.

$$S_6^2 \leq CF_6^2$$

$$S_6^2 \leq CF_6^2 * 1.01$$

In Table 4.10, the actions of model improvement are displayed. Thus, new solutions are obtained (Step 3) for the improved model (Appendix B). After iterating, obtain solutions that are away from the boundary and yield a relatively better achievement of the goals are obtained.

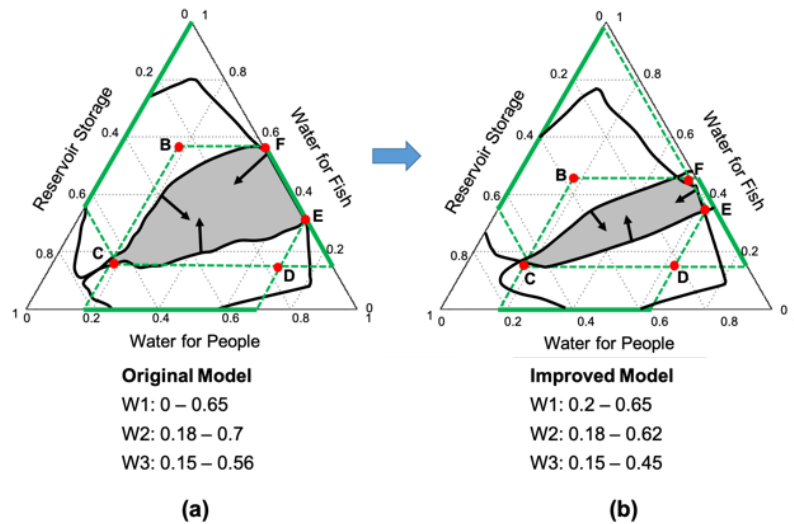
**Table 4. 10 Suggestions for Model Improvement**

#	Improvement	Constraints or Bounds function changes, i=1,2,3
1	Raise the lower bound of storage volume in Reservoir 7 in Month 2 and 4 by 1%	$S_7^t \geq CM_7^t$
		$S_7^t \geq CM_7^t * 1.01$ , where t = 2, 4
2	Raise the upper bound of storage volume in Reservoir 6 by 1%	$S_6^2 \leq CF_6^2$
		$S_6^2 \leq CF_6^2 * 1.01$

### Design Improvement Validation

Design Preferences of Improved Model: As we go through Step 1 to 3 for three iterations, we have no active constraints/bounds or improvable constraints/ bounds, so we stop the loop of exploration of the solution space. With the improved model in the final iteration, we obtain a new satisficing area of the weights. We compare the satisficing area of original model and improved model in Figure 4.17. The satisficing area and the corresponding range of weights become smaller after

model improvement because we get rid of the solutions that are sensitive to the uncertainties. In the perspective of robust design, we seek an insensitive solution and corresponding weight range, so the improvement is not to enlarge the satisficing area, but to refine the satisficing area to obtain relatively insensitive solutions.



**Figure 4. 17 The satisficing area of the weights of original model (a) and improved model (b)**

*Sensitive Segments of Improved Model:* As we go through Steps 2.1 - 2.3 with the improved model in each iteration, we identify new sensitive segments. The solutions are calculated using the improved mathematical model, but the active constraints/bounds should be identified using the original model because we need to know whether the solution is close to the physical boundary. As to the improvable constraints/bounds, we need to use the improved model in each iteration because we update the physical boundary through iterating. In Table 4.11, we list the sensitive segments in the second iteration, and their physical meanings, with improvement suggestions.

**Table 4. 11 Sensitive Segments of the Model of the Second Iteration**

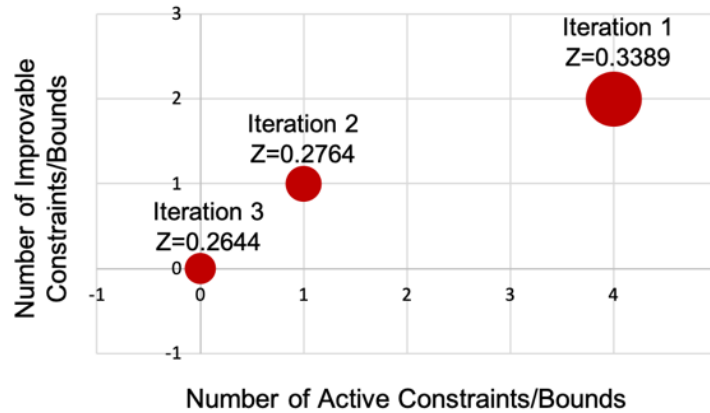
Active Bounds	Physical Meaning of the Bounds	Further Improvement Suggestions
---------------	--------------------------------	---------------------------------

S7M3L	The lower bound of storage volume in Reservoir 7 in Month 3 is relatively high	Raise the lower bound of storage volume in Reservoir 7 by 1%
Improvable Bounds	Physical Meaning of the Constraints or Bounds	Further Improvement Suggestions
S4M1	The storage volume in Reservoir 4 in the beginning of Month 1 is relatively low	Raise the lower bound of storage volume in Reservoir 4 by 1%

By carrying out the improvement suggestions in Table 4.11, in the third iteration, we have no sensitive segments, which means all the solutions are not at or highly close to the physical boundary, and there is no potential to further improve the achievement of the goals.

The evolution of the performance and the number of sensitive segments of the model along with iterating are illustrated in Figure 4.18. The horizontal and vertical axis respectively represents the number of active constraints/bounds and the number of improvable constraints/bounds. The size of the bubbles shows the average value of deviation  $z$ , of the 34 WSs and 10 ISs. In this case, improvement means driving the bubble from the up-right corner to the down-left corner, while making it smaller. As it is shown in Figure 4.18, by iterating, we improve the insensitivity of the model from (4, 2) to (0, 0), and we improve its performance by reducing  $z$  from 0.3389 to 0.2644.

After three iterations, our model is neither sensitive to the uncertainties in inflows nor has potential to improve the completeness of the goals. It is guaranteed that any extreme weather does not cause any system failure because now all the solutions in all circumstances are away from the physical boundary. The Three-Step Method of exploration of the solution space is summarized as an algorithm in Table 4.12.



**Figure 4. 18 Improvement through Iterating**

**Table 4. 12 The Algorithm for Model Improvement**

- 
- a. Identify  $n$  scenarios of parameters with uncertainties – ISs.
  - Begin Iterations of Exploration of the Solution Space**
  - b. Use the latest model to identify the feasible area of weights and identify  $m$  weight scenarios within the feasible area of weights that represent different design preferences – WSs.
  - c. Plug  $n$  ISs and  $m$  WSs into the latest model to get  $x$  solutions.
  - d. Plug  $x$  solutions into the model in the first iteration and into the model in the current iteration
  - e. Identify active constraints/bounds using the model in the first iteration
  - f. Identify improvable constraints/bounds using the model in the current iteration
  - g. **if** no sensitive segment (active or improvable constraints/bounds)
    - Go to l.**
  - else**
    - Continue with **h.**
  - h. **For** each active and non-improvable constraint/bound
    - Explore the practicality of restricting their RHSs
  - i. **For** each active and improvable constraint/bound
    - Explore the practicality of relaxing their RHSs to the physical bound
  - j. Make model improvement plans based on the conclusion in **h** and **i**.
  - k. Improve the model based on the improvement plans in **g** and **go to b.**
  - l. The latest model is relatively insensitive to uncertainties and has no potential for improvement.
- End the iteration**
- 

#### **4.2.6 Closure of Test Problem I**

In Section 4.2, we formulate a 14-dam-network problem using the compromise Decision Support Problem and explore the solution space to improve the model and obtain *satisficing* solutions that are relatively insensitive to the uncertainties in water inflows. We identify key questions and give answers by using a three-step method to explore the solution.



## **Answers to the Research Questions**

Q1.1 How can designers manage multiple conflicting goals with reasonable priorities under different circumstances?

Step 1 – With design scenarios representing different design preferences, designers identify the satisficing area of the weights of the multiple goals and provide their physical meanings.

Q1.2 How can designers manage uncertainties in the inflows by making water flow plans that are relatively insensitive to these uncertainties?

Step2 Using inflow scenarios considering different weather and climate conditions to identify the sensitive segments. Exploring the practicality of removing the sensitive segments by modifying the mathematical model and adjusting the physical system.

Q1.3 How can designers improve the mathematical model as well as the physical system to be relatively insensitive to uncertainties?

Step 3 – Using the information of the practicality of modifying the model and changing the system, make improvements, including changing parameters' value to add buffers, and reallocating water to different pools to change the physical boundary. Then go through the three steps again. By running such a loop, improve the model to give satisficing solutions that are relatively insensitive to uncertainties.

With satisficing solutions, we reduce the frequency and severity of discrepancies between water supply and water demand. The advantage of our method is that we boost the potential of the physical system while improving its robustness, hence we neither sacrifice system robustness for a better performance nor do the opposite.

The dam-network planning problem only incorporates continuous variable, and no coupling decisions are required. In Section 4.3, a supply chain design problem with mixed variables and coupling decisions is discussed, and the Three-Step Exploration method is advanced to a decision framework – the Formulation-Exploration framework.

### **4.3 Positioning the Customer Order Decoupling Point of a Supply Chain**

*- Test Problem 1.2: apply Formulation-Exploration framework to a discrete problem*

*The new knowledge from managing Test Problem 1.2 is*

*A framework that incorporates the Three-Step Exploration Method that allows to identify the sensitive elements of a mixed-variable, coupled decision model and improve the model accordingly – the Formulation-Exploration framework.*

As globalization continues in Industry 4.0, manufacturing enterprises need to do mass customization (MC) in a short lead-time to satisfy evolving market demands in different regions. One challenge of MC is to fulfill orders swiftly at an acceptable cost, meanwhile maintaining the service quality. To do this, the customer order decoupling point (CODP), where the value-adding activities take place, should be designed and adapted to the changing market demands.

In this section, a Formulation-Exploration framework is proposed to make decisions on CODP positioning and improve the SC to support MC. A test problem of auto parts manufacturing is used to establish the efficacy of our method. The Formulation-Exploration framework can be used to design SC to facilitate MC of products, especially when information is incomplete and inaccurate, goals conflict and multiple types of uncertainty add complexity.

### **Glossary (mainly applied to section 4.3)**

cDSP	compromise Decision Support Problem
CODP	Customer order decoupling point
DCI	Design capability index
EMI	Error margin index

ESS	Exploration of the solution space
MTS	Make-to-stock production mode
MTO	Make-to-order production mode
PLC	Product life cycle

#### **4.3.1 Problem Statement – Test Problem 1.2: CODP and the Challenges in Supply Chains**

Mass customization (MC) dramatically enhances the emotional interaction between the designers and the customers (Parker 2016). MC is proposed by Naylor et al. (Naylor, Naim et al. 1999) as to combine the agile and lean as a new strategy – *leagile*, and it is defined and broadly accepted as “postponing the task of differentiating a product for a specific customer until the latest possible point in the supply network” (Jacobs, Chase et al. 2004). Accordingly, being able to determine the differentiation point of a product to adapt to the rapidly changing market is of great importance in supply chain (SC) design. This point is normally defined as the customer order decoupling point (OPP) or the customer order decoupling point (CODP). The CODP is also known as the boundary between make-to-stock (MTS) production and make-to-order (MTO) production (Hajfathaliha, Teimoury et al. 2011), where the value-adding activities take place (Rudberg and Wikner 2004).

In Figure 4.19, we illustrate different characteristics of production before (upstream from) and after (downstream from) the CODP of a SC. In MTS production (before CODP), manufacturers make production plans based on demand forecast and pursue physical efficiency. MTS is often used in upstream SCs in labor-intensive industries, where low-cost and on-time delivery are prioritized. On the contrary, in MTO production, manufacturers produce goods to meet customers’ orders and emphasize market response and capacity flexibility. MTO is often used in the downstream SC of high-tech industries, or after-market items (Iravani, Liu et al. 2012), where differentiation and personalization are valued.

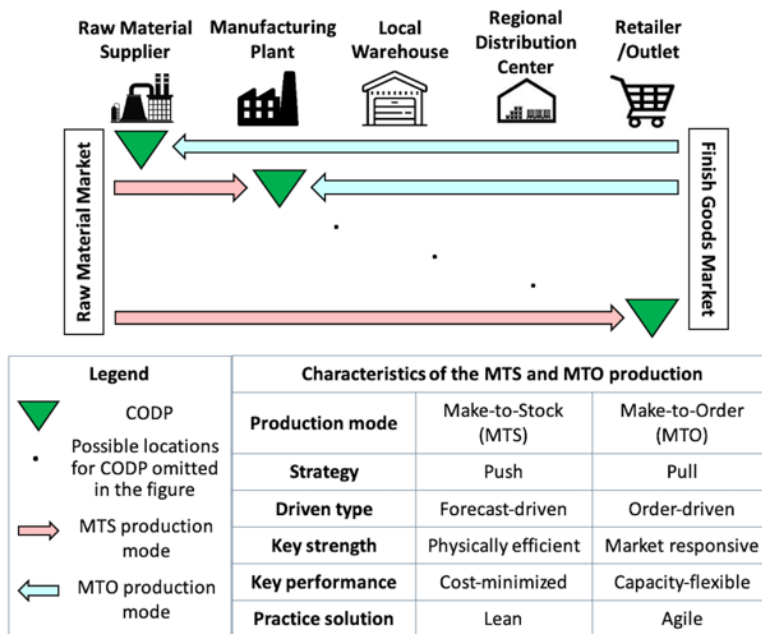


Figure 4. 19 Possible location of CODP in a SC

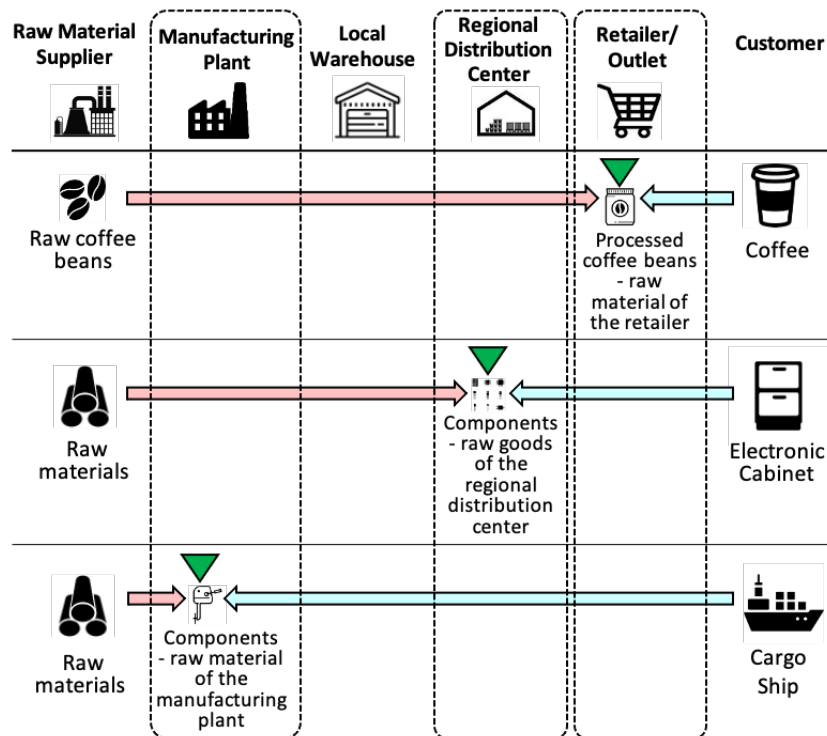


Figure 4. 20 CODP of Different Industries

Theoretically, the CODP of a product can be anywhere in the supply value chain. Figure 4.20, we show some typical CODPs in different industries. In the coffee industry, the CODP is the raw material of the retailer. When an end customer orders a cup of coffee, the retailer starts making it. In the electronic cabinet SC, MTS starts from the raw materials to the regional distribution center (RDC) where stores the standard components. After receiving orders, the assembly plant gets the customers' needs on exact models and the corresponding parts, then starts assembling and shipping. As to the shipbuilding industry, e.g., cargo ships, the whole SC is MTO due to its highly customized nature. There are common features of the CODP for different goods: convenient storage, commonality and short lead-time requirement of the goods upstream from the CODP, whereas multiple consumption channels, affordable production shifting cost and relatively long but acceptable lead-time of the goods downstream from CODP, etc.

The aim of MC is providing products to satisfy individual customer's requirements with near mass production efficiency (Salvendy 2001). Usually, the efficiency of a SC varies significantly as the CODP changes. For a single product, the CODP should vary in different phases of its life cycle to support MC. The rapid product upgrades and replacements require fast CODP switching. From the demand side, the users are continuously putting forward new needs, which force the CODP to go upwards along the SC. To maintain a near mass production efficiency in a SC with fast CODP switching is a challenge because low cost and product differentiation are contradictory. Therefore, a method that facilitates the exploration of the tradeoffs between cost and flexibility is needed.

### ***Challenges in SCs***

In the age of Industry 4.0, multiple types of workflows exist in large-scale supply networks which involve a significant number of entities of different types (Velasquez, Khakifirooz et al. 2019). A large number of goals that may conflict with each other need to be managed (Figure 4.21). Their

priorities may evolve as circumstances change. Since mathematical models used to define SCs are typically abstractions of reality, we are guided by Simon's maxim of looking for *satisficing* (good enough – not optimum) solutions [22].

According to George Box, “all models are wrong, but some are useful”(Box and Draper 1987), the mathematical models that constitute a SC are incomplete and inaccurate (Talvitie 1981) and of different fidelity. Decision model does not represent the physical world perfectly (Norman 1990). Hence, a designer should work with decision models that embody incompleteness and errors (Simon 1996).

Multiple types of uncertainty in decision models affect the performance of the SC (Nellippallil 2018). In Table 4.13, we illustrate the four types of robust design in association with the four types of uncertainty, which are defined, extended and summarized by Taguchi (Taguchi and Clausing 1990, Taguchi 1993), Chen et al. (Chen, Allen et al. 1996), Isukapalli (Isukapalli, Roy et al. 1998), Choi et al. (Choi, Austin et al. 2004, Choi, Austin et al. 2005, Choi, Mcdowell et al. 2008) and well interpreted by Nellippallil et al. (Nellippallil, Song et al. 2017). In this section, we tie it to the SC. The four types of uncertainty in mechanical engineering have equivalents in the SC (Table 4.13).



**Figure 4. 21 Multiple Conflicting Goals in SCs**

**Table 4. 13 Four Types of Robust Design and the Interpretations in SC**

Types of robust design	Types of uncertainty	Example in SC	Quantification
<b>I</b>	Uncertainty in parameter	Uncertainty in the demand side, such as unpredictable order	Type I, II: EMI, Monte Carlo simulation, Latin hyper cube, First / Second moment method, etc.
<b>II</b>	Variable uncertainty	Uncertainty in the supply side, such as variation in productivity	
<b>III</b>	Uncertainty in model structure	The number of decision variables, constraints, or mathematical relation between variable changes due to machine failure, customer loss, impatient customer or natural disaster	Type III: Design Capability Index (DCI), Variance Function Estimation, Prediction Interval Approach, etc.
<b>IV</b>	Uncertainty created in process chain	The bullwhip effect	Type IV: <i>Satisficing</i> , Exploring the Solution Space (ESS), etc.

Type I robust design is relatively insensitive to noise factors, that are uncontrollable parameters such as uncertain demand. MTS production is planned based on the demand forecast which may be different from the real demand. MTO production is planned based on the in-taken orders which may be more urgent than usual or withdrawn later. Hence, demand is a parameter with uncertainty that cannot be controlled by the decision maker.

Type II robust design is relatively insensitive to variations in design variables such as uncertain supply. The actual output of production can be different from the expected volume, because of

overuse of the machines, overtime of the labor, etc. The decision maker may set the production volume to a certain level, but the actual output can be different. The error margin index (EMI) is used to deal with Type I and Type II uncertainty by bringing the mean and minimizing the variance of the performances. EMI is a mathematical construct indicating the location of mean system performance and the spread of the performance considering variability in design variables (Choi, Austin et al. 2004). Type III robust design is relatively insensitive to uncertainties embedded within the model structure, such as dimensionality, constraints or the relation between variables due to unexpected events, such as overcapacity, infrastructure damage, the political environment changes. The design capability index (DCI) is used to handle Type III uncertainty. DCI is a mathematical construct for efficiently determining whether a ranged design specification is capable of satisfying a ranged set of design requirements (Choi, Austin et al. 2005). Using a DCI, the designer identifies a range of design variable with the minimized design specification variation and an acceptable performance variation. Within this range, the impact of the model structural change on the model performance is minimized, meanwhile, the performance is good enough and with relatively small variance. Details of using EMI and DCI can be found in (Chen, Allen et al. 1996, Choi, Austin et al. 2005, Choi, McDowell et al. 2008, Choi, McDowell et al. 2008, Nellippallil 2018).

In Type IV robust design, the propagated uncertainty in process chains and the interaction among the three types of uncertainty are managed, for example, the bullwhip effect. We propose to explore the solution space (ESS) to improve the insensitivity of the design considering various situations. Using ESS, we explore design preference, improve design capacity, and boost the system potential. The method is introduced in detail in Section 4.3.3.



### ***4.3.2 Literature Gap Analysis – in the Domain of Customer Order Decoupling Point Determination***

There are two main approaches to locate the CODPs in SC. One is the qualitative approach, through which the researchers make general rules on locations of CODP for different types of products. For example, CODP should be closer to the designer/manufacturer for personalized products, luxury goods or services. The other approach involves quantitative methods – designers position the CODP by applying mathematical modeling. The latter is mainstream in recent years. we focus on the quantitative approach in the literature review.

**Queuing model.** In a series of articles (Teimoury, Modarres et al. 2012, Teimoury and Fathi 2013), Teimoury et al. use queuing models to position CODP and make relevant decisions in SCs. In (Teimoury, Modarres et al. 2012), they develop a queuing and the matrix-geometric method to determine CODP in a multi-product SC. Customer orders are assumed to arrive according to the Poisson process. The objective is to minimize total cost with constraints of warehouse capacity and service level. The effect of impatient customer arrival on costs is considered. The assumption of the Poisson customer-order arrival may cause errors since for different products, or in different phases of the product life cycle, the customer-order arrival varies a lot. In (Teimoury and Fathi 2013), Teimoury and Fathi model a two-echelon SC offering multiple products to customers with different preferences, considering both shared and unshared capacity models. They develop an integrated operations-marketing perspective with considering price-sensitive demand for CODP positioning based on multi-class queueing model. Besides, Zhou et al. (Zhou, Huang et al. 2014) apply a two-stage queuing model with customer orders modeled as Markov processes and find that the variance and correlation of the demand increase the total operation cost. The effect of the CODP's position on the unit inventory holding cost, the lead-time quotation policy, and the penalty

cost for tardiness are considered. However, the single objective is simplified as minimizing the total cost.

**Time scheduling model.** Liu et al. (Liu, Wu et al. 2018) propose a time scheduling model to examine the dynamic positioning of CODP. Minimizing total cost and maximizing customer satisfaction are the two objectives while the incremental cost of a new order, lead-time and changing values of objectives are constraints. However, they do not scale the two objectives with different units – scaling is necessary because the importance and utility of the two objectives may change with circumstances. Jeong (Jeong 2011) propose a dynamic model based on product life cycle theory to position the CODP and make a production-inventory plan. The author applies optimal control theory to with a quadratic objective function. However, the author does not consider factors as time-varying penalty costs and the delivery lead-time.

**Stochastic programming, multi-criteria optimization, dynamic programming, etc.** Some authors apply programming methods with stochasticity or multiple criteria. Ghalekhondabi et al. (Ghalekhondabi, Sormaz et al. 2016, Ghalekhondabi, Ardjmand et al. 2017) leverage stochastic programming model to identify CODP by minimizing total cost or maximizing total profit. Constraints of working time, satisfied order percentage and manufacturing capacity are considered. Customer orders are assumed to follow Poisson distribution. Shidpour et al. (Shidpour, Da Cunha et al. 2014) propose a multi-objective programming model based on manufacturer's profit and customer perceived value to analyze the impacts of single-CODP and multi-CODP while considering service time constraint. They conclude that multi-CODP is preferred than single-CODP for a product portfolio, benefiting both the manufacturer and the customers. Ahmadi et al. (Ahmadi and Teimouri 2008) develop a dynamic programming model to optimize CODP by minimizing holding cost, delivery delay and number of modules not ready to be assigned. They

assume that customer demand is deterministic. Liu et al. (Liu, Mo et al. 2015) construct a linear programming model with discrete variables in the logistics industry to position CODP by minimizing total cost of logistics service integrator. Procedure constraints and lead-time constraints are considered in the model. They conclude that the optimal CODP is not affected by decreasing order-transferring and waiting cost but driven to the last procedure when order processing cost decreases. Other methods such as analytic hierarchy process (AHP) model (Xu and Liang 2011), value network simulation model (Daaboul, Laroche et al. 2010, Daaboul, Da Cunha et al. 2015) and tandem forecast-driven and order-driven simulation model (Wikner, Naim et al. 2017) are also explored to identify the optimal CODP.

In summary, recent researchers in CODP positioning try to address the issue by constructing models with multiple objectives and constraints. There are employed traditional optimization methods such as stochastic programming, dynamic programming and selection model such as AHP, as well as methods “borrowed” from other domains such as value network, tandem circuit design and polychromatic set theory. Researchers find various metaphors for CODP and SC design, to come up with interesting concepts and methods – such as viewing the MTS and MTO production mode as two circuits and looking for an appropriate way to connect them. This phenomenon implies that CODP design is a field combining science and art. With different metaphors, designers can have diverse perspectives and foci. Their observations are interesting and somehow useful to an extent in some situations. However, there are some limitations, and we summarize them as follows.

**Limitation 1** – There are assumptions in the distribution of non-deterministic methods (Teimoury, Modarres et al. 2012, Zhou, Huang et al. 2014, Ghalekhondabi, Sormaz et al. 2016, Ghalekhondabi, Ardjmand et al. 2017). They assume that enough data are supporting making

decisions on the distribution of customer-order arrivals, demand forecast accuracy, customer satisfaction quantification, etc. These decisions or assumptions can be wrong and irreversible once the design stage is over, and no adjustments can be made during operations.

**Limitation 2** – The optimization methods (Ahmadi and Teimouri 2008, Jeong 2011, Teimoury, Modarres et al. 2012, Shidpour, Da Cunha et al. 2014, Zhou, Huang et al. 2014, Liu, Mo et al. 2015, Ghalehkhondabi, Sormaz et al. 2016, Ghalehkhondabi, Ardjmand et al. 2017) are used to find optimal solutions, that are solutions on the boundary of the feasible solution space (for deterministic approach) or distributed close to the boundary of the fussy feasible solution space (for stochastic approach). As models are incomplete, inaccurate, and embody different levels of fidelity, the solution may be optimal to the model but may not be optimal to the real problem which is way more complicated than the model.

Facing the challenges in the SCs and MC, we need to overcome the limitations. The assumptions on data adequacy can be replaced with the information we obtain from post-solution analysis. Typical uncertainties need to be managed in the design and operation stage and positioning strategies regarding the CODP should be tailed for different market environments or different phases in the product life cycle.

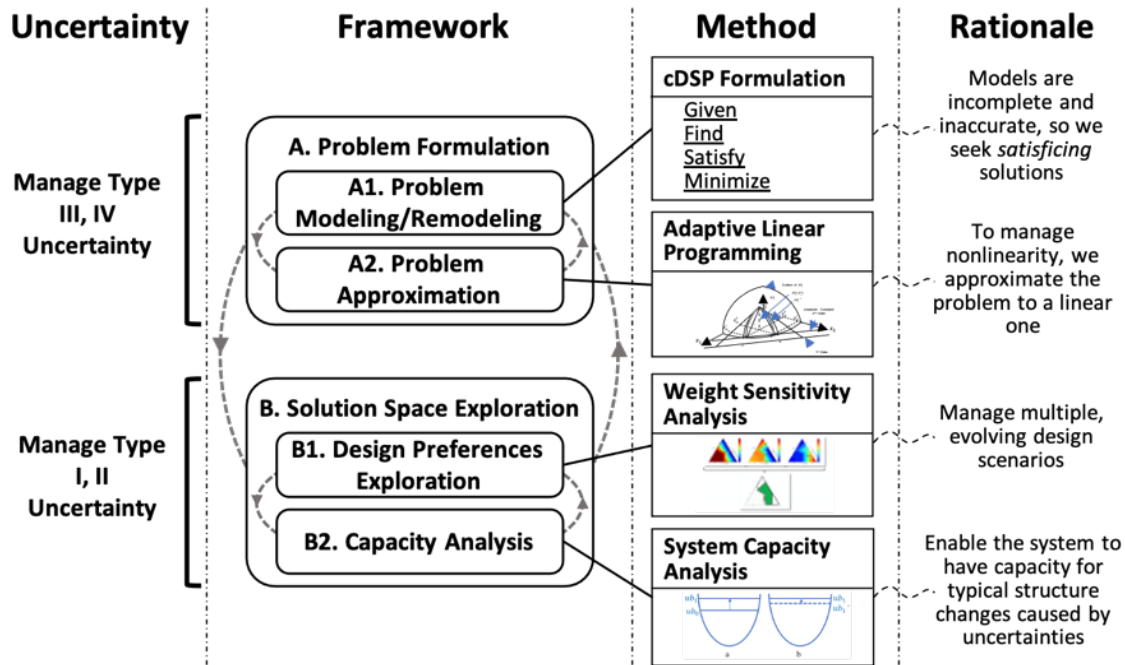
Based on the limitations in the literature and challenges in CODP design and RQ1, we pose a question and hypothesis specified in the domain of SC. By answering the question and validate the hypothesis, we bridge the gap.

**Question** – *How can designers improve the robustness (insensitivity) of the CODP design without sacrificing the performance of the SC?*

**Hypothesis** – We propose a Formulation-Exploration framework to explore the solution space with respect to the design capacity under typical uncertainties. With insight obtained from solution space exploration, we remove some errors and heuristics in the initial design and boost the potential of the system.

### 4.3.3 Proposed Methods – The Formulation-Exploration Framework

To obtain *satisficing* solutions insensitive to multi-type of uncertainty, the enterprises need to apply some systematic methods to make in-time decisions supporting MC. We propose the Formulation-Exploration framework (Figure 4.22). The method consists of two stages, namely, problem formulation and solutions space exploration. To converge on a *satisficing* solution, we need to iterate between the two stages.



**Figure 4. 22 Formulation-Exploration Framework**

**Compromise Decision Support Problem (cDSP).** In the problem formulation stage, the problem is formulated as a cDSP and then solved using the Adaptive Linear Programming algorithm

(Mistree, Hughes et al. 1993). The mathematical form of a cDSP and why it delivers *satisficing* solutions are stated in Section 2.3. Due to the complexity of the SC with MC, the cDSP usually contain non-convexity and nonlinearity features, so one way of managing the complexity is to approximate the model to a classic linear model and solve the problem using simplex algorithm. The model structural uncertainty (Type III) and process chain added uncertainty (Type IV) are tackled in the problem formulation session.

In the second stage the solution space is explored to find *satisficing* solutions associated with each design preference (scenario), in different phases in the product life cycle. Type I and II uncertainty are managed in the exploration of the solution space (ESS). In this section, when referring to design preferences, we particularly focus on the importance of the different goals.

**Weight sensitivity analysis – exploration of the design preferences.** We use weight sensitivity analysis to explore how the different weight affect the system performance regarding the achievement of the goals. See Figure 4.22, “weight sensitivity analysis” (Ahmed, Goh et al. 2014).

**System capacity analysis – identification and management of the sensitive segment and bottleneck.** To overcome the capacity limitation of constraints or bounds, we use “system capacity analysis” method to identify the sensitive segment and bottleneck (Table 4.13). If an inequality constraint has zero or tiny surplus or slack comparing with its right-hand-side value, we define it as an active constraint. The solution is on or close to the boundary of the active constraint, so the solution is sensitive to the uncertainty of active constraint. If the shadow price of an active constraint is lower than other active constraints, by relaxing such an active constraint, we may not get considerable improvement of the fulfillment of the goals, and we define such a constraint as a “sensitive segment”. We then bring the solution away from the sensitive segment by restricting the active constraint, which means adding a buffer to the mathematical model for preventing the

solution reaching the physical boundary. If the shadow price of an active constraint is the largest value in compare with that of other constraints, relaxing the constraint can result in the most improvement of the fulfillment of the goals, and we define such an active constraint as a “bottleneck”. Also, we need to find ways of relaxing the bottleneck in the physical system to boost the system potential, and once there is no more potential of physically relaxing, we bring the solution away from the newly-relaxed boundary by restricting the constraint in the mathematical model – add a buffer to the physical boundary. In such a way, we boost the potential of the physical system while improving its robustness, hence we neither sacrifice system robustness for a better fulfillment of the goals nor do the opposite.

**Table 4. 14 Algorithm for System Capacity Analysis**

---

**Step 1.** Identify  $n$  scenarios of parameters/bounds with uncertainties – ISs.  
**Begin Iterations of Exploration of the Solution Space**  
**Step 2.** Use the latest model to identify the feasible area of weights and identify  $m$  weight scenarios within the feasible area of weights that represent different design preferences – WSs.  
**Step 3.** Plug  $n$  ISs and  $m$  WSs into the latest model to get  $x$  solutions.  
**Step 4.** Plug  $x$  solutions into the model in the first iteration and into the model in the current iteration  
**Step 5.** Identify active constraints/bounds and sensitive segment using the model in the first iteration  
**Step 6.** Identify bottleneck using the model in the current iteration  
**Step 7.** if no sensitive segments or bottleneck  
    **Go to 12.**  
    **else**  
        Continue with h.  
**Step 8.** For each sensitive segment  
    Explore the practicality of restricting their right-hand-side values (RHSs) in the model  
**Step 9.** For each bottleneck  
    Explore the practicality of relaxing it in the physical system  
**Step 10.** Make model improvement plans (restricting model or relaxing physical system) based on the conclusion in 8 and 9.  
**Step 11.** Improve the model based on the improvement plans in 7 and **go to 2.**  
**Step 12.** The latest model is relatively insensitive to uncertainties and has no potential to achieve a better solution.  
**End the iteration**

---

*We hypothesize that using the Formulation-Exploration framework, the CODP can be designed/redesigned to obtain satisficing solutions, which supports MC. A successful company should be able to do repeated design based on the information attained from the previous cycle,*

*so its SC can be flexible for MC. The Formulation-Exploration framework allows us to answer the research question in this section and validate hypothesis.*

In Section 4.3.4 and 4.3.5, we apply the Formulation-Exploration framework in auto-industry for more investigation.

#### **4.3.4 Model Formulation**

The test problem is a three-echelon SC in automobile industry with three players – a supplier, a manufacturer, and a retailer as shown in Figure 4.23. We study the problem for a period of one-month. Our goals are to position the CODP, and determine the appropriate reliability of each process including the operation for each player and the transportation between two players. The concept of the reliability is from the lean process management and the six-sigma quality control principle that the processes in automobile industry range from three sigma (99.7%) to six-sigma (99.99997%). The constraints are cost and capacity in each process. We formulate the problem in a cDSP (See Appendix C). The goals and constraints of the model are visualized in Figure 4.23.

**Goals.** We have three goals – profit, service level and variance of reliability of processes. We have a target for each goal. The profit target is an ideal value that not only make each player in this SC sustainable, but also make them extremely competitive by providing the stakeholders the greatest return – the biggest in the industry. We simply the problem by assuming the unit price of the product within the modeling time is fixed and the factors affect the profit are the cost of all the processes. In this way, we focus on the process improvement regarding CODP positioning.

The service level, as the second goal, is represented by multiplying the reliability of all processes in the SC, including the procurement, manufacturing, holding, handling (the display, movement and maintenance in the retailer) and transportation. This definition of service level is from the



reliability of mechanical system with multiple components. In this sense, only the reliability of all processes maintains a high level ensures an acceptable service level of the entire SC. We set the target of the service level as the six-sigma, 97.99997%.

The variance of reliability of processes is the third goal. The aim of setting such an goal is to enable all players of the SC to share the duty and pressure of reliability, without over-relying on certain processes to maintain high reliability while ignoring the reliability management of some other processes. This is to prevent strong players from dominating the other players by squeezing their profit margins. For example, strong retailers have very strict requirements on the manufacturer's unit price, product quality, delivery time, etc., which can be reflected as extremely high reliability of the manufacturer. If we minimize the differences between the reliability of different players, the responsibility shirking and domination between players can be avoided to some extent. The overall performance and profit of the entire SC can be boosted and the sustainability of the SC as well as each player can be maximized.

***System variables.*** We have two types of system variables, binary variables and continuous variables. The CODP of each candidate locations are binary variables. Each player has two candidate positions that may store the goods, raw material or finished goods, so there are six candidate locations for CODP in this SC, namely, supplier's raw material (SRM), supplier's finished goods (SFG), manufacturer's raw material (MRM), manufacturer's finished goods (MFG), retailer's raw material (RRM), and retailer's finished goods (RFG). In each design scenario that represents a certain design preference, one of the six candidate locations should be the CODP. The production upstream to the CODP is make-to-stock and the production downstream to the CODP is make-to-order.

The reliability of the processes are continuous variables. We define the reliability is a value that each process owner can determine. High reliability result in high service level but low profit. By exploring the tradeoffs between profit and the other two goals, we can set appropriate value to the reliability of each process in different situation, such as different stages of product life cycle.

The production volume of the supplier and the manufacture and the purchasing volume of the retailer are either equal to the forecast (if the production at the point is make-to-stock) or equal to the real demand (if the production at the point is make-to-order), so the production volume are defined as dependent parameters rather than system variables.

**Constraints.** The cost of every process, the total number of CODP, and the service level are the constraints. Holding cost: in the processes in the make-to-stock production, when the real demand is less than the forecast, the extra part of the forecast becomes the remaining stock, which causes holding cost until next time period, the remaining stock offsets the production or procurement volume of the next time period. Shortage cost: in the processes in the make-to-stock production, when the real demand is more than the forecast, we assume the shortage amount can be outsourced urgently, but the urgent outsourcing causes extra cost, and we define such extra cost as the shortage cost.

In this problem, we define that there is always one CODP in this SC. The CODP can be changed as the product life cycle evolve but at any specific time, it is required that all raw material or semi-finished goods or finished goods to be stored at the single CODP.

The ideal value of the service level of the SC is defined as the target of the service level goal, but we also have a lower bound of the service level, which is the three Sigma, 99.7%. The SC is not sustainable if the service level gets lower than 99.7%.

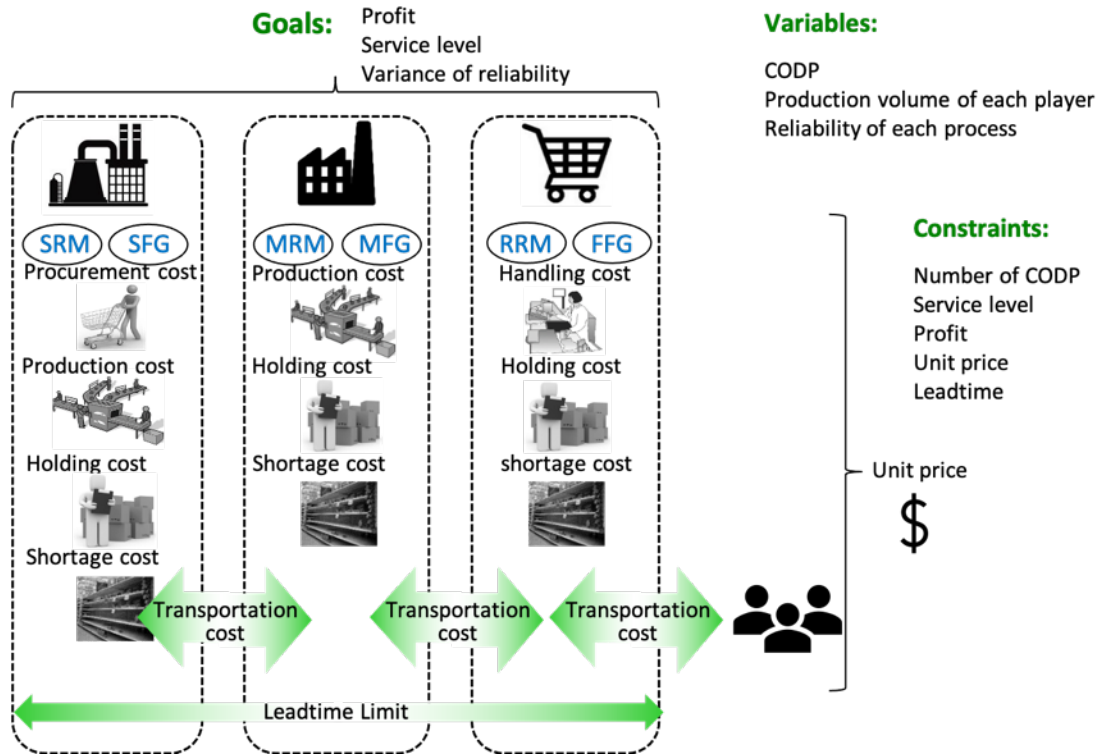


Figure 4.23 A Three-Echelon SC

*Different design scenarios* – different stages of the product life cycle (PLC). As the product enters different phases of its life cycle – introduction, growth, maturity, and decline, the preferences on the three goals vary, the CODP may move up and down accordingly.

#### 4.3.5 CODP Results and Discussion

*- Applying the Formulation-Exploration framework to improve the model and obtain satisfying solutions*

**Weight sensitivity analysis.** In design preferences exploration, we use a number of weight scenarios (WSs) to represent typical design preferences. We apply the method in (Seada and Deb 2014) to determine the WSs. For a  $M$ -goal problem, if the designer want to divide the weight range for each goal – we define the whole range is  $[0, 1]$  – into  $p$  pieces, we will have  $H = \binom{M + p - 1}{p}$

number of WSs. In this problem, since there are three goals,  $M = 3$ . We set  $p=2$ , so  $H=6$ . We want one more scenario that assigning equal weights (1/3, 1/3 1/3) to three goals, thereby there are all together seven WSs (Table 4.14). Plugging each of the seven WSs into our cDSP, we have seven sub-problems. Solving them one by one, we obtain the results listed in Table 4.15.

We use Figure 4.24 to show the achievement of the goals associated with the CODP results. From Figure 4.24, we conclude that when profit is less important than service level and variance of variability, the CODP should be at the finished goods of the retailer (RFG); if the profit has some importance, the CODP should be at the raw material of the manufacturer (MRM).

**Table 4. 15 Seven Scenarios – Type II Uncertainty**

WS	w1	w2	w3
1	1	0	0
2	0	1	0
3	0	0	1
4	0.5	0.5	0
5	0.5	0	0.5
6	0	0.5	0.5
7	0.33	0.33	0.33

**Table 4. 16 Results of the First Iteration**

WS	CODP	Profit	Service Level	Variance of Reliability
1	MRM	30489	99.13%	0.483%
2	RFG	33835	99.26%	0.304%
3	RFG	33569	98.33%	0.045%
4	MRM	30610	99.40%	0.549%
5	MRM	30233	98.17%	0.055%
6	RFG	33824	99.26%	0.279%
7	MRM	30543	99.28%	0.249%

WS							Profit, Performance stability, and Product life cycle
	SRM	SFG	MRM	MFG	RRM	RFG	
1			▼				Low profit, unstable – decline
2						▼	High profit, unstable - maturity
3						▼	Hi profit, stable - growth
4			▼				Low profit, stable - introduction
5			▼				Low profit, stable - introduction
6						▼	High profit, unstable - maturity
7			▼				Low profit, stable - introduction

**Figure 4. 24 CODP of Different Weight Scenarios**

**System capacity analysis.** We identify the holding cost of the manufacturer and retailer have limited capacity but a relatively large shadow price, which means this constraint is a bottleneck. So, if we can reduce the holding cost physically by some means (e.g., applying RFID technology, or using a third-party vender with lower unit cost), the achieved value of the goals can be improved. Assuming that the operational team take actions of the cost reduction, we reformulate the problem accordingly using the algorithm in Table 4.13. Then we run the second iteration of the Formulation-Exploration. After three iterations of formulation-exploration, we get the results in Table 4.16. The final results are better than the first-iteration results (Table 4.15) in terms of the achieved value for the goals, whereas the service level and variance of reliability in some scenarios are worsen but not much. This means that we save the cost in achieving unnecessarily high service level and equity between players to obtain higher profits. After three iterations, we have boosted the system potential, so we stop iterating.

**Table 4. 17 Results of the Seven Design Scenarios in the Third Iteration**

WS	CODP	Profit	Service Level	Variance of Reliability
1	MRM	31584	99.13%	0.483%
2	RFG	36946	99.21%	0.483%
3	RFG	36657	98.25%	0.025%
4	MRM	31584	99.13%	0.483%

5	MRM	31584	99.13%	0.483%
6	RFG	33813	99.23%	0.315%
7	MRM	31584	99.13%	0.483%

In this case, we observe that in WS 1, 4, 5 and 7, when the profit goal (Goal 1) is achieved not so good, ), the CODP is at MRM, the raw material of the manufacturer, which means if the decision maker allows the SC to give up some of its profit margin, (in the introduction phase, or decline phase of the product life cycle – PLC), In such a situation, from SRM to MRM (from the raw material of the supplier to the raw material of the manufacturer), the productions are all based on the demand forecast; from MRM to RFG (from the raw material of the manufacturer to the finished goods at the retailer), the production or goods preparation are all based on customers' order. Standard components are stored at MRM and all the locations upstream from it, and customization takes place right after MRM. The profit is not in a high level, whereas the achievement of service level (the higher the better) and the variation of the reliability (the lower the better) are not always in a high level either. This indicates that in the introduction or decline phase of a product, we need to sacrifice the performance of the SC for a better customized product and a relatively more flexible SC.

On the contrary, if the profit is the first priority for the SC, usually happens in the growth phase or the maturity phase of the PLC, the CODP should be at the last candidate node of the SC, that is RFG, the finished goods of the retailer. In this way, the whole SC is an MTS one. The relatively stable and strong demand from the customers make the demand forecast more reliable, so the MC can be done via the production through the manufacturers to the retailers based on relatively accurate forecast due to market feedback. The whole SC is now for a product family. In such a way, the MC is done with a mass production cost. We tie the results in seven WSs with the different phases in PLC (Figure. 4.25). This indicates that even for a new, customized product, as it enters

the growth phase when the demand is relatively stable and predictable, the customization can be done in advanced to the customers' order-intake, until the product goes to the decline phase. Therefore, even for a product with customized solutions, we do not have to adopt the MTO production mode all the time, because in the growth and maturity phase, we can finish the production of the mass-customized product ahead of the orders with relatively low risk.

### ***Verification and Discussion***

To verify our conclusions and attain insight on the relation between CODP candidate variables and the achieved value of the goals, we perform sensitivity analysis of the six CODP candidate locations. In each weight scenario, we fix the CODP in each of the six candidate locations and obtain the achieved value of the goals. We show the comparison of the results with CODP fixed at each point with the *satisficing* results (Table 4.16) in Table 4.17. Under each WS, we show the *satisficing* result with an asterisk "\*" in the column named "*satisficing*." The percentage numbers in those *non-satisficing* lines are comparison with the *satisficing* ones. "-24%" means that it is 24% worse than the *satisficing* result under the same WS. In some scenarios, if we assign a candidate location as CODP, never can we get any feasible solution because the constraints always get violated. For example, in WS 1, if we set SRM as the CODP, which means the whole SC is in MTO production mode, we cannot satisfy all constraints. So, if we set profit as the only goal, given the resource that can be acquired in this SC, a whole MTO production mode is not recommend for this product.

From Table 4.17, we observe that we can never position CODP at two locations, SRM and SFG, because we cannot obtain any feasible solutions. Therefore, the product is not fit for a pour pull strategy. In WS 1, 4, 5, and 7, RFG as CODP cannot give feasible solutions, whereas in all the other WSs – WS 2, 3, and 6 – RFG is the *satisficing* solution. This means RFG is a sensitive node,

either gives the best or lose it all. This proves that doing weight sensitivity analysis is necessary so that we only pick up appoint as CODP when it gives qualified result in certain design preferences.

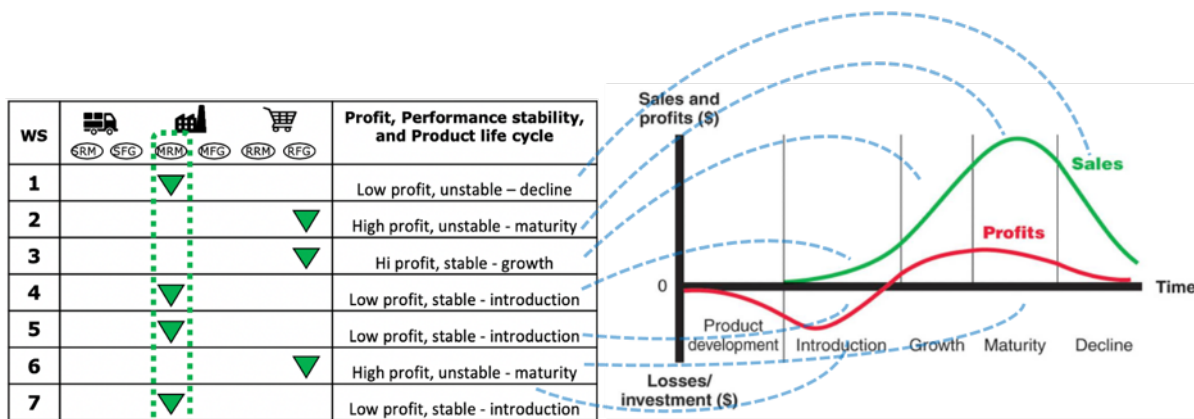
Among the three goals, profit is relatively sensitive to CODP migration because when we move CODP to *non-satisficing* nodes, the achieved values of profit get the most decreases. Service level does not change much as CODP migrates, nor does it change much as WS changes. In this sense, to reduce the computational complexity, we can remove the service level goal.

**Table 4. 18 Comparison of Satisficing Results with Results from Other CODP Candidate Locations**

WS	<i>Satisficing</i>	CODP	Profit	Service Level	Variance of Reliability
1		SRM	Infeasible		
1		SFG	Infeasible		
1	*	MRM	31584	99.13%	48.82%
1		MFG	-24%	0%	+0.25%
1		RRM	-13%	+0.1%	+85%
1		RFG	Infeasible		
2		SRM	Infeasible		
2		SFG	Infeasible		
2		MRM	Infeasible		
2		MFG	Infeasible		
2		RRM	Infeasible		
2	*	RFG	36946	99.21%	0.483%
3		SRM	Infeasible		
3		SFG	Infeasible		
3		MRM	Infeasible		
3		MFG	Infeasible		
3		RRM	Infeasible		
3	*	RFG	36657	98.25%	0.025%
4		SRM	Infeasible		
4		SFG	Infeasible		
4	*	MRM	31584	31584	31584
4		MFG	-24%	-24%	-24%
4		RRM	-13%	-13%	-13%
4		RFG	Infeasible		
5		SRM	Infeasible		
5		SFG	Infeasible		
5	*	MRM	31584	31584	31584
5		MFG	-24%	-24%	-24%



5		RRM	-13%	-13%	-13%
5		RFG	Infeasible		
6		SRM	Infeasible		
6		SFG	Infeasible		
6		MRM	Infeasible		
6		MFG	Infeasible		
6		RRM	Infeasible		
6	*	RFG	33813	99.23%	0.315%
7		SRM	Infeasible		
7		SFG	Infeasible		
7	*	MRM	31584	31584	31584
7		MFG	-24%	-24%	-24%
7		RRM	-13%	-13%	-13%
7		FRG	Infeasible		



**Figure 4. 25 CODP, Achieved Value of Goals in Different Phases of a Product Life Cycle – Managing Type I Uncertainty**

Among the six candidate locations, only two can be CODP in all seven WSs – MRM and RFG, so it simplifies the decision makers’ tasks. They need to focus on these two locations instead of all six locations. The responses of the preference change are all about switching CODP between MRM and RFG. Decision makers do not need to do one-time investment on fixed assets to make other locations as CODP.

Using the Formulation-Exploration framework, when changes take place in the SC, such as design preferences unexpectedly evolve (Type III uncertainty), cost budget cutting-down due to crises (Type II uncertainty), production/transportation cost or capacity change due to natural disasters

(Type I or IV uncertainty), etc., we can reformulate the problem and output *satisficing* solutions in a real-time manner. The switching of the phases of the PLC can be captured in time and the production strategy can be updated accordingly.

#### ***4.3.6 Closure of Test Problem II***

In this section, we propose a practical framework, Formulation-Exploration framework, for positioning CODP in SC networks to facilitate MC. We review the literature of CODP positioning and identify two major limitations – problematic assumptions on stochastic parameters and relying on model accuracy and we analyze the challenges in SC regarding CODP positioning. Also, we define the concept of the robust design to SC and categorize the uncertainties in the SC into four types as in Table 4.12. Based on the limitations and research gaps in the literature and the challenges in the CODP design, we raise a question and hypothesis. To answer the question, we propose the Formulation-Exploration framework to explore the solution space to exploit the potential of the SC. Using a test problem in automobile industry, we answer the question.

***Question (Q1.4) – How can designers improve the robustness (insensitivity) of the CODP design without sacrificing the performance of the SC to facilitate MC?***

Using the Formulation-Exploration framework, we explore the solution space by using representative design scenarios and explore the potential of boosting the performance of the system by adopting physical means, such as applying a new technology to cut down a certain cost. By doing this in iterations, we exploit all means that lead us reaching the most desirable *satisficing* given the resources on hand. With the Formulation-Exploration framework, we can iteratively determine and update the CODP, which facilitate providing products to satisfy individual customer’s requirements with near mass production efficiency. We define that the formulation-

exploration framework is a dynamic, open meta-design scaffold, on which new, customized modules can be added to manage both domain-dependent and domain-independent design problems. The current four incorporated methods – cDSP, adaptive linear programming, weight sensitivity analysis, and system capacity analysis are examples to launch our practice for *satisficing* design.

There are two major contributions of this work. First, the proposed Formulation-Exploration framework, allows designers to manage multiple conflicting goals by minimizing the distance between their targets and the achieved values. As different design scenarios that represent evolving preferences in various stages of a product life cycle (PLC) are explored, one can design the MC in different situations to have various flexibility. By exploring the tradeoffs between goals, appropriate design scenario can be adopted to each stage of a PLC. As the product enters different stages, a real-time switching of CODP facilitates mass customization.

Second, we provide a new perspective of SC design – improving the sustainability of a SC by enabling players to share the duty and risk in reliability. Instead of modeling the SC in a conventional way by determining the production volume or inventory level, we determine the reliability of the processes in the SC and minimizing their difference. We view the SC as a system and encourage the players in the system to share responsibility and boost the sustainability of the system by balancing the duty and the gains of each player. Multiple types of uncertainty are managed in this section by adding buffer to the physical model and keep removing bottleneck of the system.

There are two other contributions in this section. We apply the robust design concept in engineering into SC design and categorize the typical uncertainties in a SC into four types. Different modules in FME enable designers to manage different types of uncertainty in a SC. In

addition, the concept of CODP can be expanded in other types of complex systems, such as decision workflow design. The essential idea of adopting the CODP in complex-system design and operation is positioning the key node or several key nodes as the CODP(s) of the system, where the characteristic of the system activity changes from active preparation to order-driven. By identifying the requirements at the CODP, a designer can proceed the upstream and downstream processes. The benefit of having a CODP in a complex system is that by positioning and switching the CODP, one can ensure that the entire system maintains a sustainable and robust state, meanwhile maximize the reconfigurability and enable mass customization.

The Formulation-Exploration framework can be applied to engineering design problems. Designers can improve the model and manage uncertainties by removing the sensitive and improvable segments of the engineering-design system. By viewing the different segments of a system as players collaborating with one another while competing resource, we can improve the system performance by letting them share responsibilities and risks. The Formulation-Exploration framework is especially useful for continuous improvement and sustainability improvement of a system with conflicting goals, evolving design preferences and multiple players who seek equality and common interest.

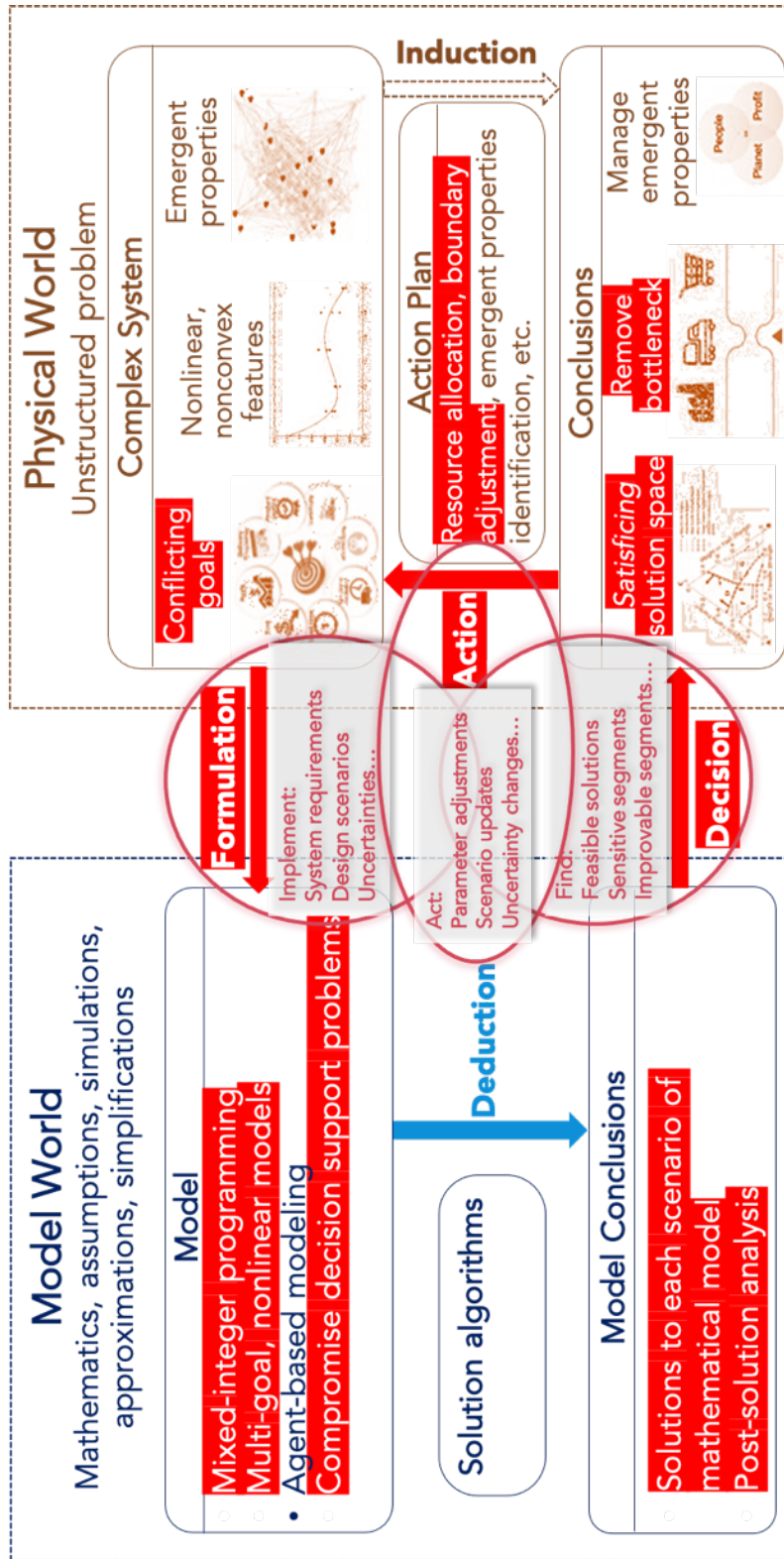
#### **4.4 Role of Chapter 4 in this Dissertation**

##### ***4.4.1 Summarizing How We Finish Task 1 – Connecting Formulation and Exploration***

When designing a complex system, especially in the early stage, designers may not have sufficient information and knowledge on the requirements or uncertainty in and around the system. Some mathematical relations among parameters and variables are based on designers' experience, heuristics, or preferences.

The essence of Specific Hypothesis 3, “*explore the sensitivity of the segments of the model boundary and improve accordingly*” is to refine the model formulation, use appropriate design scenarios, and manage typical uncertainties. Designers often need sufficient explorations and experiments to acquire data and interpret the data into knowledge on the model improvement actions. In this chapter, we standardize such exploration-interpretation-improvement procedure by incorporating typical uncertainties, such as the variation in the water inflow of the dam network and the uncertainty in product demand and supply in a supply chain, then identifying the sensitive and improvable segments of the model. By exploring the ways of removing the sensitive and improvable segments of the model, we ensure the model become less sensitive to scenario changes and uncertainties.

In other words, we strengthen the connections among the three processes in Figure 4.26, formulation, decision, and action. Without using the Formulation-Exploration framework, the three processes are relatively isolated, without information exchange. Even with designers’ effort of design improvement by passing through the information among them, the operations are not standardized nor generalized. In this dissertation, we establish the information exchange, knowledge awareness, and instructions sharing among the three processes and make it standardized; see Table 4.12 and 4.14.



**Figure 4. 26 The Methods and Procedures Involved in Formulation-Exploration – Establish the information exchange, knowledge awareness, and instructions sharing among the three highlighted processes**

#### 4.4.2 Summarizing How We Realize Type I & II Robust Design

For the dam network problem, Type I uncertainty is identified as the variation water inflow, which is managed by concretizing the typical cases as inflow scenarios (Table 4.7); Type II uncertainty is concretized as the design scenarios that represent the different priorities of different users' water demand (Table 4.5).

For the supply chain problem, Type I uncertainty is represented as the variation in product demand in different phases of the product life cycle (Figure 4.25); Type II uncertainty is reflected as the evolving preference of different system goals (Table 4.15).

By implementing the scenarios, identifying the segments to be improved, and improving the model formulation, Type I and Type II uncertainty is managed. In this way, we realize Type I & II robust design; see the summary in Table 4.19 as the closing remarks of Table 3.2 regarding the robust design realization and uncertainty management for Test Problems 1.1 and 1.2.

**Table 4. 19 Summary of Test Problems 1.1 & 1.2 regarding Type I&II Uncertainty Management**

RD Type	RDI-II		RDIII		RDIV	
	Method	M1: Formulation-Exploration Framework		M2: Adaptive Linear Programming with Parameter Learning (ALPPL)	M3: Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)	M4: Scenario Planning in Agent-Based Modeling
Chapter	Ch 4		Ch 5	Ch 6	Ch 7	
Uncertainty Test Problem	T1: Dam network	T2: Supply chain	T3: Hot rolling process chain	T4: Thermal system	T5: Promoting second-season farming	

Type I	<i>Uncertainty in timing and amount of inflow – Table 4.7</i>	<i>Uncertainty in demand side – Figure 4.25</i>	<i>Uncertainty in hyper parameter setting (Parameters in approximation algorithm)</i>	<i>Uncertainty in parameter setting in solution algorithm (Starting point of searching)</i>	<i>Uncertainty in price (Price of agriculture products)</i>
Type II	<i>Uncertainty in outflow (water release target) – Table 4.5</i>	<i>Uncertainty in supply side - Table 4.15</i>	<i>Uncertainty in user preferences</i>		<i>Promotion effort and timing</i>
Type III			<i>Uncertainty in model approximation due to heuristics in approximation</i>	<i>Uncertainty in model approximation (ways of combining multiple goals)</i>	
Type IV				<i>Uncertainty in using domain knowledge to simplify the model (fixing decision variables and selecting design scenarios)</i>	<i>Interventions that change the mathematical relation among promotion and result (developing local market)</i>

RD – robust design

M – method

EVe – empirical verification of the method

T – test problem

#### 4.4.3 Role of Chapter 4

In Chapter 4, given the frame of references on *satisficing* strategy, especially focusing on the exploration of the boundary of the model, which is an extension of the frame of references in Chapter 2. A method, the Three-Step Exploration Method, and a framework that encompasses the method, Formulation-Exploration framework, are proposed to explore boundary of the model. Two test problems, a continuous system – dam network, and a coupled decision-making problem – design the customer-order-decoupling-point (CODP) of a supply chain, are used to verify the proposed methods. It is proved that using the Three-Step Exploration Method and its advanced framework, Formulation-Exploration framework, the system can be evolved to be relatively insensitive to the uncertainties happen to the boundary of the system, without sacrificing the system performance (the achievement of the goals). Research Question 1 is addressed.



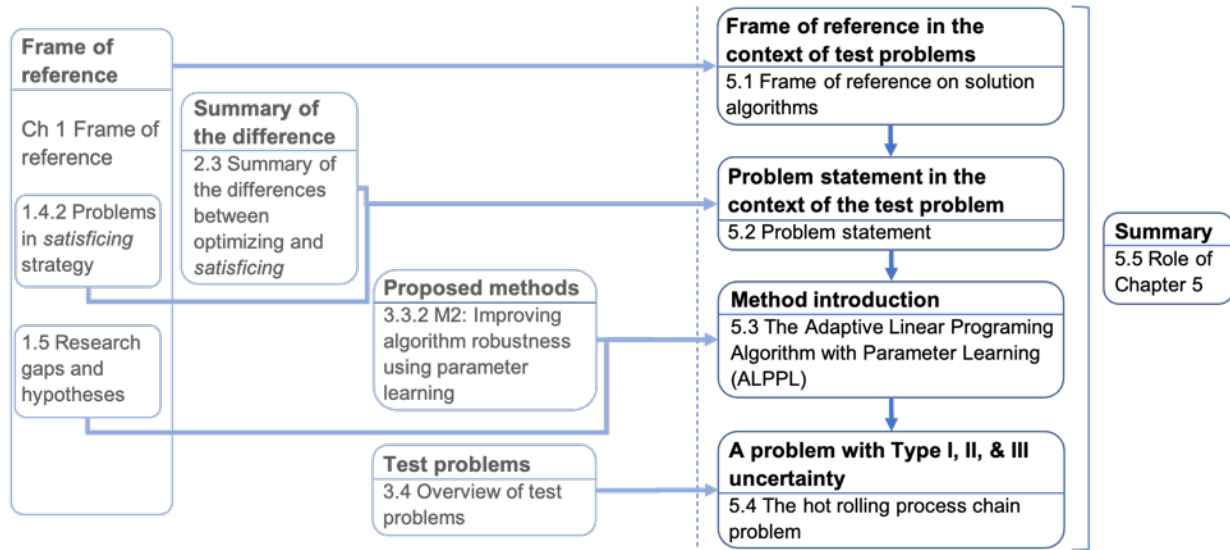
## CHAPTER 5 TYPE I, II, & III ROBUST DESIGN THROUGH IMPROVING APPROXIMATION

### – ADAPTIVE LINEAR PROGRAMING ALGORITHM WITH PARAMETER LEARNING (ALPPL)

#### ***The new knowledge in Chapter 5:***

*An improved algorithm that incorporates parameter learning to realize model evolution by improving the determination of the critical parameter in the approximation and solution algorithm – Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL)*

In Chapter 5, see Figure 5.1: in Section 5.1, the reference on solution algorithms is framed, which is an extension of Section 1.2 model strategies and their foci; in Section 5.2, the problem statement regarding the research gaps in the approximation and solution algorithms in *satisficing* strategy is discussed, which is a specification based on Section 1.4.2 and Section 2.3; in Section 5.3, based on the research gaps described in Section 1.5 and the Method 2 proposed in Section 3.3.2, we introduce the Adaptive Linear Programming algorithm with Parameter Learning (ALPPL) in details; in Section 5.4, the ALPPL is applied to the hot rod process chain problem for identifying the solution space relatively insensitive to parameter setting and model structure variation caused by problematic heuristics; in Section 5.5, summarized the role of Chapter 5.



**Figure 5. 1 Organization of Chapter 5**

The plan of specifying and answering Research Question 2 in the context of the test problems is shown in Table 5.1. In Chapter 5, the Proposed Method 2 (M2), Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL), is empirically verified (EVe2) using Test Problem 2 (T2), the cooling stage of the hot rod process chain. Research Question 2 (RQ2) is specified into the context of the test problem (SQT2) and answered (AQ2) by testifying M2. The empirical validation and theoretical validation are in Chapters 8 and 9.

**Table 5. 1 Plan of Specifying Research Question 2 (RQ2) and Empirically Verifying the Adaptive Linear Algorithm with Parameter Learning (ALPPL) (M2)**

Chapter	Ch1	Ch2	Ch3	Ch 4-7			Ch8	Ch9
				Ch4	Ch5	Ch6-7		
Actions	RG H	RD RQ SH	TVe M	EVe1 SQT1 AQ1	<i>EVe2: use an engineering-design problem with uncertainties in parameters and metaheuristics to verify SH2 and demonstrate M2.</i> <i>SQT2: What is the mathematics in the design method that allows approximation improvements in the realization of complex systems?</i>	EVe SQT AQ	CQ EVa	TE

					<i>AQ2: by incorporating parameter learning in the approximation algorithm, designers can learn the association between parameter and approximation performance so they can guarantee the approximation performance through updating the heuristics in parameter settings.</i>			
<b>Nomenclature</b>	RG – give research gaps							
	H – give hypotheses							
	RD – tie to robust design							
	RQ – pose research questions							
	SH – specify hypotheses							
	TVE – theoretically verify hypotheses							
	M – introduce methods							
	EVE – empirically verify hypotheses							
	SQT – specify research questions in the context of test problems							
	AQ – answer research questions							
	CQ – closure the answers to research questions							
EVA – empirically validate hypotheses								
TE – theoretically extend the research								

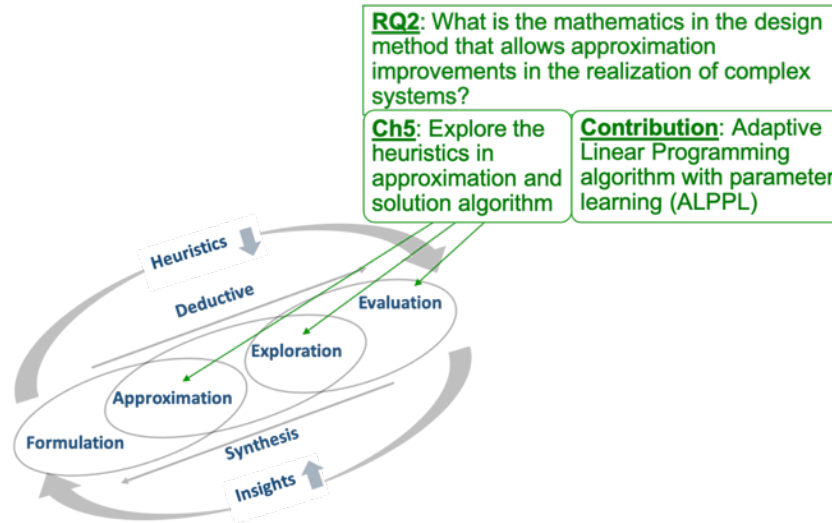
In this chapter, the Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL) is proposed and tested by using an engineering-design problem – the hot rolling process chain. The Research Question 2 (RQ2) is answered.

***RQ2: What is the method to evolve model to update metaheuristics?***

In this chapter, the research question is specified into the context of the test problem.

***SQT2: What is the mathematics in the design method that allows approximation improvements in the realization of complex systems?***

To answer RQ2, interactions between the approximation, exploration, and evaluation of a design problem should be studied and the mechanisms of information sharing and intervention between the three stages are established; see Figure 5.2.



**Figure 5. 2 Specified Research Question 2 and the Relevant Stages to be Connected in Design Evolution Cycle**

### Glossary (mainly applied in Chapter 5)

**ALP** = Adaptive Linear Programming

**ALPPL** = Adaptive Linear Programming with Parameter Learning

**DEI** = Desired Range of Evaluation Index

**DSP** = Decision Support Problem

**EI/EIs** = Evaluation Index/Indices

**RMC** = Reduced Move Coefficient

**Completion of a Goal (or Fulfillment of a Goal)**: the degree of completion of a goal versus its target.

**RMC Performance**: the quality of the solution regarding certain criteria for an RMC value

**Quality of a Solution**: the comprehensive level of performance of a solution based on certain criteria, such as the fulfillment of the goals, the number of active constraints, the number of accumulated constraints, etc.

**Active Constraints**: when plugging a solution into a constraint, its left-hand-side value equals its right-hand-side value, then the constraint is an active constraint.

**Accumulated constraints**: when linearizing a nonlinear constraint, multiple linear constraints are acquired, then all the linear constraints are accumulated to replace the nonlinear constraints. Those linear constraints are accumulated constraints or accumulated linear constraints.

### Nomenclature (mainly applied in Chapter 5)

$X$   $X = \{x_1, x_2, \dots, x_n\}$ . System variables (decision variables).

$G_k$  The  $k^{\text{th}}$  goal of a K-goal compromise formulation.

$T_k$  The target value of the  $k^{\text{th}}$  goal of a K-goal compromise formulation.

$d_k^-, d_k^+$  Negative deviation variables and positive deviation variables of Goal k.  $d_k^-$  and  $d_k^+$  are the under-achievement and over-achievement of Goal k. The formulation of Goal k is:  $\frac{G_k}{T_k} + d_k^- - d_k^+ - 1 = 0$ .  $d_k^- \cdot$

$d_k^+ \equiv 0$ .

$\frac{G_k}{T_k}$  Fulfillment of Goal k – measures how well Goal k achieves its target.

$W$  Weight Scenarios – The different weight vectors used to combine multiple goals to represent various design preferences linearly.  
 $W_k^s$  Satisficing Weight set for Goal  $k$  – The range of weight values that satisfy Goal  $k$ .  
 $W^s$   $W^s = \bigcap_{k \in K} W_k^s$ . Satisficing Weight Set – The set of weight scenarios that satisfy all  $K$  goals.  
 $Z$   $Z = \sum_{k \in K} W_k \cdot (d_k^- + d_k^+)$ . Deviation function  $Z$  is the weighted sum of deviation variables of all goals. By minimizing  $Z$ , goals are achieved in a compromise way.  $Z$  measures how much the goals do not achieve their targets.  
 $P^0$  The original problem – may be nonlinear.  
 $P_i^L$  The approximate linear problem in the  $i$ th iteration.  
 $X_i^*$   $X_i^* = \{x_{i1}^*, x_{i2}^*, \dots, x_{in}^*\}$ . The solution of the  $i$ th iteration.  
 $X_i^0$   $X_i^0 = \{x_{i1}^0, x_{i2}^0, \dots, x_{in}^0\}$ . The starting point of the  $i$ th iteration. The original problem  $P^0$  is linearized at  $X_i^0$  in the  $i$ th iteration. See Figure 5.4 and Equation 5.1-5.3.  $X_i^0$  is determined using Equation 5.4.  
 $AOC$  Active Original Constraints – The inequality constraints with zero slack or the surplus when plugging in a solution. See Figure 5.10 (a).  
 $AB$  Active Bounds – For a solution, if the value of a variable is on its upper or lower bound, then the bound is an active bound.  
 $J$  The set of nonlinear constraints of  $P^0$ .  
 $NF_j$  The  $j$ th nonlinear constraint of  $P^0$ ,  $j \in J$ .  $NF_j$  maintains the same in every iteration. See Figure 5.4 and 5.5.  
 $NF_{i,j}'$  The approximated second-order parabolic of  $NF_j$  using the second-order derivatives at  $X_i^0$ . See Figure 5.5 and Equation 5.1 and 5.2.  
 $LF_{i,j}$  The approximated linear constraint of  $NF_j$  in the  $i$ th iteration.  $LF_{i,j}$  is a part of  $P_i^L$ . See Figure 5.4 and 5.5, and Equation 5.1-5.6.  
 $ACC$  Accumulated Constraints. When the convexity of a nonlinear constraint  $NF_j$  is relatively large – greater than 0.015, approximated linear constraints from multiple iterations are accumulated to replace  $NF_j$ . See Figure 5.4 and Equation 5.7.

In this chapter, an approximation algorithm – the Adaptive Linear Programming algorithm with parameter learning (ALPPL) is applied to manage a test problem that is parameter-sensitive. Further, different issues are addressed in the approximation of complex-system design: (1) enabling scientific determination and updating of critical parameters to manage approximation efficiency, (2) meeting the requirement on problem fidelity, (3) meeting the requirement on computational complexity, and (4) managing uncertainty. The efficacy of the method is demonstrated using a hot rod rolling problem as Test Problem 3.

The establishment of the context is in Section 5.1. The answer to the Research Question 2 is the mechanism and process of improving the ALP by incorporating parameter learning, see Section 5.2 and 5.3. The empirical structural validity of the method is presented in Sections 5.4.

## 5.1 Frame of Reference on Solution Algorithms

Solution algorithms for solving complex problems fall into two categories (Table 5.2): formulate a problem exactly and solve it approximately or approximate a problem and solve it exactly. Examples of solution algorithms in the first category are gradient-based methods (Williams and Zipser 1995), pattern search methods (Rios and Sahinidis 2013), and penalty function methods (Viswanathan and Grossmann 1990). Using these methods may lead to relatively higher computational complexity, and the solution is usually not on the vertex of the feasible space. In contrast, the methods in the second category, such as Sequential Linear Programming, allow designers to obtain solutions on one or several vertices. This enables designers to use duality embodied in linear programming to explore the solution space without performing interior-point searches using the methods in Category I (Mistree, Hughes et al. 1981, Mistree and Kamal 1985). The dual problem is worthy of study, especially for engineering-design problems because, in engineering problems, the number of constraints is often more than the dimensionality of the problems (Mistree, Hughes et al. 1981). Further, methods in Category II facilitate rapid identification of robust solutions<sup>18</sup>.

In this chapter, our focus is Category II because we deal with engineering design problems to obtain useful, practical, but not necessarily optimal solutions.

The Adaptive Linear Programming (ALP) algorithm proposed by Mistree et al. in (Reddy 1992) and described in detail in (Mistree, Hughes et al. 1993) falls into Category II. The reported use of the ALP includes the design of ships (Smith, Kamal et al. 1987), damage tolerant structural and

---

<sup>18</sup> Solutions which are relatively insensitive to approximations made to make the solution to the problem tractable.

mechanical systems (Mistree, Hughes et al. 1993), design of aircraft, mechanisms, thermal energy systems (Smith, Milisavljevic et al. 2014), composite materials, and the concurrent design of multi-scale, multi-functional materials and products (Nellippallil, Rangaraj et al. 2018). A detailed set of early references to these applications is presented in (Mistree, Muster et al. 1990). However, like most methods in Category II, heuristics are used to approximate and search solutions to the design problem. One such heuristic is the reduced move coefficient (RMC), which is a parameter that helps determine the starting point for approximation for every new iteration. It determines the step size of the search (Mistree, Hughes et al. 1981) and minimizes the oscillation of the solution from one synthesis cycle (Figure 5.4) to the next (Mistree, Hughes et al. 1993). Oscillation results in poor convergence; therefore, we use the RMC to get a more gradual change from a set of system variables from one synthesis cycle to another. Nevertheless, the RMC value is set by the designer without any knowledge on the connection between the RMC value and the quality of the solution or the solution improvement. This limitation is addressed in this chapter by applying parameter learning to improve the Adaptive Linear Programming (ALP) algorithm by incorporating parameter learning (ALPPL).

**Table 5. 2 Advantages and disadvantages of the two categories of solution algorithms**

#	Category	Example Methods	Advantages	Disadvantages
<b>I</b>	Formulate a problem exactly and solve it approximately	Gradient-based methods, pattern search methods, penalty function methods, etc.	- Maintaining a relatively accurate model along with the solution search (given the information that the designer has on hand).	- The solution is still an approximate, inaccurate one; - Heuristics are used in solution algorithms, which may result in premature convergence or unnecessarily high computational complexity.
<b>II</b>	Approximate a problem and	Adaptive linear programming,	- Solutions are on the vertices of the approximated problem so	- Heuristics are used in approximation

solve it exactly	sequential linear programming, etc.	the problem with a large number of constraints can be solved relatively easily by solving the dual; - The approximation accuracy can be improved by accumulating the linearized constraints along with iterations.	algorithms, which may result in inaccurate approximation.
------------------	-------------------------------------	---	---

In Section 5.2, we introduce the ALP and the limitation of the algorithm regarding the RMC determination, and the hypothesis for improving the algorithm; in Section 5.3, we propose a method – ALP with parameter learning (ALPPL) to verify the hypothesis. In Section 5.4, we use an engineering design problem – the cooling stage of the hot rod rolling process chain, to establish the efficacy of the ALPPL. In Section 5.5, we summarize the contributions and limitations and suggest possible directions for future work.

**5.2 Problem Statement – Limitations of the ALP regarding Parameter Determination**

Section 5.2 is an extension of Section 1.4.2. The more detailed description of the ALP algorithm is in Section 5.2.

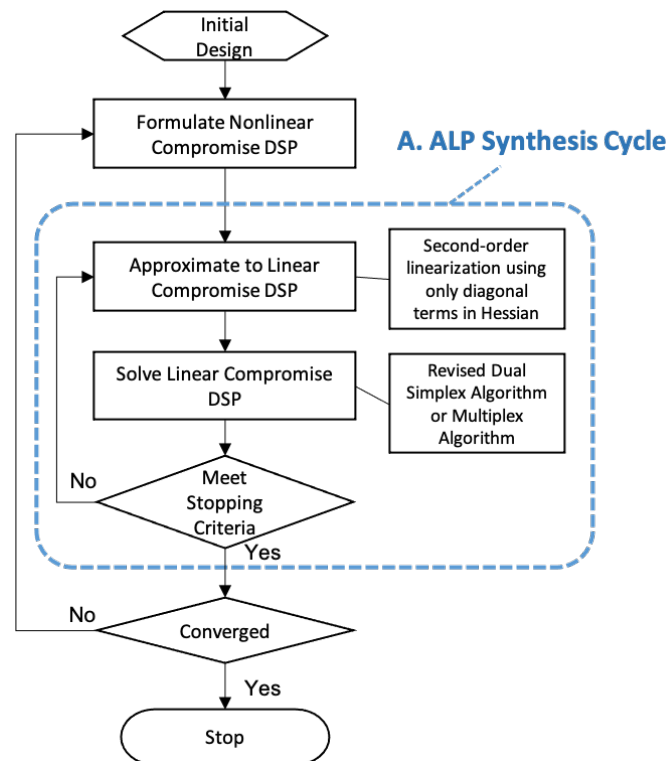
**5.2.1 Adaptive Linear Programming (ALP) Algorithm**

Using the Adaptive Linear Programming (ALP) algorithm, one can approximate and solve nonlinear problems. The procedures of the ALP algorithm are shown in Figure 5.3. The ALP is implemented in DSIDES (Ming, Nellippallil et al.), a decision support system. In Figure 5.3, the processes in the dotted rounded rectangle A are executed in a synthesis cycle. In the synthesis cycle, continuous improvement (or repeated modification) takes place, that includes two parts – approximation of the nonlinear problem and solving the approximated linear problem via the



revised dual simplex algorithm. If the stopping criteria<sup>19</sup> are met, then the solution with the best value of the deviation function  $Z$  is returned as the *satisficing* solution.

The ALP incorporates a local approximation algorithm (Barthelemy and Haftka 1993), in which a secant plane of the paraboloid (with the second-order derivatives at a point as the coefficients) is used to replace the original high-order nonlinear function.



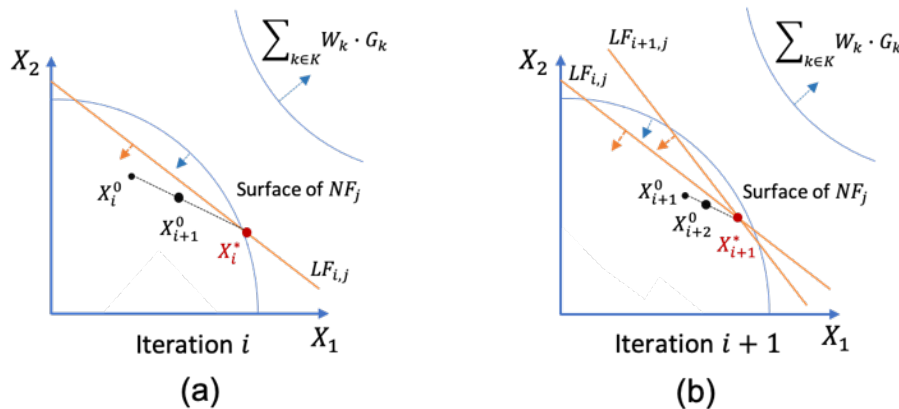
**Figure 5. 3 The ALP Algorithm**

In Figure 5.4, illustrated the projection of an n-dimension problem onto a two-dimension plane and how the design problem is approximated and solved in two iterations using the ALP. In the

<sup>19</sup> The stopping criteria include maximum number of synthesis cycle (NITER), desired stationarity of deviation function (EPSZ), desired stationarity of system variables (EPSX), and desired stationarity of the RMC (threshold value  $\epsilon$ ) Mistree, F. and S. Kamal (1985). DSIDES: Decision Support in the Design of Engineering Systems, University of Houston.

. The program is stopped if any one of the four stopping criteria is met.

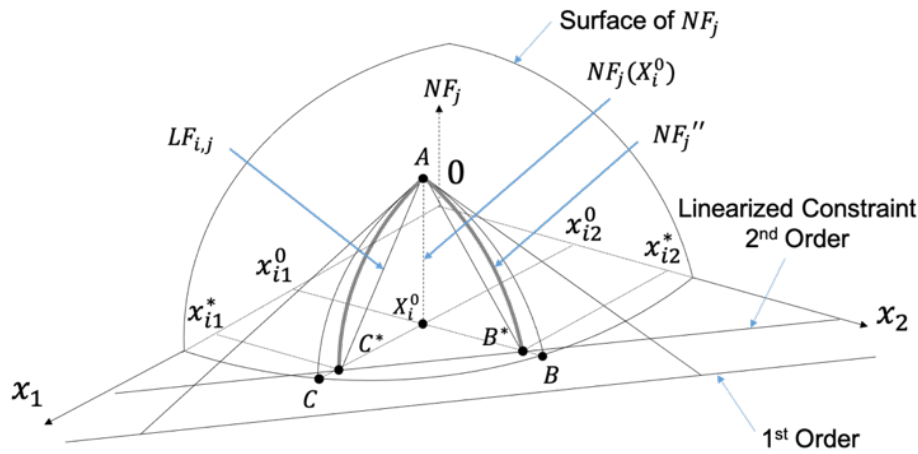
first iteration, the starting point,  $X_0^0$  (user-defined), may not be in the feasible region, The Hook-Jeeves pattern search algorithm is used to bring  $X_0^0$  into the feasible region, as  $X_1^0$ , so it enters the first iteration, and the problem is linearized at  $X_1^0$ . In Iteration  $i$ , Figure 5.4 (a), a nonlinear constraint<sup>20</sup>,  $NF_j$ , is approximated at  $X_i^0$ , so an approximated constraint  $LF_{i,j}$  is obtained. With the revised simplex dual algorithm, a solution  $X_i^*$  is returned. With the reduced move coefficient (RMC), we get the point  $X_{i+1}^0$ , a point between  $X_i^0$  and  $X_i^*$ , as the starting point of Iteration  $i + 1$ . In Iteration  $i + 1$ , Figure 5.4 (b), the approximated linear constraints of both iterations,  $LF_{i,j}$  and  $LF_{i+1,j}$  are accumulated, and a solution  $X_{i+1}^*$  is returned and the starting point of Iteration  $i + 2$ . The approximation continues until one of the stopping criteria is met – either the solution points or the fulfillment of the goals maintains in a stationary range, or, the total iterations reach an upper limit.



**Figure 5. 4 The Approximation and Obtained Solution using the ALP in Two Iterations**

<sup>20</sup> If  $NF_j$  is an inequality constraint,  $NF_j \geq 0$ , then the curve denoted as  $NF_j$  in Fig. 2 is the surface of  $NF_j$ ; if  $NF_j$  is an equality constraint,  $NF_j = 0$ , then the curve denoted as  $NF_j$  in Fig. 2 is  $NF_j$ .

To illustrate the approximation, we use a three-dimensional graph – Figure 5.5. The approximation takes place in two steps. First,  $NF_j$  (Paraboloid  $ABC$  in Figure 5.5) is approximated to a paraboloid  $NF_{i,j}''$  (Paraboloid  $AB^*C^*$  in Figure 5.5) by using the diagonal terms of its Hessian matrix at  $X_i^0$  as the coefficient. Then,  $NF_j''$  is approximated to a secant Plane  $LF_{i,j}$  (Plane  $AB^*C^*$  in Figure 5.5). The calculations of  $NF_j''$  and  $LF_{i,j}$  are given as follows.



**Figure 5. 5 The Original Nonlinear Constraint, the Second-Order Paraboloid, and the Secant Plane [4]**

$NF_{i,j}''$  is obtained by using only the second-order full derivatives at  $X_i^0$ , shown in Equation 5.1, because the second-order partial derivatives are proved to have very limited impact on the gradient (Mistree, Hughes et al. 1981).

$$NF_{i,j}'' = NF_j(X_i^0) + \sum_{p=1}^n (x_{ip} - x_{ip}^0) \left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0 + \frac{1}{2} \sum_{p=1}^n (x_{ip} - x_{ip}^0)^2 \left( \frac{\partial^2 NF_j}{\partial x_{ip}^2} \right)_0$$

**Equation 5. 1**

Then, from Equation (1), for the  $p^{th}$  dimension, the quadratic to be solved to determine  $(x_{ip} - x_{ip}^0)$  is Equation (2).

$$NF_j(X_i^0) + (x_{ip} - x_{ip}^0) \left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0 + \frac{1}{2} (x_{ip} - x_{ip}^0)^2 \left( \frac{\partial^2 NF_j}{\partial x_{ip}^2} \right)_0 = 0 \quad \text{Equation 5.2}$$

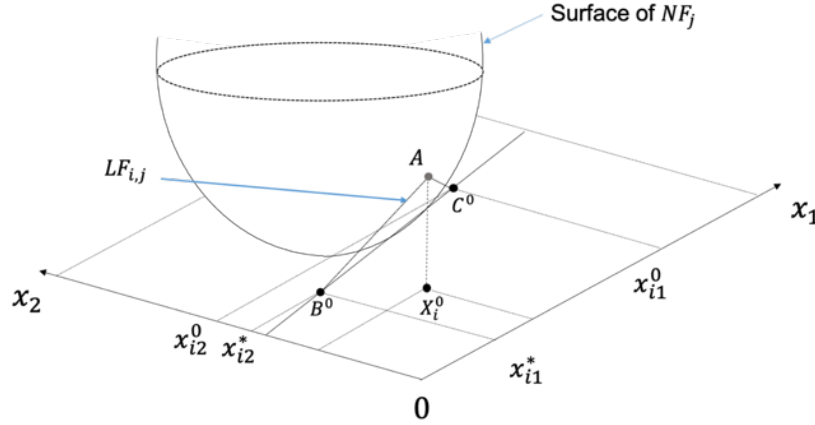
If Equation 5.2 has real roots, by solving Equation 5.2 and selecting the root between Equation 5.3 and Equation 5.4 with the smaller absolute value for each dimension, we obtain the intersection that is closer to the paraboloid in each dimension, such as  $B^*$  and  $C^*$ .

$$\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^* = \frac{-NF_j(X_i^0) \left( \frac{\partial^2 NF_j}{\partial x_{ip}^2} \right)_0}{-\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0 - \sqrt{\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^2 - 2NF_j(X_i^0) \left( \frac{\partial^2 NF_j}{\partial x_{ip}^2} \right)_0}} \quad \text{Equation 5.3}$$

$$\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^* = \frac{-NF_j(X_i^0) \left( \frac{\partial^2 NF_j}{\partial x_{ip}^2} \right)_0}{-\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0 + \sqrt{\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^2 - 2NF_j(X_i^0) \left( \frac{\partial^2 NF_j}{\partial x_{ip}^2} \right)_0}} \quad \text{Equation 5.4}$$

If Equation 5.2 has no real roots, for example, as the case shown in Figure 5.6,  $NF_j''$  does not have intersect with Plane  $x_1x_2$ , then the first-order derivative function at  $X_i^0$  is used, so  $\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^*$  is estimated by the tangent at  $x_{ip}^0$ , as Equation 5.5.

$$\left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^* = \left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0 \quad \text{Equation 5.5}$$



**Figure 5. 6 When the Second-Order Paraboloid Has No Intersection with Plane  $x_1x_2$ , The First-Order Tangent is Used to Approximate  $NF_j$**

Based on the intersections in each dimension, such as  $B^*$  and  $C^*$ , we can calculate  $LF_{i,j}$  using Equation 5.6.

$$LF_{i,j} = \sum_{p=1}^n x_{ip} \left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^* - \left( \sum_{p=1}^n x_{ip}^0 \left( \frac{\partial NF_j}{\partial x_{ip}} \right)_0^* - NF_j(X_i^0) \right) \quad \text{Equation 5.6.}$$

If the convexity of  $NF_j$  is relatively large – greater than 0.015 at any starting point in and before the  $i^{\text{th}}$  iteration, we use the accumulated constraints  $\cup_{q \leq i} LF_{q,j}$  to replace  $NF_j$ . See Equation 5.7.

If the convexity of  $NF_j$  at all starting point in and before the  $i^{\text{th}}$  iteration is less than or equal to 0.015, we use the single linear constraint in the  $i^{\text{th}}$  iteration to replace  $NF_j$ . See Equation 5.8.

### Constraint Accumulation Algorithm

In the  $i^{\text{th}}$  iteration,

**for** every  $j$  in  $J$

**if**  $\max \left\{ \left[ \text{conv}(NF_j) \right]_{x_q^0} \right\}_{q \leq i} > 0.015$

$$P_i^L = (P^0 \setminus \{NF_j\}) \cup \{ \cup_{q \leq i} LF_{q,j} \} \quad \text{Equation 5.7}$$

**Else**

$$P_i^L = (P^0 \setminus \{NF_j\}) \cup \{LF_{i,j}\} \quad \text{Equation 5.8}$$

Then, the revised simplex dual algorithm is applied to solve the linear problem  $P_i^L$ , so a solution  $X_i^*$  is obtained.  $X_{i+1}^0$ , as a point between the starting point  $X_i^0$  and the solution  $X_i^*$ , becomes the starting point of the next iteration. The Reduced Move Coefficient (RMC) is used to determine  $X_{i+1}^0$ . See Equation 5.9.

$$X_{i+1}^0 = X_i^0 + RMC \cdot (X_i^* - X_i^0) \quad \text{Equation 5.9}$$

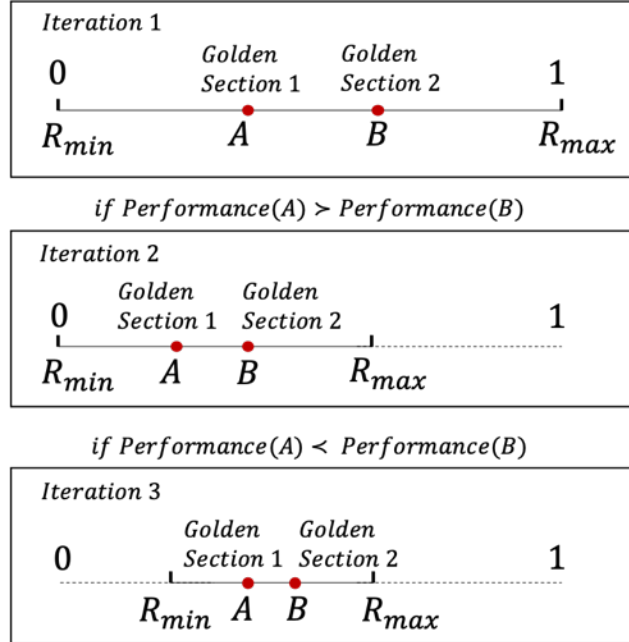
### 5.2.2 Reduced Move Coefficient (RMC)

The value<sup>21</sup> of RMC is in  $[0,1]$ . In the ALP, the RMC is either set by the designer as a fixed value or updated using the golden section search algorithm. In (Reddy 1992), the authors recommend setting 0.5 to the RMC as a fixed value through all iterations based on the observations from around 1100 experiments conducted by them in the seventies. However, 0.5 may not be the best value for every design problem. Therefore, the golden section search algorithm was added by Reddy et al. (Reddy 1992) to improve the RMC determination, in the sense of using metaheuristics to explore the solution space. Users can choose whether to set the RMC as a fixed value or to trigger the golden section search algorithm to allow the RMC to be adapt to new situations.

The use of golden section search reduces the range of RMC by cutting off the region with undesired deviation function value or bigger violation of the constraints, till the range is smaller than a threshold  $\epsilon$ , and at last return the best solution. The range updating mechanism is illustrated in Figure 5.7.

---

<sup>21</sup> The RMC does not have to be in  $[0, 1]$ . We set its range as  $[0, 1]$  because we want to explore the region between  $X_i^0$  and  $X_i^*$  in case the approximation fidelity in this region is poor as we assume this is a relatively important region. However, it is fine to have  $RMC < 0$  or  $RMC > 1$  so that designers may explore the region out of the  $[X_i^0, X_i^*]$  to avoid being stuck in local *satisficing*. Yet as in this section we deal with nonlinear problems, there is no hassle of local *satisficing*, so we bound the RMC in  $[0, 1]$  to stay focused on this relatively important region.



**Figure 5. 7 The Golden Section Search for The RMC in the ALP**

Two factors, the deviation function value and the violation value of constraints, are considered to evaluate the performance. In the  $(i - 1)^{th}$  iteration, two golden section points,  $A$  and  $B$ , are used to acquire two starting points of the  $i^{th}$  iteration,  $X_i^0(A)$  and  $X_i^0(B)$ . In the  $i^{th}$  iteration,  $P^0$  is approximated at  $X_i^0(A)$ , and  $X_i^0(B)$  and two solutions  $X_i^*(A)$  and  $X_i^*(B)$  are obtained by solving the linear problems using revised dual simplex, with which, we compute i) the violation values of constraints  $V(X_i^*(A))$  and  $V(X_i^*(B))$ , by adding all the violation values of the constraints, and ii) the values of the deviation functions,  $Z(X_i^*(A))$  and  $Z(X_i^*(B))$ , by plugging in the solutions. Then, the decisions on how to update the RMC range are made based on the four values. The performance evaluation rules and the RMC updating rules are given as follows.

### **RMC Range Updating using Golden Section Search**

- if**  $V(X_i^*(A)) == 0$  and  $V(X_i^*(B)) == 0$
- if**  $Z(X_i^*(A)) \leq Z(X_i^*(B))$
- $Performance(A) \geq Performance(B)$

```

    else
        Performance(A) < Performance(B)
else
    if    V(Xi*(A)) ≤ V(Xi*(B))
        Performance(A) ≥ Performance(B)
    else
        Performance(A) < Performance(B)
if    Performance(A) ≥ Performance(B)
    Rmax = B
else
    Rmin = A

```

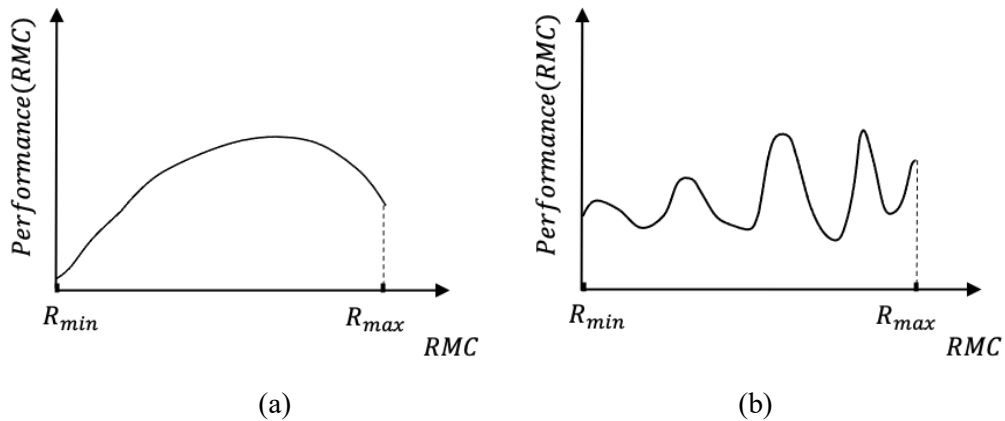
In this way, the RMC range is reduced through iteration, until the range is smaller than a threshold,  $R_{max} - R_{min} < \varepsilon$ , when we end the RMC search and return the solution  $X^*$  with the best performance.

### ***5.2.3 Limitations of the ALP Regarding the Determination of the RMC***

With golden section search, the desired sub-range of RMC may be missed, since this is based on two assumptions:

First, the approximation is a continuous function of the RMC and has no more than one inflection point in the range of RMC. In other words, the performance function of the RMC is either convex or concave, as shown in Figure 5.8 (a).





**Figure 5. 8 Possible Patterns of the Performance of the RMC in a Sub-Range**

However, for a number of design problems, the performance function is neither convex nor concave. It is fluctuating without any pattern, as in Figure 5.8 (b). For the fluctuating case, the golden section search may miss a sub-range of the RMC with desired performance.

Second, the criterion to evaluate the approximation performance is oversimplified – only the values of deviation function and the constraint violation are considered. Other critical aspects that matter to engineering design problems are not considered, such as the robustness of the results to uncertainties and the approximation accuracy.

Given the above two limitations, in this section, we propose to use parameter learning to improve the ALP algorithm learning the relation between the RMC value and the quality of the solutions and use it to wisely determine the RMC value so as to improve the robustness of the approximation algorithm to the parameter.

#### ***5.2.4 Hypothesis of Improving the ALP***

It is hypothesized that by incorporating parameter learning in the ALP algorithm, we can identify the range of RMC with more desired performance, that is returning relatively more stable results

regarding the goal achievement and the insensitivity of the solution to uncertainties. So, we need to define what is desired performance, then capture the relation between the RMC and the approximate performance and use such relation to tune the appropriate RMC value. There are three steps to verify the hypothesis.

*Step 1. Identifying the criteria to evaluate the approximation performance.*

*Step 2. Based on the identified criteria, developing evaluation indices (EIs) that help quantify the approximation performance regarding RMC values.*

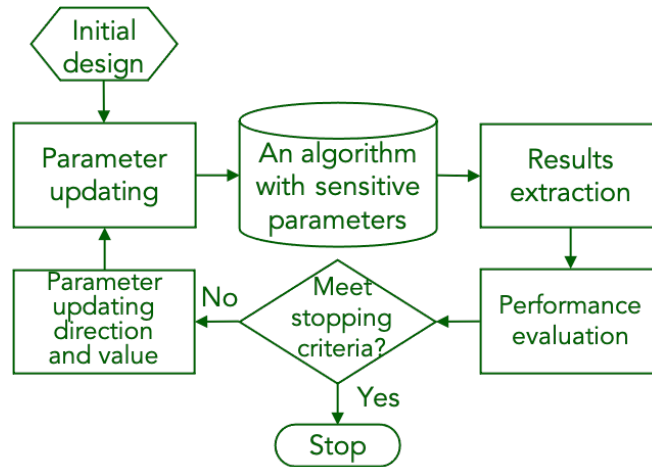
*Step 3. Learning the desired range of each EI (DEI) and tuning the RMC to give results falling in the DEI.*

In Section 5.3, an improved algorithm is proposed – the ALP with parameter learning (ALPPL), to realize the three steps and verify the hypothesis.

### **5.3 The Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL)**

In this section, we work out the three steps proposed in Section 5.2.4 to make improvements to the adaptive linear programming (ALP) algorithm.

Given these drawbacks in the golden section search, we propose the adaptive linear programming algorithm with parameter learning (ALPPL). See Figure 5.9. We embed the ALP algorithm into a loop, incorporating the parameter updating, results extraction, performance evaluation, and feedback on parameter updating. The loop is ALPPL. In this chapter, the parameter is specifically referred to the RMC.



**Figure 5. 9 The Concept of Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL)**

### ***5.3.1 Step 1 – Identify the Criteria for Evaluating the Quality of a Solution***

In the RMC search method currently used in the ALP, the golden section search, two criteria are considered – the fulfillment of the goals and the feasibility of the solutions. In ALPPL, we build upon them.

Criterion 1 – fulfillment of the goals.

In engineering design problems, the fulfillment of the goals is a basic criterion to assess whether the potential of the system has been explored relatively sufficiently (Jiang, Wang et al. 2012), so we keep it as a criterion in ALPPL.

Criterion 2 – the robustness of the solution.

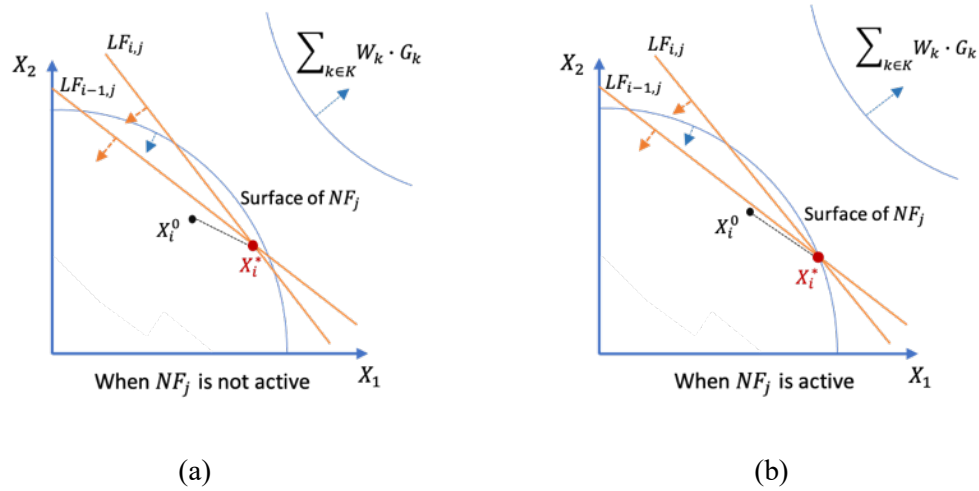
When we discuss the feasibility of the solutions, we only ensure that the solution is feasible in a deterministic situation but cannot ensure that the solution maintains feasible under uncertainties.

As all models are approximations of reality (Box and Draper 1987), we cannot capture all the information in reality perfectly in a model. We need to manage the unexpected uncertainty by making the solutions relatively insensitive to the errors and incompleteness of the model. So, we

extend the notion of “feasibility” to “robustness,” which means the solutions should be feasible under multiple design scenarios and robust to some model errors and uncertainties, such as wrong parameter values, inaccurate equations, or unstable functional relationship between parameters and variables. In this chapter, we define the robustness of a solution as its insensitivity to model errors and uncertainties, and we assume all errors and uncertainties result in the boundary change, specifically the change of the right-hand-side value of each constraint, therefore, the robustness of the solution can be measured by how far it is away from the boundary. The robustness of the solution and the fulfillment of the goals often contradict with each other, as an interior solution (a solution that is away from the boundary) often makes the goals less fulfilled than a boundary solution does, hence, we take both criteria into considerations and make decisions based on their tradeoffs. In Figure 5.10, we show two solutions with different levels of robustness. We take the  $X_i^*$  in Figure 5.10 (a) as a robust solution (relatively insensitive solution) and take the  $X_i^*$  in Figure 5.10 (b) as a sensitive solution. The solution in Figure 5.10 (a) is not on the boundary of the feasible space, so it is relatively insensitive (or more robust) to errors of the model or variations.  $NF_j$  in Figure 5.10 (a) is not an active constraint. On the contrary, the solution in Figure 5.10 (b) is on the boundary of the feasible space bounded by  $NF_j$ , so it is sensitive (or less robust) to errors or variations of the constraint.  $NF_j$  in Figure 5.10 (b) is an active constraint. The approximation does have an impact on the robustness of the solution, and the RMC affects the approximation result. We want the approximation to bring more robust solutions by adjusting the RMC<sup>22</sup>.

---

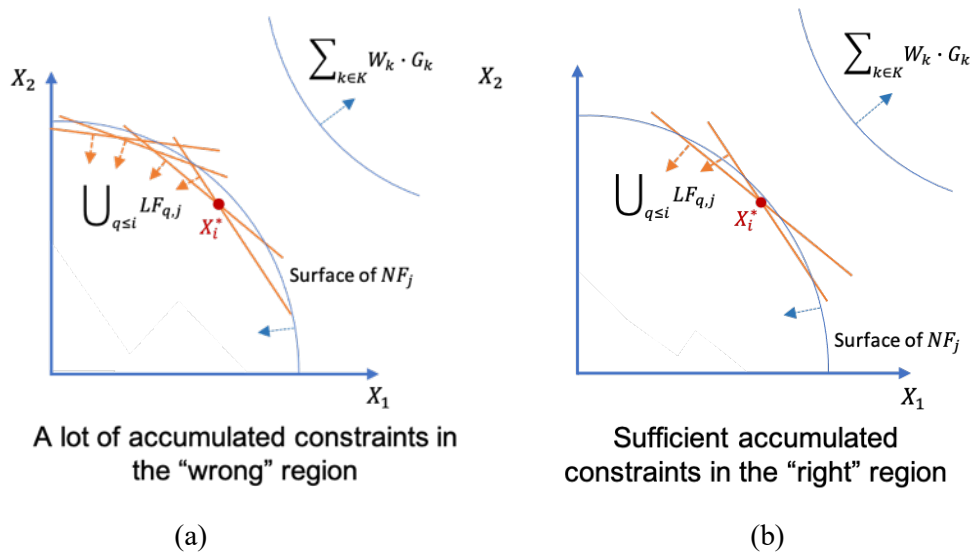
<sup>22</sup> In this chapter, the starting point of the test problem is a user-defined value and unchangeable, so the RMC is the only factor that affects the approximation quality.



**Figure 5. 10 A Relatively Sensitive Solution and a Robust Solution (Relatively Insensitive to Uncertainties)**

- *Criterion 3 – approximation accuracy.*

In general, an increase in accumulated constraints leads to an improvement of the approximation accuracy, but not necessarily the more the better; see Figure 5.11. Therefore, we want sufficient and useful accumulated constraints to bring desired solutions rather than having as many accumulated constraints as possible. Details of how we develop evaluation indices (EIs) for approximation accuracy are given in Section 5.3.2.



**Figure 5. 11 Unnecessary Accumulated Constraints versus Necessary Accumulated Constraints**

In Table 5.3, we summarize the criteria for the evaluation of the approximation performance. Based on these criteria, we develop the Evaluation Indices (EIs) in Section 5.3.2.

**Table 5. 3 Criteria for the Evaluation of Approximation Performance**

Criteria	Meaning and representation
Goals Fulfillment	Distance between achieved value and the target value of each goal
Robustness	Whether a solution is away from the physical boundary of the system
Approximation Accuracy	Whether the nonlinear constraints are approximated well in the sub-region that contain more desired solutions

### 5.3.2 Step 2 – Developing the Evaluation Indices (EIs)

To manage different preferences for multi-goal design problems, we obtain a number of solutions using multiple design scenarios<sup>23</sup>. As the design scenarios we use are discrete and do not cover all

<sup>23</sup> In this chapter, design scenarios include different weight vectors of the goals, different scenarios of constraint capacities (may result from different parameter values, different scenarios of quantitative relationship between variables, etc.), and any other factors that lead to model variations.

situations, we obtain limited, discrete solutions and use them to predict the *satisficing* solution space.

By running the Synthesis cycle (Rectangle A in Figure 5.3) using multiple design scenarios, the information such as the fulfillment of the goals, the activeness of the constraints and the accumulated constraints, etc., is acquired. Based on such information and the criteria in Table 5.3, the evaluation indices (EIs) are developed (Table 5.4). The description of the EIs are given as follows.

**Table 5. 4 Develop the Evaluation Indices (EIs) from the Information Obtained from ALP Running**

<b>Criteria</b>	<b>Information</b>	<b>EIs</b>	<b>Meaning</b>
Goals Fulfillment	Weighted sum of the deviation variables: $Z = \sum_{k \in K} W_k \cdot (d_k^- + d_k^+)$	$\mu_Z$	The average goals fulfillment in multiple design scenarios
		$\sigma_Z$	The standard deviation (Zarfl, Lumsdon et al.) of fulfillment of the goals (stability of performance in fulfillment of the goals) in multiple design scenarios
Robustness	The number of active bounds: $Nab$	$\mu_{Nab}$	The average sensitivity to variable bounds in multiple design scenarios
		$\sigma_{Nab}$	The SD of sensitivity to variable bounds in multiple design scenarios
	The number of active original constraints: $Naoc$	$\mu_{Naoc}$	The average sensitivity to original constraints in multiple design scenarios
		$\sigma_{Naoc}$	The SD of sensitivity to original constraints in multiple design scenarios
Approximation Accuracy	The number of accumulated constraints: $Nacc$	$\mu_{Nacc}$	The average complexity of the approximated problem in multiple design scenarios
		$\sigma_{Nacc}$	The SD of the complexity of the approximated problem in multiple design scenarios
	The number of iterations: $Nit$	$\mu_{Nit}$	The average convergence speed in multiple design scenarios
		$\sigma_{Nit}$	The SD of the convergence speed in multiple design scenarios

The development of the EIs is built on the index for the robust concept exploration method, EMI (error margin index) (Chen, Allen et al. 1996) and DCI (design capacity index) (Choi, Austin et

al. 2005). This is based on the observation that the results of each criterion follow Gaussian distribution, and the mean ( $\mu$ ) and standard deviation ( $\sigma$ ) can represent their characteristics. We assume that Gaussian distribution applies in general engineering design problems with continuous solution space. Therefore, we use the mean ( $\mu$ ) and the standard deviation ( $\sigma$ ) as the EIs and tune the RMC by minimizing the mean and the variance of each EI.

***Index for evaluating the fulfillment of the goals*** –  $\mu_Z$  **and**  $\sigma_Z$ . We obtain a set of solutions satisficing a set of design scenarios (Sabeghi, Shukla et al. 2016). We minimize the mean and standard deviation of Z,  $\mu_Z$  and  $\sigma_Z$ , to maximize the fulfillment of the goals and minimize the dispersion of fulfillment of the goals.

***Index for evaluating robustness*** –  $\mu_{Nab}$ ,  $\sigma_{Nab}$ ,  $\mu_{Naoc}$  **and**  $\sigma_{Naoc}$ . To develop the indices based on the robustness criterion, we have definitions and assumptions as follows.

In this chapter, we assume that all errors and incompleteness of a model can be reflected as the change of the slack or surplus of an inequality constraint or a variable bound (as another type of inequality constraint).

If an inequality constraint has zero slack or surplus when plugged in a feasible solution, we define the constraint as an active constraint. If the active constraint belongs to the original nonlinear problem  $P^0$ , as  $NF_j$  in Figure 5.10 (a), then the solution is sensitive to the physical boundary, because an error or variation that takes place to the active constraints may make the solutions infeasible. We define an original constraint with zero slack as an active original constraint, noted as AOC. We want to minimize the number of AOC.



If a solution makes a variable reach its upper bound or lower bound, the solution is relatively sensitive to physical boundary too. We define such bound as an active bound, noted as AB. We want to minimize the number of AB as well.

Any solution returned by using simplex (or any solution algorithms that are derived from simplex, such as dual simplex, revise dual simplex, etc.) is a vertex solution to the linear problem, therefore there should be at least one active constraint or active bound of the approximated linear problem, but we prefer fewer AOC and AB, so the solution can be more robust (or relatively insensitive) to potential errors or variations.

In summary, we simplify the robustness of a solution by looking at its number of active bounds  $Nab$  and its number of active original constraints  $Naoc$ . When the design scenarios change, we maximize the robustness of the design and minimize the variation of the robustness of the design by minimizing the mean and standard deviation of  $Nab$  and  $Naoc$ . Therefore, the EIs for robustness are  $\mu_{Nab}$ ,  $\sigma_{Nab}$ ,  $\mu_{Naoc}$ , and  $\sigma_{Naoc}$ .

***Index for evaluating the computational complexity*** –  $\mu_{Nacc}$ ,  $\sigma_{Nacc}$ ,  $\mu_{Nit}$  **and**  $\sigma_{Nit}$ . To evaluate how accurate the approximation is, we use the number of accumulated constraints  $Nacc$  and the number of iterations  $Nit$ . Since the approximation accuracy does not always get improved with the increase of accumulated constraints or the number of approximation iterations (Figure 5.10), we need to learn the range of  $Nacc$  and  $Nit$  that associated with good goal fulfillment and robustness. We desire the  $Nacc$  and  $Nit$  to be acceptable under all design scenarios and not too sensitive to the design change, so we measure their mean and standard deviation,  $\mu_{Nacc}$ ,  $\sigma_{Nacc}$ ,  $\mu_{Nit}$ , and  $\sigma_{Nit}$ .

In summary, we tune the RMC by satisfying the EIs –  $\mu_Z, \sigma_Z, \mu_{Nab}, \sigma_{Nab}, \mu_{Naoc}, \sigma_{Naoc}, \mu_{Nacc}, \sigma_{Nacc}, \mu_{Nit}$  and  $\sigma_{Nit}$  to get into their desired range (DEI). The formulation of the RMC tuning process is given as follows. Based on this formulation, we can learn the DEI and tune the RMC.

**Given**

A compromise formulation

Design scenarios

EIs:  $\mu_Z, \sigma_Z, \mu_{Nab}, \sigma_{Nab}, \mu_{Naoc}, \sigma_{Naoc}, \mu_{Nacc}, \sigma_{Nacc}, \mu_{Nit}, \sigma_{Nit}$

Iteration:  $i$

Number of EIs:  $n$

**Find**

RMC

DEI

Value of EIs of Iteration  $i$ :  $\mu_{Z_i}, \sigma_{Z_i}, \mu_{Nab_i}, \sigma_{Nab_i}, \mu_{Naoc_i}, \sigma_{Naoc_i}, \mu_{Nacc_i}, \sigma_{Nacc_i}, \mu_{Nit_i}, \sigma_{Nit_i}$

Rank of the EIs in the  $i^{th}$  iteration among EIs of all iterations:

$Rank(\mu_{Z_i}), Rank(\sigma_{Z_i}), Rank(\mu_{Nab_i}), Rank(\sigma_{Nab_i}),$

$Rank(\mu_{Naoc_i}), Rank(\sigma_{Naoc_i})$

$Sort[Rank_i]$ : Sort the Rank of the EIs in the  $i^{th}$  iteration among EIs of all iterations

**Satisfy**

$$\{\mu_{Z_i}, \sigma_{Z_i}, \mu_{Nab_i}, \sigma_{Nab_i}, \mu_{Naoc_i}, \sigma_{Naoc_i}, \mu_{Nacc_i}, \sigma_{Nacc_i}, \mu_{Nit_i}, \sigma_{Nit_i}\} \in DEI$$

**Minimize**

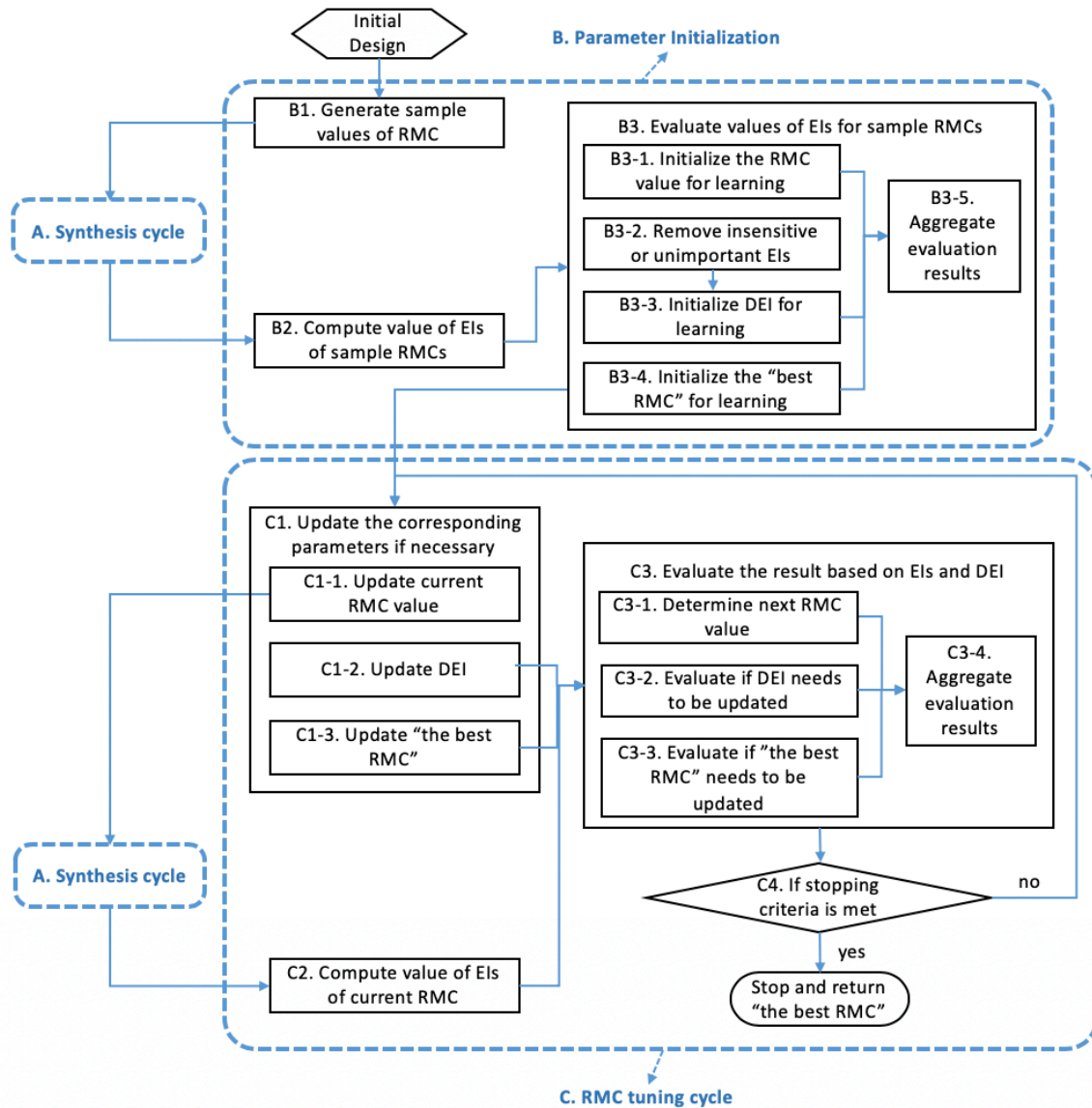
First  $\kappa$  items of  $Sort[Rank_i]$

**5.3.3 Step 3 – Learning the DEI and Tuning the RMC**

We learn the desired range of the EIs (DEI), learn the connections between the RMC value and the result of the EIs so that we can find the RMC to output *satisficing* solutions. While being updated along the iterating, DEI is expected to facilitate the evaluation of the approximation performance better.

To make the learning process efficient, we first adopt an off-line learning process using a sample of RMC values to initialize the parameters and then adopt an on-line learning process to tune the

RMC. In Figure 5.12, we illustrate the two processes in the dotted Rectangle B and C and show their relationship with the synthesis cycle A in Figure 5.3 (the main process of the ALP).



**Figure 5. 12 ALPPL Includes Parameter Initialization and the RMC Tuning**

During parameter initialization (Rectangle B in Figure 5.12), a sample of RMC values is generated (B1). By using each RMC value to run the synthesis cycle (A), the results (the corresponding EIs) are obtained (B2). Based on the results, we act evaluations (B2). We pick the RMC value that gives the best EIs as the starting RMC value for tuning (B3-1), and also as the initial best RMC to

be updated during the tuning process (B3-2). If there are any EI that is insensitive to RMC changing, or does not have any correlation with the other EIs, or simply not important for the current problem, We remove them to reduce the computational complexity (B3-3). We initialize the DEI based on the sample results (B3-4), to allow a certain percentage (e.g., 75%) RMC values to fall into the DEI. These evaluation results are aggregated (B3-5) and used as the input of the RMC tuning cycle (C).

With the new RMC value, we run the synthesis cycle (A) and obtain the results (C2). By evaluating the new results using the DEI and comparing with previous cycles (C3), we determine the next RMC value (C3-1), evaluate if current DEI needs to be updated to either restrict or relax the satisficing solution space based on the tradeoffs between EIs (C3-2), and evaluate if the best RMC needs to be updated (C3-3). These evaluation results are aggregated (C3-4). After judging whether the iterating should stop (C4), the program either goes to the next iteration of RMC tuning with the aggregated results (C3-4) as input or stops with the best RMC as the returned value. The stopping criteria include the number of total iterations and the number of iterations without updating the best RMC. The RMC tuning process is summarized in Table 5.5. In Section 5.4, a test problem is used to verify the efficacy of the ALPPL.

**Table 5. 5 The Parameter Learning Process – for RMC tuning**

---

1	<i>Given:</i> DEI, the best RMC sample value, DEI updating rules
2	<i>Initialize:</i> t <- 0, best <- the best RMC sample value, RMC <sub>0</sub> <- the best RMC sample value, the maximum iteration number T, stopping criterion 2 <- {best has not been updated in n iterations}
3	<b>While</b> t ≤ T do // Define stopping criterion 1
4	RMC <sub>t</sub> <- Next_RMC
5	Run synthesis cycle
6	Calculate EI[RMC <sub>t</sub> ]
7	<b>if</b> EIs[RMC <sub>t</sub> ] > EIs[best]
8	best <- RMC <sub>t</sub> // Update the best RMC
9	<b>if</b> DEI should be updated
10	update DEI
11	<b>if</b> stopping criterion 2 is met

---

---

```

12   break
13   else
14     Next_RMC <- Determine_next{EIs[RMCt], EIs[RMCt-1], EIs[RMCt-1], best}
// Use EIs of the tth, (t-1)th, (t-2)th, and best to determine the RMC of the next iteration24
15     t <- t+1
16   return best

```

---

## 5.4 The Hot Rolling Process Chain Problem

### *- Test Problem 3: apply ALPPL an engineering-design problem*

In this section, we illustrate the efficacy of the ALPPL using an industry-inspired test problem – the integrated design of a hot rolling process chain for the production of a steel rod (Nellippallil, Rangaraj et al. 2018). We choose this problem because it is a nonlinear problem formulation, and the RMC value has a significant impact on the result.

#### *5.4.1 Statement of Test Problem 2*

Hot rolling is a multi-stage manufacturing process in which a reheated billet, slab, or bloom that is produced after the casting process is further thermo-mechanically processed by passing through a series of rollers, see (Nellippallil, Song et al. 2017, Nellippallil, Rangaraj et al. 2018). During the thermo-mechanical processing, there is an evolution of microstructure of the material. The columnar grains in the material are broken down to equiaxed grains. Along with the evolution of grain size, the phase transformation of the steel happens. Phase transformation is predominant during the cooling stage that follows the hot rolling process chain. The transformation of the austenite phase of steel to other phases like ferrite, pearlite, martensite, etc., takes place during this stage. The final microstructure of the material after rolling and cooling process defines the

---

<sup>24</sup> The details of the determination algorithms are given in Appendix D.

mechanical properties of the product. Many plant trials are required to produce a new steel grade with improved properties and performance. These trials are usually expensive and time-consuming. Hence there is a need to address the problem from a simulation-based design perspective to explore solutions that *satisfice* multiple conflicting property/ performance goals. In this problem, the requirement is to produce steel rods with improved mechanical properties like yield strength (*YS*), tensile strength (*TS*), and hardness (*HV*). These mechanical properties are defined by the microstructure after cooling, which includes, the phase fractions (ferrite and pearlite phases are only considered in this problem), pearlite interlamellar spacing, ferrite grain size, and chemical compositions, see (Nellippallil, Rangaraj et al. 2018) for the mechanical property-microstructure relationships identified. Using a goal-oriented inverse design method, Nellippallil et al. (Nellippallil, Rangaraj et al. 2018) identify the microstructural requirements after the cooling stage to meet the mechanical properties of the rod. The microstructural requirements are to achieve a high ferrite fraction value, low pearlite interlamellar spacing, and low ferrite grain size value within the defined ranges. The requirement is to carry out the integrated design of the material and the process by managing the cooling rate (cooling process variable), final austenite grain size after rolling (rolling microstructure variable) and the chemical compositions of the material. Hence, our interest in this problem is to explore the solution space of the defined variables using ALPPL to meet the target values identified for the microstructure after cooling stage such that the mechanical property requirements of the steel rod are met. Our focus in this chapter is to use this example problem in improving the solution algorithm rather than the detailed design of the material and the manufacturing process.

The initial design formulation of the problem using the *Given*, *Find*, *Satisfy*, and *Minimize* keywords is shown as follows.

## Given

1) Target values for microstructure after cooling

Ferrite Grain Size Target,  $D_{\alpha Target} = 8 \mu m$

Ferrite Fraction Target,  $X_{f Target} = 0.9$

Pearlite Interlamellar Spacing Target,  $S_{o Target} = 0.15$

2) Well established empirical and theoretical correlations, response surface models, and complete information flow from the end of rolling to the end product mechanical properties (Details provided in (Nellippallil, Rangaraj et al. 2018))

3) System variables and their ranges

## Find

System Variables

$X_1$ , Cooling Rate ( $CR$ )

$X_2$ , Austenite Grain Size ( $D$ )

$X_3$ , the carbon concentration ( $[C]$ )

$X_4$ , the manganese concentration after rolling ( $[Mn]$ )

Deviation Variables

$d_i^-, d_i^+, i=1,2,3$

## Satisfy

System Constraints

Minimum ferrite grain size constraint

$$D_{\alpha} \geq 8 \mu m \quad \text{Equation 5. 10}$$

Maximum ferrite grain size constraint

$$D_{\alpha} \leq 20 \mu m \quad \text{Equation 5. 11}$$

Minimum pearlite interlamellar spacing constraint

$$S_o \geq 0.15 \mu m \quad \text{Equation 5. 12}$$

Maximum pearlite interlamellar spacing constraint

$$S_o \leq 0.25 \mu m \quad \text{Equation 5. 13}$$

Minimum ferrite phase fraction constraint (manage banding)

$$X_f \geq 0.5 \quad \text{Equation 5. 14}$$

Maximum ferrite phase fraction constraint (manage banding)

$$X_f \leq 0.9 \quad \text{Equation 5. 15}$$

Maximum carbon equivalent constraint

$$C_{eq} = (C + Mn)/6; C_{eq} \leq 0.35 \quad \text{Equation 5. 16}$$

Mechanical Property Constraints

Minimum yield strength constraint

$$YS \geq 250 \text{ MPa} \quad \text{Equation 5. 17}$$

Maximum yield strength constraint

$$YS \leq 330 \text{ MPa} \quad \text{Equation 5. 18}$$

Minimum tensile strength constraint

$$TS \geq 480 \text{ MPa} \quad \text{Equation 5. 19}$$

Maximum tensile strength constraint

$$TS \leq 625 \text{ MPa} \quad \text{Equation 5. 20}$$

Minimum hardness constraint

$$HV \geq 130 \quad \text{Equation 5. 21}$$

Maximum hardness constraint

$$HV \leq 150 \quad \text{Equation 5. 22}$$

System Goals

The target values for system goals are identified in (Nellippallil, Rangaraj et al. 2018) and are listed in the Given keyword above.

Goal 1: Achieve Ferrite Grain Size Target

$$\frac{D_{\alpha Target}}{D_{\alpha}(X_i)} + d_1^+ - d_1^- = 1 \quad \text{Equation 5. 23}$$

Goal 2: Achieve Ferrite Fraction Target

$$\frac{X_{f Target}}{X_f(X_i)} + d_2^- - d_2^+ = 1 \quad \text{Equation 5. 24}$$

Goal 2: Achieve Pearlite Interlamellar Spacing Target

$$\frac{S_{o Target}}{S_o(X_i)} + d_3^+ - d_3^- = 1 \quad \text{Equation 5. 25}$$

Variable Bounds

$$\begin{aligned}
11 &\leq X_1 \leq 100 \text{ (K/min)} \\
30 &\leq X_2 \leq 100 \text{ (\mu m)} \\
0.18 &\leq X_3 \leq 0.3 \text{ (\%)} \\
0.7 &\leq X_4 \leq 1.5 \text{ (\%)} \\
\text{Bounds on deviation variables} \\
d_i^-, d_i^+ &\geq 0 \text{ and } d_i^- * d_i^+ = 0, i = 1, 2, 3
\end{aligned}$$

Equation 5. 26

### **Minimize**

$$\text{Minimize the deviation function in the initial design } Z = \sum_{i=1}^3 W_i(d_i^- + d_i^+); \sum_{i=1}^3 W_i = 1$$

Equation 5. 27

There are three goals in the problem discussed – i) Minimize Ferrite Grain Size ( $D_\alpha$ ), ii) Maximize Ferrite Fraction ( $X_f$ ), and iii) Minimize Interlamellar Spacing ( $S_o$ ). The target values and the acceptable values of the three goals are determined and defined. All three goals require nonlinear goal formulation. There are four design/system variables – i) cooling rate ( $CR$ ), ii) final austenite grain size after rolling ( $D$ ), iii) the carbon concentration ( $[C]$ ), and iv) the manganese concentration after rolling ( $[Mn]$ ). We obtain: i) the range of the system variables that *satisfice* the goals for the different design preferences assigned, and ii) the *satisficing* weight set,  $W^s = \bigcap_{k \in K} W_k^s$ , the set of weight scenarios that *satisfices* all three goals for the different design preferences assigned.

In (Nellippallil, Rangaraj et al. 2018), the authors formulate and execute the initial design compromise decision support problem for the hot rolling process chain problem and carry out weight sensitivity analysis to identify the *Satisficing* Weight Set of the three goals. Ternary plots are generated to visualize and explore the weight set. In each ternary plot, the three axes represent the weights assigned to the three goals, respectively, and the color contours indicate the achievement of each goal,  $\frac{G_k}{T_k}$ ,  $k = 1, 2, 3$ . Since the goals conflict with each other and the total resources for the goals are limited, compromise solutions are desired. Weight sensitivity analysis is a way to mediate compromise or innovate around the conflicts between goals.  $W_1^s$ ,  $W_2^s$ ,  $W_3^s$ , and  $W^s$  are the *satisficing* weight regions identified in the ternary plots, see areas identified using



arrows in Figures 11 (a), (b), (c), and (d), respectively. To discuss and compare the achievements of multiple goals,  $\frac{G_k}{T_k}$  are normalized within the range [0, 1]. An acceptable value of each goal is identified and plotted as a dashed line in each ternary plot. The area between the corner with the best  $\frac{G_k}{T_k}$  value and the dashed line (the acceptable value) is the *satisficing* weight area of one goal,  $W_k^s$ , and the superimposed area is the *satisficing* weight area of all three goals,  $W^s$ .

In Figures 5.13-5.15, we illustrate how different RMC values affect  $W_k^s$  and  $W^s$ . When RMC is 0.1, as Fig. 11 shows,  $W^s$  is large, whereas when RMC is 0.8, as Figure 5.15 shows,  $W^s$  is small. However, we do not have rules to evaluate which RMC value results in a relatively accurate approximation and thereby gives us the most robust  $W^s$ . Therefore, the ALPPL is applied to fill in this gap.

We implement the parameter initialization (Process B in Figure. 5.12) and the RMC tuning (Process C in Figure 5.12) using Python and incorporate executing the ALP (Synthesis cycle A in Figure. 5.12) to obtain the results.

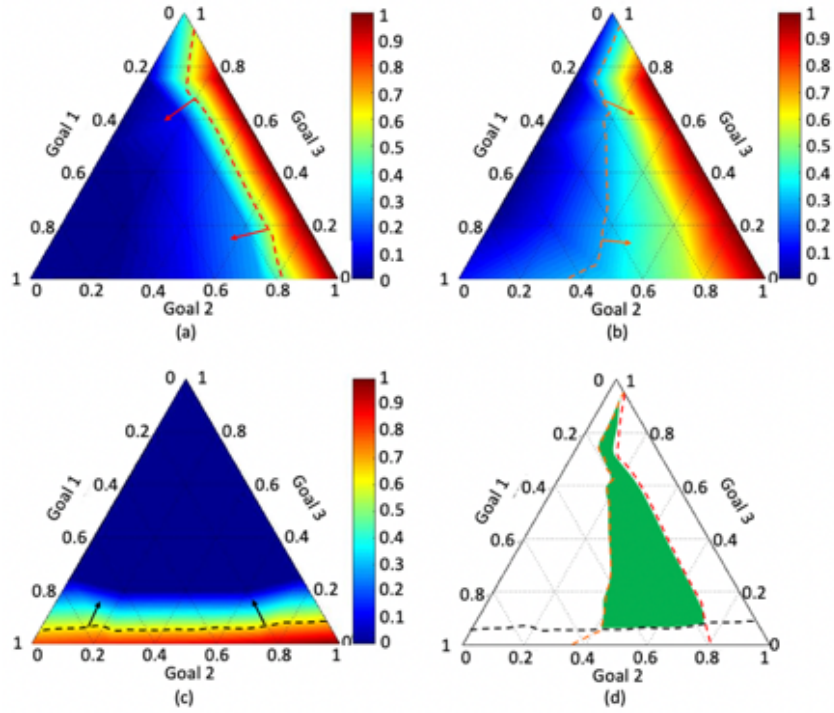


Figure 5. 13 The Satisficing Weight Set When Setting RMC=0.1

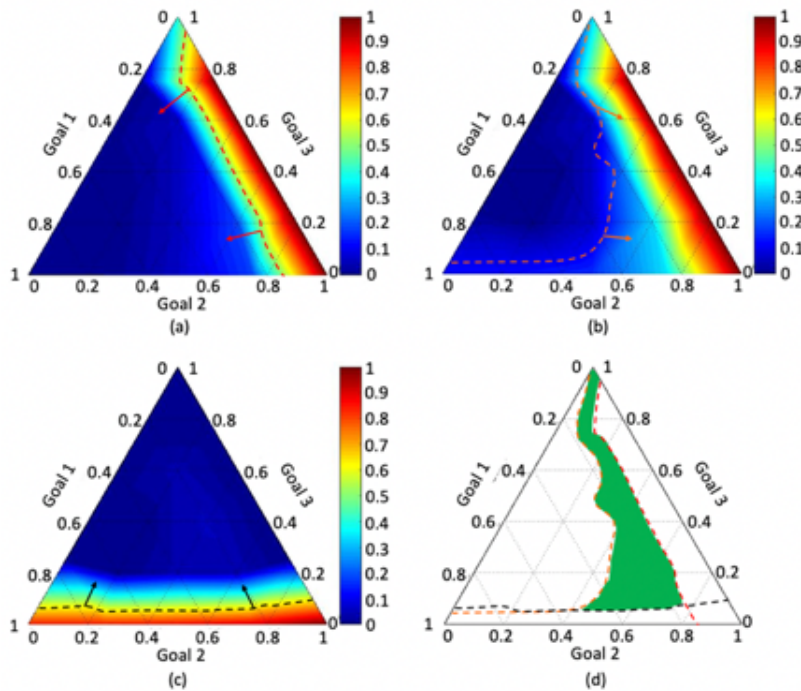


Figure 5. 14 The Satisficing Weight Set When Setting RMC=0.5

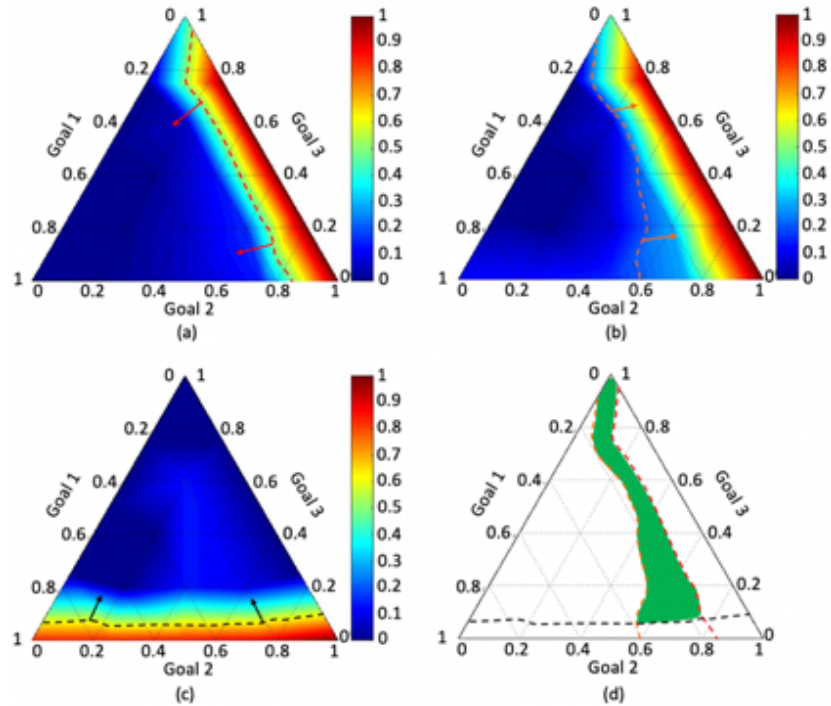


Figure 5.15 The Satisficing Weight Set When Setting RMC=0.8

#### 5.4.2 Applying ALPPL

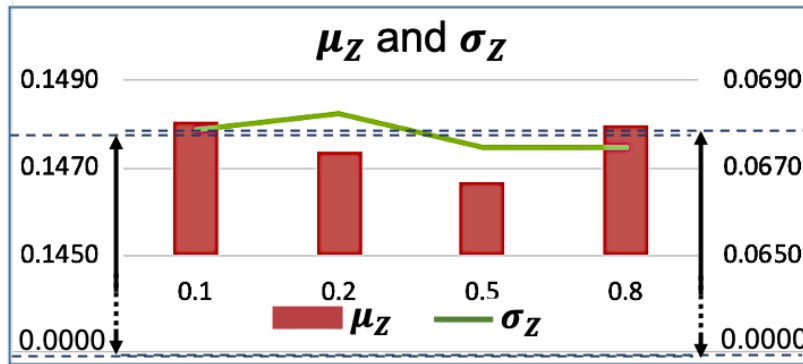
##### Parameter Initialization

The design scenarios used in (Nellippallil, Rangaraj et al. 2018) are nineteen weight vectors (weight scenarios) of the goals (Table 5.6), that represent a variety of design preferences. Then we go through Process B1, B2, and B3 in Figure 5.12 as follows.

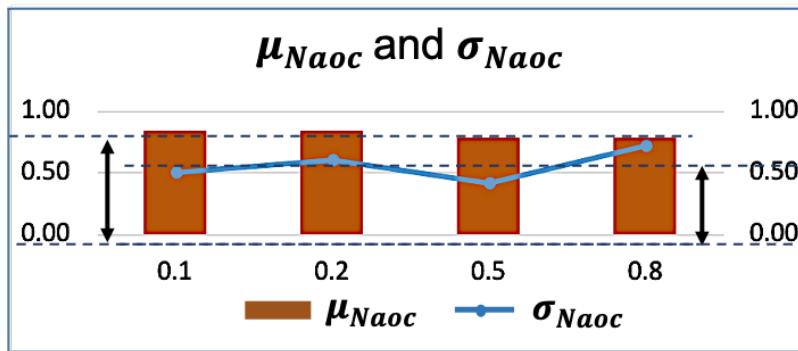
Table 5. 6 Weight Vectors Used in (Nellippallil, Rangaraj et al. 2018) as Different Design Scenarios

$W$	W1	W2	W3	$W$	W1	W2	W3
1	1	0	0	11	0	0.75	0.25
2	0	1	0	12	0	0.25	0.75
3	0	0	1	13	0.33	0.33	0.33
4	0.5	0.5	0	14	0.2	0.2	0.6
5	0.5	0	0.5	15	0.4	0.2	0.4
6	0	0.5	0.5	16	0.2	0.4	0.4
7	0.25	0.75	0	17	0.6	0.2	0.2
8	0.25	0	0.75	18	0.4	0.4	0.2
9	0.75	0	0.25	19	0.2	0.6	0.2

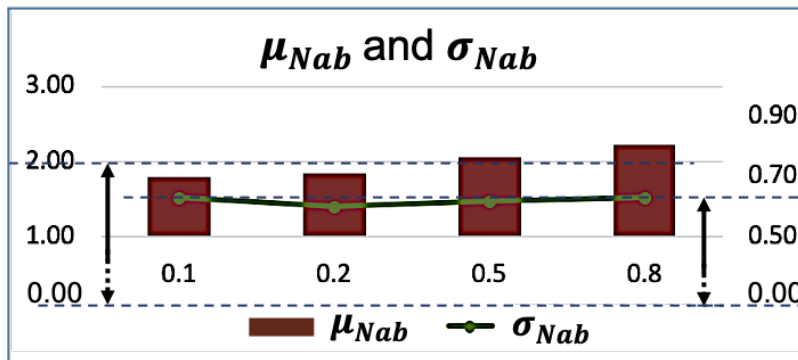
10	0.75	0.25	0
----	------	------	---



(a)



(b)



(c)

**Figure 5. 16 EIs and DEI of the Sample RMC Values**

**B1:** We randomly choose some values – 0.1, 0.2, 0.5 and 0.8 as the RMC sample.

**B2:** By using the four RMC values and the nineteen weight vectors, we solve the problem for  $4 \times 19 = 76$  times and obtain 76 results of each EI. The results of the EIs of the four RMC values are in Table 5.7.

**B3:** Based on the EIs of the sample RMC (Table 5.4), we act an evaluation as follows.

**B3-1:** Initialize the RMC value for learning. As  $RMC_0 \leftarrow$  the best RMC sample value, hence,  $RMC_0 = 0.5$ .

**B3-2:** Remove insensitive or unimportant EIs –  $\mu_{Nacc}$ ,  $\sigma_{Nacc}$ ,  $\mu_{Nit}$ , and  $\sigma_{Nit}$ .

**B3-3:** Initialize DEI. For each EI, we define the range from the median of the four sample results to the ideal value as the initial DEI. For example, the ideal value of  $\mu_Z$  is zero because ideally, we want the deviation function  $Z$  to be zero so that the goals can be fully achieved; the median of  $\mu_Z$  of the four RMC sample values is 0.1477; therefore, the initial DEI of  $\mu_Z$  is  $[0, 0.1477]$ . We visualize the results of the EIs for the four sample RMC values in Figure 5.16. For each EI, there is a graph showing its mean and standard deviation, as Figure 5.16 (a)-(c). In each graph, the left vertical axis represents the mean value, the right vertical axis represents the standard deviation value, and the horizontal axis represents RMC value. The columns are means and the lines are standard deviations. The left arrows and the right arrows show the desired range of the mean and standard deviation, respectively. The initial DEI are summarized in Table 5.8.

**B3-4:** Initialize the best RMC. As  $best \leftarrow$  the best RMC sample value, hence,  $best = 0.5$ .

**B3-5:** We aggregate the results of the parameter initialization –  $RMC_0$ , EIs, DEI, and best, as the input of RMC tuning.

**Table 5. 7 Results of EIs Using Sample RMC Values with Nineteen Design Scenarios**

RMC	Statistics	Z	Nit	Nacc	Nab	Naoc
0.1	$\mu$	0.1480	46.58	18.74	1.79	0.84
	$\sigma$	0.0679	5.95	0.87	0.63	0.50
0.2	$\mu$	0.1474	34.16	19.00	1.84	0.84
	$\sigma$	0.0682	7.88	0.82	0.60	0.60
0.5	$\mu$	0.1467	20.42	19.47	2.05	0.79

	$\sigma$	0.0675	9.47	0.84	0.62	0.42
0.8	$\mu$	0.1480	8.32	14.16	2.21	0.79
	$\sigma$	0.0675	5.56	6.94	0.63	0.71

**Table 5. 8 The Initial DEI**

DEI of $\mu_Z$	DEI of $\sigma_Z$	DEI of $\mu_{Naoc}$	DEI of $\sigma_{Naoc}$	DEI of $\mu_{Nab}$	DEI of $\sigma_{Nab}$
[0, 0.1477]	[0, 0.0677]	[0, 0.82]	[0, 0.55]	[0, 1.95]	[0, 0.63]

### RMC Tuning

In RMC tuning, we make rules to proceed with each procedure based on heuristics. The heuristics are generalized from parameter learning and can be adjusted through the search process.

In the first iteration of RMC tuning, Process C1 is based on the output of Process C3 and Process C2 is the same with Process B2, so we start making rules for Process C3 and go back to C1 later in this section.

**C3:** Evaluate the result of current RMC based on EIs and DEI.

**C3-1:** Determine the next RMC value.

**Rule 1: Compare the performance of multiple EIs and define the comparison rules** (Table 5.3, Line 14). Lines 22-30 in Appendix D are an expansion of this rule. We define RMC A is better than RMC B as “no less than  $\kappa$  of the EI(A) are better than EI(B), whereas other EI(A) do not exceed  $\gamma$  of the upper and lower bound of DEI. This rule can be applied differently to other problems. In this problem, we set  $\kappa=1/2$  and  $\gamma=30\%$ .”

**Rule 2: Determine when and how the RMC should be updated.** Line 6-13 in Appendix D explain this rule. We use a hill-climbing approach to update the RMC (Appendix D, Line 5-8). If the updating in the previous RMC tuning cycle does improve the performance, then the previous updating is in the “hill-climbing direction”, as a result, we keep updating the RMC in this direction

with a step size  $\alpha$ ; otherwise, we need the best RMC to help us “back to track” with a portion  $\beta$ , hence we update the RMC as the linear combination of the best RMC (elite) and the RMC in the two cycles ago (parent). In this problem, we set  $\alpha$  as a random value that uniformly distributed in  $[0, 1]$  and set  $\beta$  as a random value that uniformly distributed in  $[0.5, 1]$ . In this way, we incorporate greediness, elitism, and randomness in evolution.

**C3-2:** Evaluate if DEI needs to be updated.

**Rule 3: Determine when and how the DEI should be updated.** See Appendix D, Line 18-38. If in an RMC-tuning cycle, the RMC leads us to get more than  $\kappa$  EIs better than the EIs of the previous cycle, and more than  $\iota$  EIs are in the desired range (DEI) whereas only  $(1 - \iota)$  EIs have minor violations (Appendix D, Line 27), we define the current RMC performs better than the previous RMC and update the DEI. In this problem, we set  $\kappa = \iota = 2/3$  because the number of EIs is small. For the problem that has many EIs,  $\kappa$  and  $\iota$  can be tuned using the performance improvement rate or proportion of acceptable results among all the results as the tuning goal. In this way, we prevent insensible DEI stopping us from walking to a better range, meanwhile ensure gradual and relatively conservative updating of DEI.

**C3-3:** Evaluate if “the best RMC” needs to be updated.

**Rule 4: Determine when and how the current best RMC is updated.** See Appendix D, Line 34-36. We use a variable (“best”) to store the current best RMC. If more than  $\kappa$  EIs of current RMC are better than the EIs of the best, we set the current RMC as the new best.

**C3-4:** We aggregate the results of the RMC tuning –  $\text{RMC}_{t+1}$ , DEI, and best, as the input of the next tuning iteration.

**C4:** Determine if the iterating should stop.

**Rule 5: Make the stopping criteria.** In order to stop the RMC tuning at the appropriate time avoid overwhelming computation, we use two stopping criteria – the maximum number of RMC tuning iterations and maximum number of RMC-tuning iterations without updating the best RMC. See Appendix D, Line 41-42.

### 5.4.3 Parameter Learning Results and Discussion

After running the RMC tuning for fourteen iterations, the tuning stops. We identify 0.55 as the best RMC for the cooling problem. Comparing with the first “best RMC” 0.5, the finalized best RMC 0.55 brings improvement of  $\sigma_Z$ ,  $\sigma_{Naoc}$ , and  $\sigma_{Nab}$ . The RMC in the fourteen iterations and their EIs are given in Table 5.9. During the fourteen iterations, the DEI is updated four times, and the best RMC is updated three times. The final best RMC is in the ninth iteration. In the first seven cycles, the RMC value varies considerably because we need relatively big oscillation at the early stage to make sure that the desired sub-range of RMC is explored; in the last seven cycles, the RMC value varies little, which is an indication that the desired sub-range of RMC is explored sufficiently and the appropriate value can be identified.

**Table 5.9 The Record of the EIs, DEI, RMC, Best RMC of the Fourteen Iterations of RMC Tuning**

Iteration	RMC	Deviation Function		Number of Active Original Constraints		Number of Active Bounds		Better than Cycle (t-1)	Better than best RMC	Update DEI
		$\mu_Z$	$\sigma_Z$	$\mu_{Naoc}$	$\sigma_{Naoc}$	$\mu_{Nab}$	$\sigma_{Nab}$			
1	0.5	0.147	0.068	0.79	0.42	2.05	0.62	-	-	$\mu_{Nab} \sim [1, 1.95]$ ->
2	1.0	0.152	0.071	0.95	0.78	2.26	0.45	N	N	$\mu_{Nab} \sim [1, 2.05]$ -
3	0.8	0.148	0.068	0.79	0.71	2.21	0.63	N	N	-



4	0.6	0.147	0.067	0.95	0.52	2.00	0.58	Y	Y	$\mu_{Naoc} \sim [0, 0.82]$ ->
5	0.4	0.147	0.068	0.68	0.48	2.00	0.58	Y	Y	$\mu_{Naoc} \sim [0, 0.95]$ -
6	0.2	0.147	0.068	0.84	0.60	1.84	0.60	N	N	$\sigma_{Naoc} \sim [0, 0.55]$ ->
7	0.3	0.154	0.069	0.95	0.52	1.79	0.71	N	N	$\sigma_{Naoc} \sim [0, 0.6]$ -
8	0.45	0.147	0.068	0.89	0.66	2.00	0.58	Y	N	-
9	0.55	0.147	0.067	0.84	0.37	2.05	0.52	Y	Y	-
10	0.65	0.147	0.067	1.00	0.58	2.05	0.62	N	N	$\mu_{Naoc} \sim [0, 0.95]$ ->
11	0.48	0.147	0.068	0.89	0.66	2.05	0.62	N	N	$\mu_{Naoc} \sim [0, 1.00]$ -
12	0.53	0.147	0.067	0.84	0.69	2.00	0.58	Y	N	-
13	0.57	0.147	0.067	0.84	0.37	2.11	0.57	N	N	-
14	0.43	0.145	0.069	0.84	0.60	2.00	0.58	N	N	-

To verify the efficacy of ALPPL, we evaluate the adequacy and the necessity of the algorithm in obtaining robust solutions.

**Adequacy** – The best RMC (0.55) ensures the solutions falling in a relatively insensitive range.

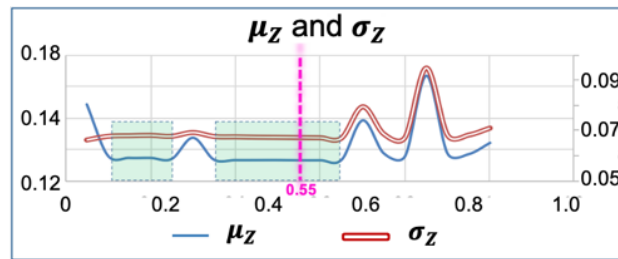
**Necessity** – The insensitive range is sufficiently explored during the RMC tuning.

First, we identify what is the insensitive range of RMC value. We test 20 RMC values that uniformly distributed in  $[0, 1]$  and obtain their EIs (Figure 5.17). In each graph of Figure 5.17, the range in the dotted rectangle is the insensitive range<sup>25</sup> of RMC for each EI based on the twenty results.

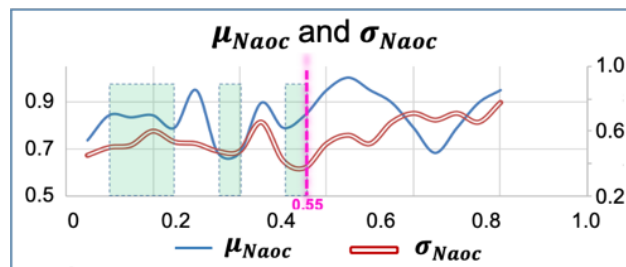
**Verification of the adequacy.** We observe from Figure 5.17 that 0.55 is in the insensitive RMC range for all EIs, so it is verified that when RMC is 0.55, it gives a relatively robust performance.

<sup>25</sup> We identify the insensitive range by using quantitative methods such as searching for  $[x_1, x_2]$  where  $\frac{\max[y(x)] - \min[y(x)]}{x_2 - x_1} \leq \gamma$ ,  $x \in [x_1, x_2]$ , and determine the value of  $\gamma$  as a rate of  $(\min[y(x)] - \max[y(x)])$ ,  $x \in [0, 1]$ , for each EI.

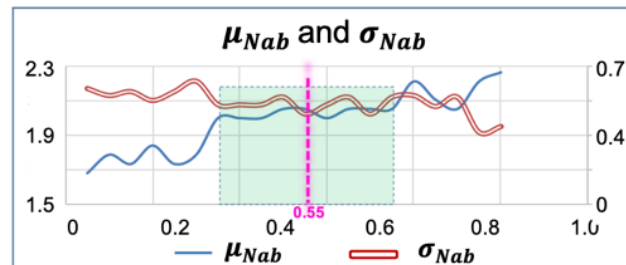
**Verification of the necessity.** From Figure 5.17, we observe that the ranges [0.35, 0.4] and [0.5, 0.55] are insensitive ranges in all three EIs, so within the RMC values in these two ranges, the solutions are relatively insensitive. In Figure 5.18, we illustrate all the fourteen RMC values used during RMC tuning. The horizontal axis represents the iteration number, and the vertical axis represents the RMC value. There are four out of the fourteen RMC values falling in the two insensitive ranges, so 28.5% of the RMC values we test during the tuning fall in the insensitive ranges, whereas the insensitive ranges only occupy 10% of the whole RMC range. Hence, we conclude that our rules in the RMC tuning enable a relatively sufficient exploration of the insensitive ranges.



(a)

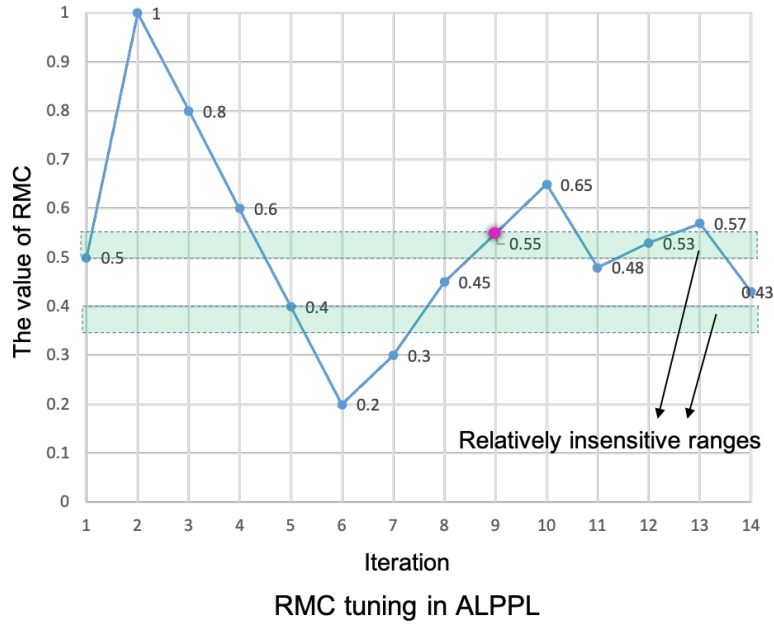


(b)



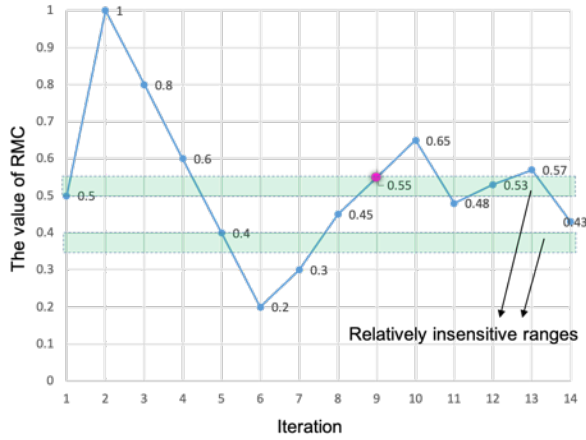
(c)

**Figure 5. 17 Identifying the Insensitive Range of RMC Value Using Twenty RMC Values**



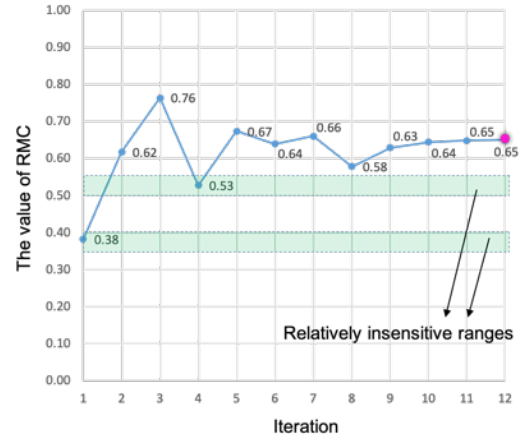
**Figure 5. 18 The Fourteen RMC Values in the RMC Tuning**

*Verification of the improvement of ALPPL over ALP.* By visualizing the RMC values tested using ALPPL versus the RMC values tested using the golden section search, in Figure 5.19, we observe that the best RMC identified using golden section search is 0.65, which is not in the insensitive range; in addition, the RMC values tested in golden section search are concentrated in [0.57, 0.77], which misses the insensitive ranges [0.35, 0.4] and [0.5, 0.55]. Other improvements of ALPPL over ALP are summarized in Table 5.10.



RMC tuning in ALPPL

(a)



Golden section search in ALP

(b)

**Figure 5. 19 The Comparison of ALPPL And ALP regarding the RMC Updating**

**Table 5. 10 ALPPL with RMC Tuning versus ALP with Golden Section Search**

	ALPPL	ALP
	Rule-based parameter learning	Golden section search
<b>General comparison</b>	Search method	Search method
	Criteria used for evaluation of the RMC	Deviation (fulfillment of the goals), the robustness of the solution
	If the approximation is sensitive to the scenario changing	Considering different scenarios, the most appropriate RMC is identified. The approximation is relatively insensitive to scenario changing
Stopping criteria	The best RMC has not been updated for n iterations, or the total iteration number reaches a threshold	Fulfillment of the goals
<b>Comparison of the cooling problem results</b>	Number of search iterations	In each scenario, the best RMC is identified, and it may vary as scenario changing. The approximation is relatively sensitive to scenario changing
	If the identified best RMC is in the insensitive range	The distance between two golden section points are less than a threshold $\epsilon$
	Number of tested RMC values falling into insensitive range	14
	Is the insensitive range explored sufficiently	Yes
	4	No
	Relatively sufficiently	Insufficiently

## 5.5 Role of Chapter 5 in this Dissertation

### *5.5.1 Summarizing How We Finish Task 2: Connecting Approximation, Exploration, and Evaluation*

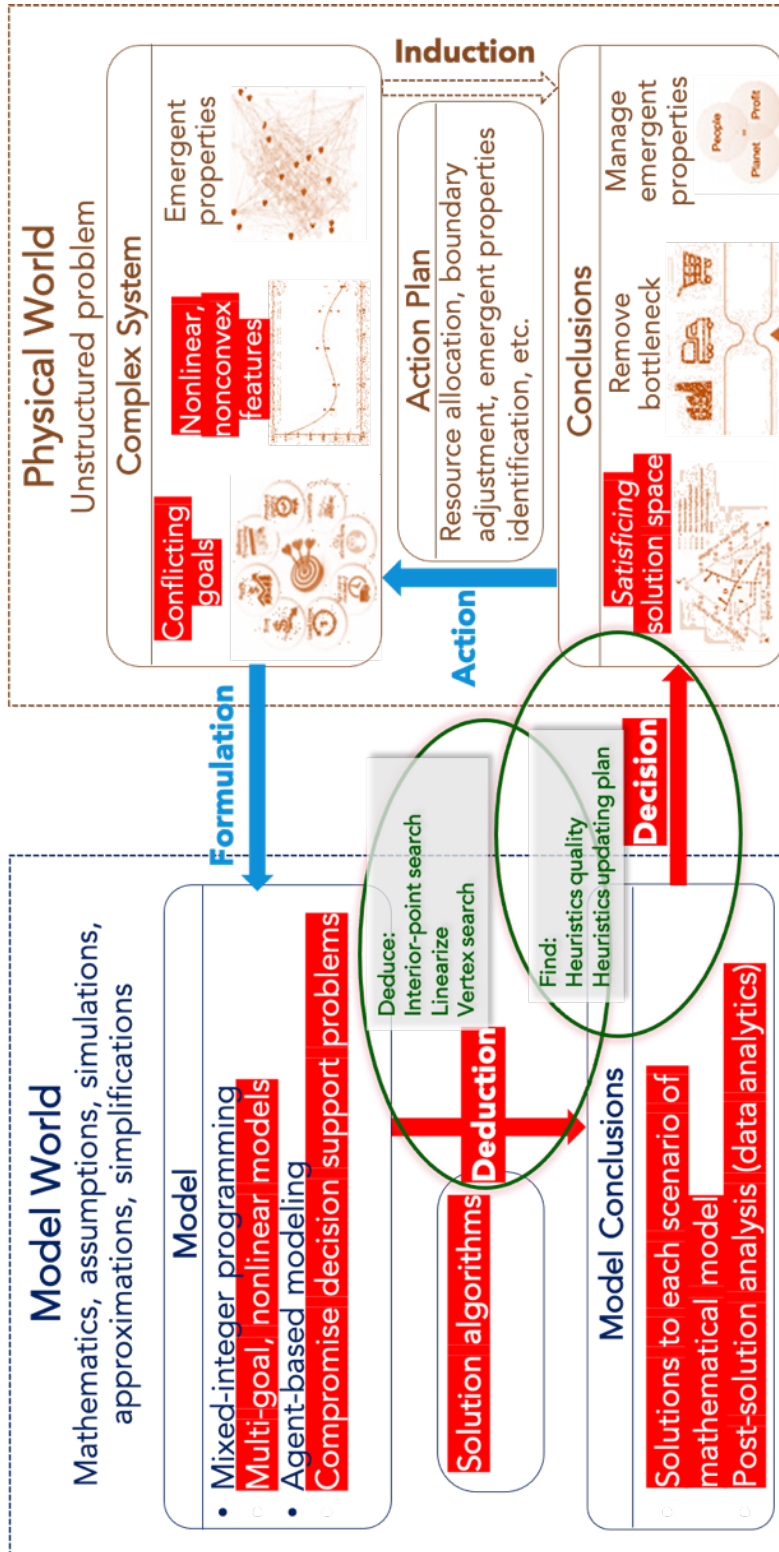
For the decision model of a complex system that contains nonlinear and probably non-convex equations, when approximating the problem to a linear and convex problem for solving, designers apply heuristics or metaheuristics, which simplify the problem and guarantee feasible solutions but may lose information. Even those metaheuristics are not the best ones that help acquire the approximations and solutions with an acceptable quality – here quality may implicate but is not limited to approximation accuracy, solution robustness to multiple types of uncertainty, the computational complexity of the approximation and solution searching – there is not any mechanism to evaluate such quality and learn the association between the heuristics and the quality.

The essence of Specific Hypothesis 2, “*learn, evaluate, and update metaheuristics to improve model performance*” is, to improve the model approximation, evaluate and update the heuristics, and obtain knowledge on the tradeoffs between exploration and exploitation of the problem approximation and post-solution analysis, designers need to explore different heuristics and establish the connection among the approximation, exploration, and evaluation.

In this chapter, we summarize the process of exploring, learning, and updating heuristics that needs to be customized for each problem into a more generally useful algorithm, the parameter learning process in Table 5.5. The customization of the algorithm for the test problem, the hot rod rolling process chain, is in Appendix D.

In other words, we strengthen the connections among deduction and decision, as shown in Figure 5.20. In this dissertation, we establish the information exchange, knowledge awareness, and

instructions sharing between the three processes and make it standardized; see Table 5.5 and Appendix D.



**Figure 5. 20 The Procedures Involved in Approximation-Exploration-Evaluation – Establish the information exchange, knowledge awareness, and instructions sharing between deduction and decision**

### 5.5.2 Summarizing How We Realize Type I, II, & III Robust Design

For the test problem, the cooling stage of the hot rod process chain, Type I uncertainty is identified as the various values of the reduced move coefficient (RMC) of the adaptive linear programming (ALP) algorithm (Table 5.7, Figure 5.16). Type II uncertainty is implemented as the design scenarios that represent the different priorities of the goals (Table 5.6). Type III uncertainty is a result of the incorporation of Type I uncertainty – when using different RMC values, the model structure changes as the linearized problem vary with the RMC. A representation of the influence of Type III uncertainty on the result is shown in Figures 5.13-5.15. Type I, II, and III uncertainties are managed though the parameter learning algorithm in Table 5.5.

By implementing the parameter learning algorithm and customizing it in the rod rolling process chain problem as the method in Appendix D, we can identify the solution space that is relatively insensitive to the Type I, II, and III uncertainty that we determine to manage. In this way, we realize Type I, II & III robust design; see the summary in Table 5.11 as the closing remarks of Table 3.2 regarding the robust design realization and uncertainty management for Test Problem 2.

**Table 5. 11 Summary of Test Problems 2 regarding Type I, II&III Uncertainty Management**

RD Type	RDI-II		RDIII	
			RDIV	
Method	M1: Formulation-Exploration Framework	<b>M2: Adaptive Linear Programming with Parameter Learning (ALPPL)</b>	M3: Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)	M4: Scenario Planning in Agent-Based Modeling
Chapter	Ch 4	<b>Ch 5</b>	Ch 6	Ch 7



Uncertainty / Test Problem	T1: Dam network	T2: Supply chain	<b>T3: Hot rolling process chain</b>	T4: Thermal system	T5: Promoting second-season farming
Type I	<i>Uncertainty in timing and amount of inflow – Table 4.7</i>	<i>Uncertainty in demand side – Figure 4.25</i>	<b><i>Uncertainty in hyper parameter setting – Table 5.7, Figure 5.16</i></b>	<i>Uncertainty in parameter setting in solution algorithm (Starting point of searching)</i>	Uncertainty in price (Price of agriculture products)
Type II	<i>Uncertainty in outflow (water release target) – Table 4.5</i>	<i>Uncertainty in supply side - Table 4.15</i>	<b><i>Uncertainty in user preferences – Table 5.6</i></b>		<i>Promotion effort and timing</i>
Type III			<b><i>Uncertainty in model approximation due to heuristics in approximation – Table 5.5</i></b>	<i>Uncertainty in model approximation (ways of combining multiple goals)</i>	
Type IV				<i>Uncertainty in using domain knowledge to simplify the model (fixing decision variables and selecting design scenarios)</i>	<i>Interventions that change the mathematical relation among promotion and result (developing local market)</i>

RD – robust design

M – method

EVe – empirical verification of the method

T – test problem

### 5.5.3 Role of Chapter 5

In this chapter, we use parameter learning to improve the adaptive linear programming (ALP) algorithm. In ALP, one critical parameter, the reduced move coefficient (RMC), that has severe impact on the approximation performance, is determined using golden section search, and no mechanism of obtaining insight to improve the approximation during the search, which may result in missing the sub-range of RMC value with good approximation performance; the best RMC value is sensitive to design scenario changes; no criteria other than the fulfillment of the goals are taken into account when evaluating the approximation performance.

To improve the ALP, we hypothesize that by incorporating parameter learning in the ALP, as ALPPL, we can improve the approximation performance, especially for multi-goal engineering-design problems. To verify the hypothesis, we implement the parameter learning in three steps – identifying the criteria for approximation performance evaluation, developing evaluation indices (EIs), and using them to tune the RMC.

With an application of the hot rolling process chain problem, we depict the procedure of applying ALPPL and demonstrate the improvements of ALPPL over ALP. With ALPPL, using different design scenarios, we initialize the parameters and use them to tune the RMC and update them during the tuning. We validate that the ALPPL helps find the RMC value facilitating the identification of more robust solutions, and the insensitive range of RMC gets explored more sufficiently.

The proposed algorithm ALPPL can be applied to multi-goal engineering-design problems, especially when goals conflict with one another, the priority of the goals evolves with the environment changing, and the outputs of the model need to be insensitive to model errors and variations.

The three-step procedure of rule-based parameter learning can be used to improve other algorithms, especially when there are no customizable criteria for evaluation of the algorithm performance, or the algorithm performance is highly sensitive to some critical parameters that are determined with heuristics or human intuition while the critical parameters do not get updated based on the algorithm performance during the design iterations.

# CHAPTER 6 TYPE I, III, & IV ROBUST DESIGN THROUGH UNSUPERVISED LEARNING

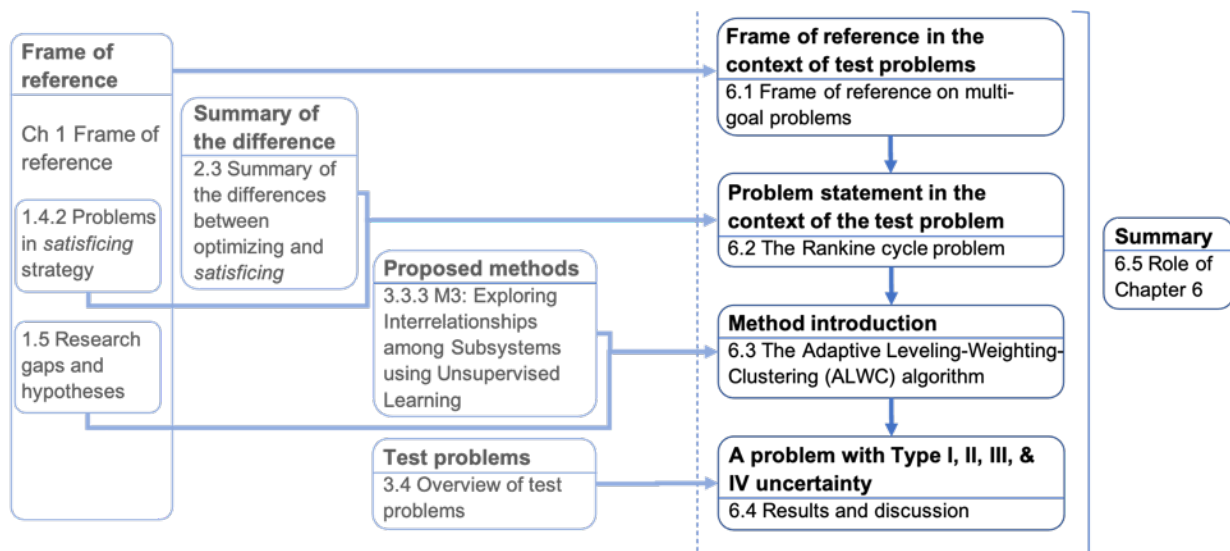
## – ADAPTIVE LEVELING-WEIGHTING-CLUSTERING ALGORITHM (ALWC)

### *The new knowledge in Chapter 6:*

*An algorithm using data analyses to facilitate knowledge discovery for managing many-goal problems*

*– Adaptive Leveling-Weighting-Clustering (ALWC) algorithm*

In Chapter 6, see Figure 6.1: in Section 6.1, the reference on the modeling constructs and solution algorithms for multi-goal or multi-objective problems is framed, which is an extension of Section 1.2 model strategies and their foci; in Section 6.2, the Rankine cycle problem is described; in Section 6.3, based on the research gaps described in Section 1.5 and the Method 3 proposed in Section 3.3.3, we introduce the Adaptive Leveling-Weighting-Clustering (ALWC) algorithm in details; in Section 6.4, the ALWC is applied to the Rankine cycle problem for awareness of subsystems and learning the interrelationship among them; in Section 6.5, summarized the role of Chapter 6.



**Figure 6. 1 Organization of Chapter 6**

The plan of specifying and answering Research Question 3 in the context of the test problems is shown in Table 6.1. In Chapter 6, the Proposed Method 3 (M3), the Adaptive Leveling-Weighting-Clustering (ALWC) algorithm, is empirically verified (EVe3) using a test problem, designing a Rankine cycle thermal system (T3). Research Question 3 (RQ3) is specified in the context of the test problem (SQT3) and answered (AQ3) by testifying M3. The empirical validation and theoretical validation are in Chapters 8 and 9.

**Table 6.1 Plan of Specifying Research Question 3 (RQ3) and Empirically Verifying the Adaptive Leveling-Weighting-Clustering (ALWC) Algorithm (M3)**

Chapter	Ch1	Ch 2	Ch 3	Ch 4-7			Ch 8	Ch 9
				Ch 4-5	Ch 6	Ch 7		
Actions	RG H	RD RQ SH	TVe M		<i>EVe3: use a concurrent, multi-goal problem with uncertainties and unknown features in the interrelationships among the subsystems to verify SH3 and demonstrate M3.</i>	EVe SQT AQ	CQ EVa	TE
				EVe 1-2 SQT 1-2 AQ 1-2	<i>SQT3: What is the method that facilitates managing multiple conflicting goals and exploring the tradeoffs of the performance of multiple sub-systems?</i>			
					<i>SQT3.1: What are the methods that facilitate the exploration of the structure of the goals if there are more than three goals in a system?</i>  <i>SQT3.2: How can the knowledge in the post-solution analysis be obtained and used to support design improvement if the domain knowledge is missing?</i>			

					<p><i>AQ3: Using the ALWC algorithm, with increasing weight vectors, the interrelationship among goals based on their deviations, or achievement rates are evolved and converged. Based on their interrelationship, goals are grouped into clusters to represent different subsystems. The combinations of the goals are explored iteratively, by using either the Pre-emptive or Archimedean strategy. This facilitates assigning each cluster a different level (leveling) and combining the goals in each level using weight vectors (weighting). Through iteration more design scenarios are identified, and the corresponding solutions are obtained for designers to choose the appropriate design scenario and thence improve the design. As a tool to acquire insight when domain knowledge is lacking, the combination of the goals is explored so that better solutions regarding the average deviations, standard deviations, worst case and Euclidean distance to the Utopia point are identified, whilst the computational complexity is reduced.</i></p>			
Nomenclature	RG – give research gaps							
	H – give hypotheses							
	RD – tie to roust design							
	RQ – pose research questions							
	SH – specify hypotheses							
	TVe – theoretically verify hypotheses							
	M – introduce methods							
	EVe – empirically verify hypotheses							
	SQT – specify research questions in the context of test problems							
	AQ – answer research questions							
	CQ – closure the answers to research questions							
	EVa – empirically validate hypotheses							
	TE – theoretically extend the research							

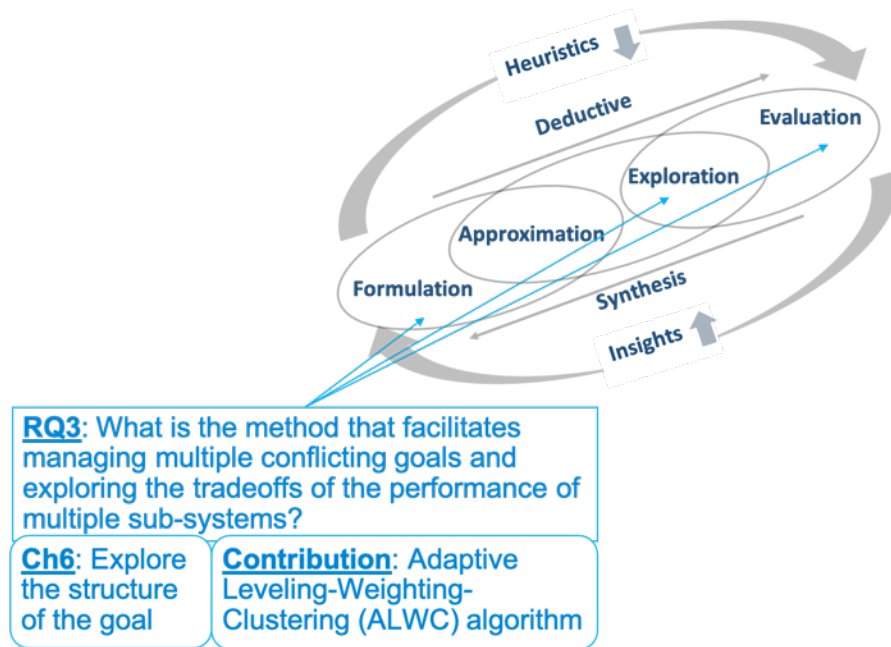
In this chapter, the Adaptive Leveling-Weighting-Clustering (ALWC) algorithm is proposed and tested by using a concurrent engineering design problem – the thermal system around a Rankine cycle. The Research Question 3 (RQ3) “What is the method to speed up learning the system nature?” is specified regarding the Rankine cycle design problem (T3) as follows (SQT3) and then further specified into two sub-questions indicating the tasks and answered.

***SQT3: What is the method that facilitates managing multiple conflicting goals and exploring the tradeoffs of the performance of multiple sub-systems?***

***SQT3.1: What are the methods that facilitate the exploration of the structure of the goals if there are more than three goals in a system?***

***SQT3.2: How can the knowledge in the post-solution analysis be obtained and used to support design improvement if the domain knowledge is missing?***

It is hypothesized that interactions among the formulation, exploration, and evaluation of a design problem should be studied and the mechanisms of information sharing and intervention between the three procedures are established. See Figure 6.2.



**Figure 6. 2 Specified Research Question 3 and the Relevant Stages to be Connected in Design Evolution Cycle**

In this chapter, we address the issue of solving a many goal decision problem, that is, a problem with more than three goals. There are limitations in managing many-goal problems documented in the published literature, such as the need for domain expertise to combine the goals. In this chapter, we propose a domain-independent method, Adaptive Leveling-Weighting-Clustering (ALWC), to manage the process of the exploration of the design scenarios of many-goal, concurrent design problems. Using the ALWC, designers can explore the combination of the goals based on their interrelationship iteratively. Through iteration, design scenarios with better

achievement rates of the goals are obtained without increasing the computational complexity. Further, knowledge of the relationship between subsystems is gleaned. This knowledge is useful for concurrent design. The ALWC algorithm is illustrated using a thermal-system design problem as a test problem. The focus in this chapter is on the method rather than the results.

As the complexity of manufacturing systems is increasing rapidly, designers are facing challenges in dealing with many-goal concurrent design problems – the design problems with more than three goals, associated with diversity and robustness of the solution space and computational efficiency. In this chapter, we address the issue of solving a many goal decision problem. There are limitations in managing many-goal problems documented in the published literature, such as the need for domain expertise to combine the goals. Moreover, whether the goals represent the interest of subsystems. By learning the correlation or orthogonality among the goals, can designers obtain knowledge on the interrelationship among the subsystems? Given the designers aware the subsystems, can they reorganize those subsystems based on their interrelationship to boost the system's performance?

As information in a solution space can improve design, there is a requirement of passing through the information from post-solution analysis to the design modification, in a systematic, iterative manner. Hence, a method that facilitates information exchange between solution space and design space is needed. In this chapter, it is proven that the many-goal problems can be managed by exploring the combination of the goals. The possible structures can be Preemptive (leveling), Archimedean (weighting), or the combination of the two in various forms.

To address the specified research questions, it is proposed an algorithm, the Adaptive Leveling-Weighting-Clustering (ALWC) algorithm. Using ALWC, the structure of the goals is explored by running three loops – leveling, weighting and clustering. With the ALWC, the information from

the post-solution analysis can be used to improve the design and make corresponding decisions, and this process can take place independent with any domain knowledge.

### **Glossary (mainly applied in Chapter 6)**

Archimedean	A strategy of managing multiple goals by compromising the achievement of the goals. Also known as weighted sum function. Goals are weighted combined to form a compromise goal (Ignizio 1976, Guéret, Prins et al. 1999).
Concurrent design	An integrated design method where a designer should use various methods, technologies, and strategies (Gao, Wang et al. 2012).
Deviation	Deviations are measured from these goals both above and below the target. Unwanted deviations from this set of target values are then minimised in an achievement function <sup>26</sup> .
Goal	In this chapter, the term “goal” is used to refer to the objective of a design problem when its target value is determined. The problem is solved by minimizing the deviation between the achieved value and the target value of the goals. In other words, the problem is solved by maximizing the goal achievement rate.
Goal achievement rate	The rate at which the target of a goal is achieved.
Many-goal problems	Problems with more than three goals.
Pre-emptive	A strategy of managing multiple goals by decentralizing the problem. Also known as Lexicographic (Kortanek and Maxwell 1969). The goals are placed at multiple levels of priority. The first level goal function will be satisfied as far as possible and then while holding it within a tolerance; the second level goal function will be addressed, and so on in an attempt to address all the goals across all levels (Ignizio 1976).
Robustness	The capability of a system to be insensitive to variations or uncertainties.
<i>Satisficing</i>	A decision-making strategy or a cognitive heuristic that entails searching through the available alternatives until an acceptability threshold is met (Byron 1998).

---

<sup>26</sup> From Wikipedia, Goal Programming:

[https://en.wikipedia.org/wiki/Goal\\_programming#:~:text=Goal%20programming%20is%20a%20branch,criteria%20decision%20analysis%20\(MCDA\).&text=Deviations%20are%20measured%20from%20these,minimised%20in%20an%20achievement%20function.](https://en.wikipedia.org/wiki/Goal_programming#:~:text=Goal%20programming%20is%20a%20branch,criteria%20decision%20analysis%20(MCDA).&text=Deviations%20are%20measured%20from%20these,minimised%20in%20an%20achievement%20function.)



<i>Satisficing</i> solutions	The solutions that are not necessarily optimal but good enough by minimizing the distance between what the system can achieve and what the ideal case should be (Simon 1996).
Scalarization	The reduction of multiple goals to a single function that can be solved using a single goal. Two most common scalarization methods are lexicographic ordering (Pre-emptive) and weighted sum function (Archimedean).

## 6.1 Frame of Reference on Multi-Goal Problems

### 6.1.1 Features of Concurrent Engineering Problems

In concurrent design, a designer needs to consider the interactions and coupling effects between subsystems and tradeoffs among conflicting requirements concurrently (Wang 1994). Furthermore, the designer must simultaneously meet multidisciplinary constraints, such as cost, physical properties, manufacturing capacity, etc., meaning that there can be different units of the goals and the evaluation of the tradeoffs may not be simple; hence, the *satisficing* solutions are desired, which meet multiple design preferences and are relatively insensitive to certain uncertainties (Wang, Nellippallil et al. 2018).

In concurrent design, there is a need to infuse knowledge of downstream activities into the design process so that model formulation and approximation can be improved iteratively. The aim is to explore the *satisficing* solution space and enhance the design regarding the goal achievement rate, robustness, and computational complexity. In this chapter, design improvement is realized by exploring the combination of the goals.

### 6.1.2 Two Categories of Studies on Multi-Goal Problems

In this chapter, as an extension of Section 1.2, we categorize multi-goal problems into two categories – the performance of the solution algorithms and the indication in the design improvement. In the papers on the solution algorithms, the main focus is on sorting near-Pareto

front (Deb, Pratap et al. 2002, Seada and Deb 2014), and the performance of an algorithm is evaluated by criteria such as the optimality of the solutions, diversity of the solutions, and the computational complexity of the algorithm (Soltani, Tawfik et al. 2002); whereas in the papers on the design improvement, the authors focus on improving the design or acquiring more knowledge about the problem (Tang, Zhu et al. 2010). Some typical methods in both categories are shown in Table 6.2. In this chapter, the combination of the two emphases is addressed.

**Table 6.2 The Features and Limitations of Some Classic Multi-Objective (Multi-Goal) Solution Algorithms and Methods**

Emphases	Algorithm or method	Description	Emphases			
			Searching for interior solutions	Having adaptable ways of producing more solutions (offsprings)	Giving information of tradeoffs between multiple characteristics (such as solution accuracy vs convergence speed)	Use the knowledge of tradeoffs between goals (objectives) to improve model formulation
Solution algorithms	VEGA (Schaffer 1985)	Using vector-valued feedback with adaptive procedures for searching high-order multi-objective problems		*		*
	SPEA2 (Zitzler, Laumanns et al. 2001)	Using fitness assignment, archiving and truncating (the near-Pareto front) to evolve solutions to approach the Pareto-optimal set	*	*	*	
	MOEA/D (Zhang and Li 2007)	Decomposing a problem into scalar optimization subproblems and optimizing them simultaneously	*	*	*	*
	NSGA-II/III (Deb, Pratap et al. 2002, Seada and Deb 2014)	Using the nondominated sorting evolutionary algorithm to adaptively update reference points to approach the Pareto front	*	*		*
	REDGA (Jaimes, Coello et al. 2009)	Reducing the number of objectives by removing redundant (to some degree) objectives			*	*

	HypE (Bader and Zitzler 2011)	Using Monte Carlo simulation to approximate the exact hypervolume values and seeking ranking of solutions	*	*	*		
	Multi-Level Decisions (Mistree, Patel et al. 1994)	Using two design strategies and multi-level decisions to foster discussion on multi-objective problems				*	*
	RCEM (Chen, Allen et al. 1997, Choi, Austin et al. 2005)	Improving the robustness of the design using indices based on the results of exploring the solution space					*
Design improvement	Interval analysis (Hao and Merlet 2005)	Using parallel robots based on interval analysis to determine geometries to satisfy all compulsory requirements					*
	Level diagrams (Reynoso-Meza, Blasco et al. 2013)	Comparing multiple Pareto fronts based on different design concepts using level diagrams, to support decision making on design concept selection	*	*			*
	XPLORE in DSIDES (Smith, Milisavljevic et al. 2015, Sabeghi, Shukla et al. 2016)	Exploring the design space by exploring different goal structures using a compromise Decision Support Problem				*	*
	CORTHOOG (Warwick 2019)	Removing poor measurement degrees-of-freedom iteratively until pseudo-orthogonality check was optimized	*	*			*

The limitation of the literature focusing on the solution algorithms is summarized as:

The authors focus on identifying near Pareto front. The improvements of the solution algorithms over past decades are mainly on a better spreading of solution points along the near Pareto front, and a faster identification. Discussions and decision supports on how the solutions in different parts of the near Pareto front can be used are missing.

The limitation of the literature focusing on design improvement is:

Relying on domain knowledge or case-by-case analyses to explore the combination of the goals or decomposing of the problems, which make the methods less generic and less reusable.

Given such limitations in both categories of the literature, in this chapter, data analysis is applied to obtain insight to provide decision support in the combination of goals, especially when domain

knowledge is insufficient. The hypothesis is that by exploring the combination of the goals based on their interrelationship, the achievement rate of each goal can be improved, more reasonable and diverse design scenarios can be discovered, and the process of knowledge discovery and reuse is domain independent. In this chapter, goals are defined as the objectives with target values as right-hand sides and the design problem is managed by minimizing the deviation between the left-hand side and the target of each goal.

### ***6.1.3 Differences between a Goal and an Objective***

There are differences between a goal and an objective. A goal represents an objective with the right-hand side value as the target value to be achieved (Parra, Terol et al. 2001). In this chapter, we discuss goals instead of objectives. For a nonlinear problem, when we maximizing the objective without a right-hand side, the solution  $x^*$  is optimal, meeting both necessary and sufficient Kuhn-Tucker conditions; when we minimize the distance between left-hand side and right-hand side of the goal, the solution  $x^s$  is *satisficing*, meeting only necessary Kuhn-Tucker conditions.

The essence of the necessary conditions is that at a solution point  $x^s$ , where both the primal and the dual are feasible, the gradient vector of the objective (which is the left-hand side of the goal) can be represented as the linear combination of the gradient matrix of all equality constraints and the active inequality constraints<sup>27</sup>. The essence of the sufficient conditions is that at a solution point  $x^*$ , the convexity degree of the objective should not exceed the convexity degree of the constraints combined by Lagrange multipliers.

---

<sup>27</sup> An active inequality constraint is a constraint with its left-hand side equals to its right-hand side at the solution point.

To avoid “no solution” caused by the strong convexity of the objective, or to avoid losing a solution due to uncertainties that breaks the Kuhn-Tucker conditions, we obtain *satisficing* solutions by using goals instead of objectives. A target value  $t$  as the right-hand side of the objective is assigned thereby the objective  $f(\mathbf{x})$  becomes a goal  $G(\mathbf{x})$ , whose position is fixed in the solution space. In  $\mathcal{F}$ , the feasible space bounded by all active constraints, the point, or several points, or an area, that is/are on the goal  $G(\mathbf{x})$  or closest to it (using Euclidean distance in this chapter) are the *satisficing* solution(s)  $\mathbf{x}^s$ , see

$$G(x): f(x) + d^- - d^+ = t, \text{ where } 0 \leq d^-, d^+ \leq 1, d^- \cdot d^+ = 0 \quad \text{and} \quad \mathbf{Error!}$$

**Reference source not found..** Using deviation variables  $\mathbf{d} = (d^-, d^+)$  to measure the under-achievement and over-achievement of a goal versus its target and minimizing the deviation variables, we make the goal  $G(\mathbf{x})$  become an equality constraint to be satisfied, the deviation variables  $\mathbf{d}$  become decision variables that form the new objective  $\mathbf{z}(\mathbf{d})$ , and the original decision variables  $\mathbf{x}$  become auxiliary variables that do not show up in the objective. The combination form of merit function  $\mathbf{z}(\mathbf{d})$  is discussed in Section 6.1.4.

$$G(x): f(x) + d^- - d^+ = t, \text{ where } 0 \leq d^-, d^+ \leq 1, d^- \cdot d^+ = 0 \quad \mathbf{Equation 6. 1}$$

$$\mathbf{Minimize} \mathbf{z}(\mathbf{d}) \quad \mathbf{Equation 6. 2}$$

Such a construct is known as the compromise Decision Support problem (cDSP) (Mistree, Hughes et al. 1993).

**Given:**  $\mathcal{P}, t$

**Find:**  $x, d$

**Satisfy:**

$$x \in \mathcal{F}: \{g(x) \geq 0, h(x) = 0, \text{lower bound} \leq x \leq \text{upper bound}\},$$

$$G(x), 0 \leq d^-, d^+ \leq 1, d^- \cdot d^+ = 0$$

**Minimize:**  $z(d)$

For a n-dimension, K-goal problem, by adding deviation variables, we increase the dimensionality of a design problem, from  $[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T$  to  $[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{d}_1^-, \mathbf{d}_1^+, \mathbf{d}_2^-, \mathbf{d}_2^+, \dots, \mathbf{d}_K^-, \mathbf{d}_K^+]^T$ , thus make it possible to absorb the risk of uncertainty that breaks the second-order sufficient conditions. This results in a robust solution, a solution that is relatively insensitive to uncertainties. By returning solutions consisting only of the original decision variables,  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T$ , we decrease the dimensionality. Such “dimension expansion and reduction” ensures a solution that is relatively insensitive to the uncertainties embodied in the modeling of an optimization problem.

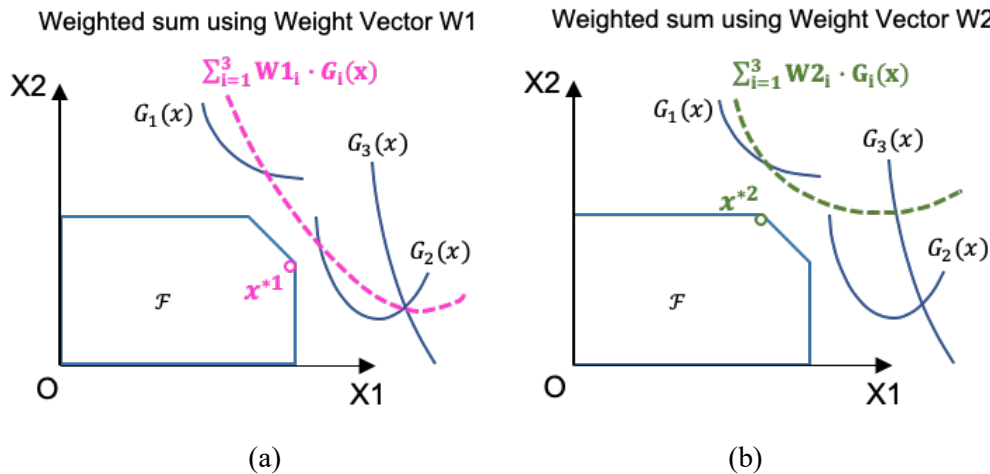
Using the cDSP construct, the nonlinear problem is linearized using the Adaptive Linear Programming (ALP) algorithm (Mistree and Kamal 1985, Mistree, Hughes et al. 1993), so that the linearized problem can be solved using Simplex algorithm and a vertex solution is obtained. The benefits of using a vertex solution versus an interior point solution are 1) a vertex solution is a good enough solution that guarantees the closest distance to the target, and 2) it does not require computing power to search for better interior point solutions.

#### **6.1.4 Common Ways of Combining the Goals $z(d)$**

A common approach to solve nonlinear multi-goal problems is to define a scalarizing function (Bandaru, Ng et al. 2017), which is applied to many-goal problems as well. A scalarizing function is used to combine all the goals to form a single function. The solution is a compromise among the achievement, or deviation, of all goals. The most popular scalarization is Archimedean strategy,

also known as weighted sum function; see Equation 6.3. Sometimes, multiple weight vectors are used to combine the goals, so multiple solutions are obtained to be selected by the designers based on different situations; see Figure 6.3.

**Minimize:**  $\sum_{k=1}^K W_k \cdot (d_k^- + d_k^+)$  **Equation 6. 3**

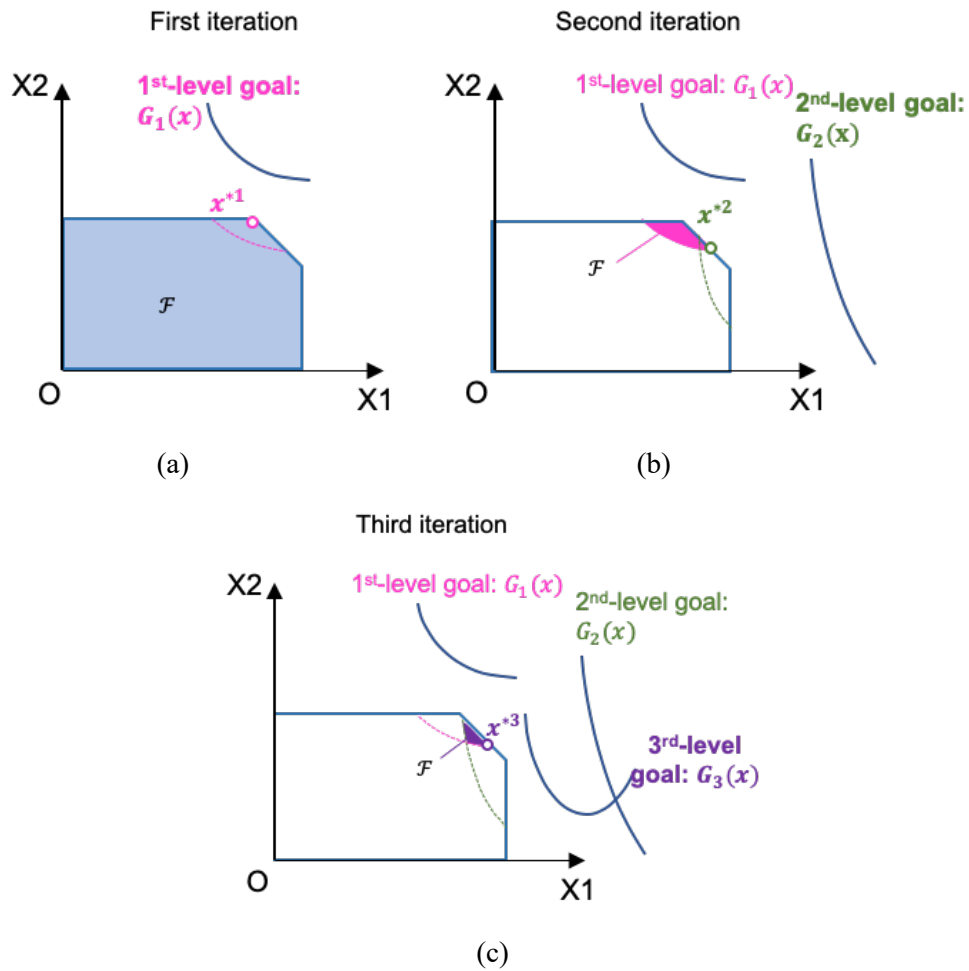


**Figure 6. 3 Archimedean Strategy (Weighted Sum)**

However, an even distribution of solutions in the solution space is not guaranteed by using an even distribution of weight vectors (Messac and Mattson 2002). To determine weight vectors appropriately, prior knowledge on the priority and expected tradeoff among the goals is required, and a weight sensitivity analysis can be done (Vevea and Woods 2005). However, the weight sensitivity analysis is computational expensive, and the metaheuristics applied in weight generation and selection are not generic as the interdependence among the goals may vary from problem to problem (Seada and Deb 2014). A generic priori technique that integrates scalarization and domain knowledge is necessary.

Another priori approach for managing multi-goal or many-goal problems is Pre-emptive strategy, also known as Lexicographic ordering, that is converting all goals but one to constraints in iterations (Rae 1972); see Figure 6.4. The goals are placed at multiple levels of priority. The first

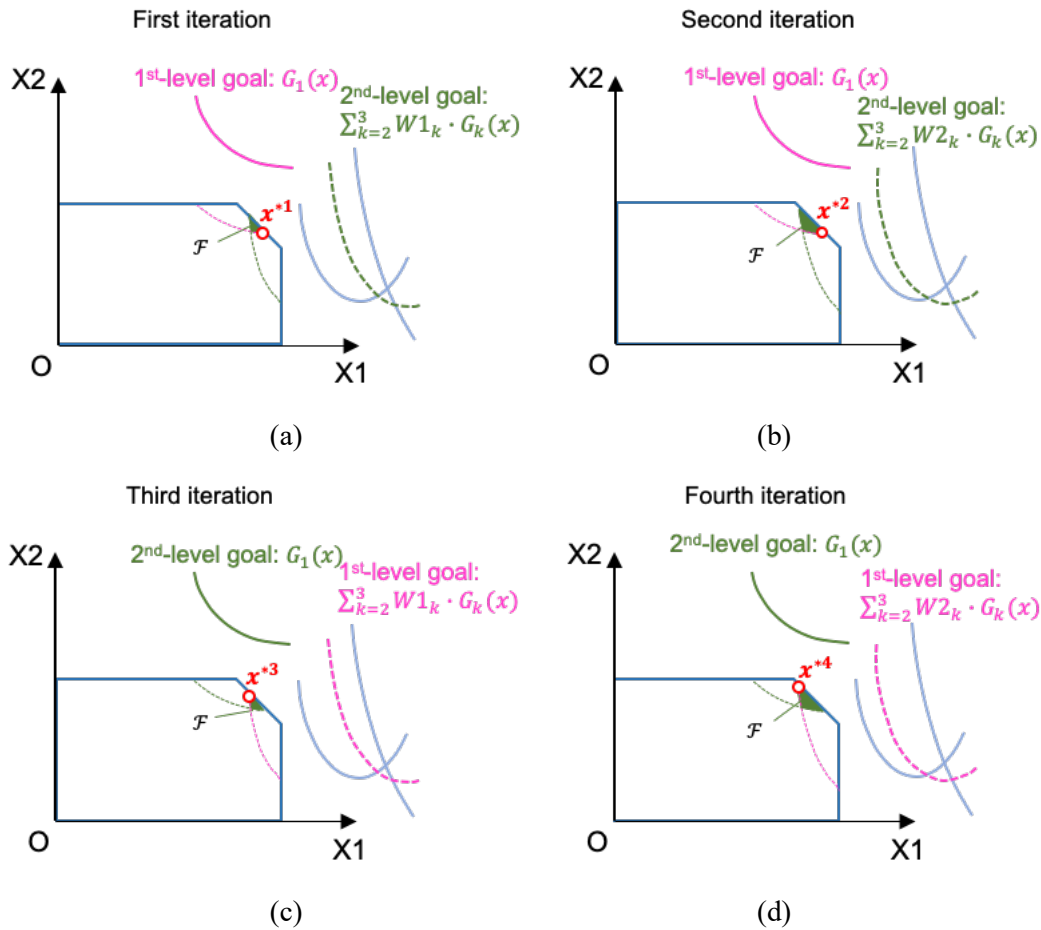
level goal function will be satisfied as far as possible and then while holding it within a tolerance; the second level goal function will be addressed, and so on in an attempt to address all the goals across all levels (Ignizio 1976). Level settings can be switched. Like Archimedean strategy, some domain knowledge on the priority among the goals are needed when using Pre-emptive strategy, otherwise the computational cost is high – for a  $K$ -goal problem, if each goal is set to one level, the number of design scenarios for prioritization is  $K!$ . It requires designers to have insight regarding design preferences and interrelationship among goals, to avoid exploring unnecessary design scenarios.



**Figure 6. 4 Pre-Emptive Strategy (Lexicographic Ordering)**



An ensemble or comprehensive strategy using a mixture of Archimedean and Pre-emptive strategy is also a choice; see Figure 6.5. In each level, there can be one or multiple goals. The goals in the same level are combined as a single goal using weight vectors. The levels are switched so that the solution space can be explored more sufficiently based on various prioritizing scenarios. The drawback of ensemble strategy is that insight on the problem is also required to effectively explore the design scenarios, otherwise, the permutation and combination of the leveling and weighting increases exponentially as the goals increase.



**Figure 6. 5 An Ensemble Strategy using a Mixture of Archimedean and Pre-emptive Strategy**

In this chapter, we want to use data analysis to fill in the deficiencies of domain knowledge, or to avoid applying too many heuristics without evaluating or improving them. We answer this question regarding knowledge management in dealing with multi-goal problems:

*What is the domain-independent method to capture and reuse the knowledge of a many-goal, concurrent-engineering-design problem to facilitate the exploration and selection of design scenarios?*

In Section 6.2, we introduce a thermal-system design as a test problem. In Section 6.3, we propose a knowledge-driven method, the adaptive leveling-weighting-clustering (ALWC) algorithm, to

manage the knowledge capturing and reusing in multi-goal problems. In Section 6.4, we use the test problem to demonstrate the utility of the ALWC algorithm and present the results and discussion. In Section 6.5, summarized the contributions of this chapter and the scope of the application of the ALWC algorithm.

## **6.2 Problem Statement – Test Problem 3: The Rankine Cycle Problem**

### ***- Test Problem 3: apply ALWC to a concurrent, multi-goal problem***

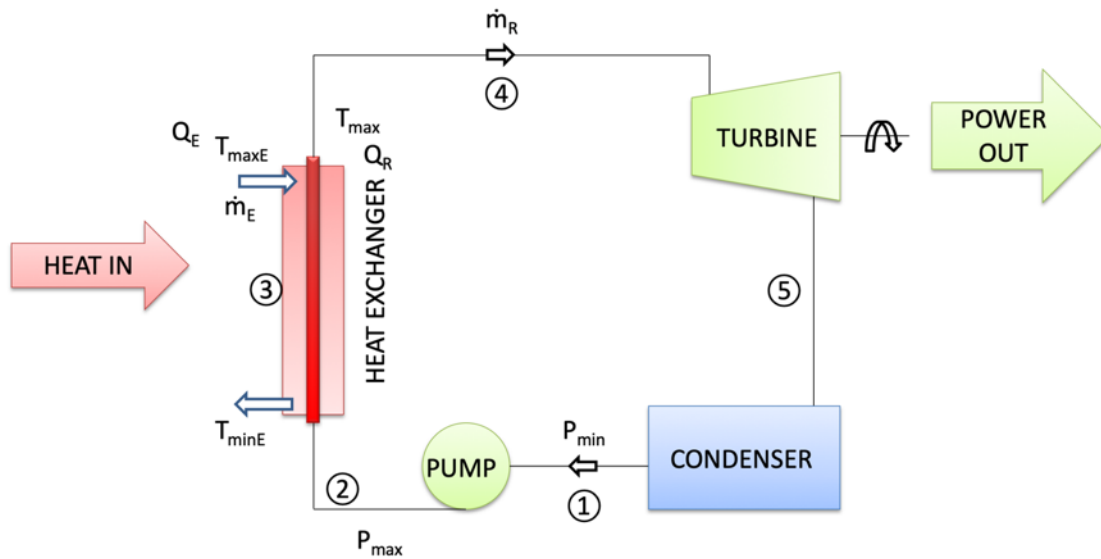
We use an example to demonstrate how a concurrent, multi-goal problem looks like – a thermal system design problem. The problem is first published in (Smith, Milisavljevic et al. 2015).

#### ***6.2.1 Problem Description***

There can be various applications for small scale “power” plant systems that run small generators to produce electricity or that make direct mechanical use of the power produced. For example, providing power to equipment in irrigation systems, driving reverse osmosis systems to produce fresh water for underdeveloped areas, and generating electricity for general use in small collectives.

Building a system around the Rankine cycle is a common approach given an available heat source. The Rankine cycle is a mathematical representation of a heat engine that converts heat into mechanical work while undergoing phase change (Macquorn Rankine 1853, Wikipedia 2019). A schematic representation of the Rankine cycle, where the major components of the system are a power producing turbine, a pump to pressurize the flow to the turbine and two heat exchangers, a condenser, and a heater is shown in Figure 6.6.

In the perspective of decision support and design improvement, such a thermal system presents complexity and dilemmas to be managed and resolved. Expansion within the laboratory will deal with heat source issues (left side of Figure 6.6) and power use issues (right side of Figure 6.6) and the choice of working fluids. The common working fluid in a Rankine cycle is water.



**Figure 6. 6 The Thermal System**

The foundational example model is defined by the cycle's maximum and minimum pressures and maximum temperature (P<sub>MAX</sub>, P<sub>MIN</sub> and T<sub>MAX</sub>). Energy is transferred to the closed loop Rankine cycle through a heat exchanger. The heat exchanger is assumed to be of a counter flow design where the key characteristic is the maximum temperature of the heating flow (T<sub>MAXE</sub>).

### 6.2.2 Model Formulation

The ideal Rankine cycle involves 4 processes, as shown graphically in the Temperature (T) versus Entropy (S) plot in Figure 6.7. There are two adiabatic isentropic processes (constant entropy) and two isobaric processes (constant pressure).

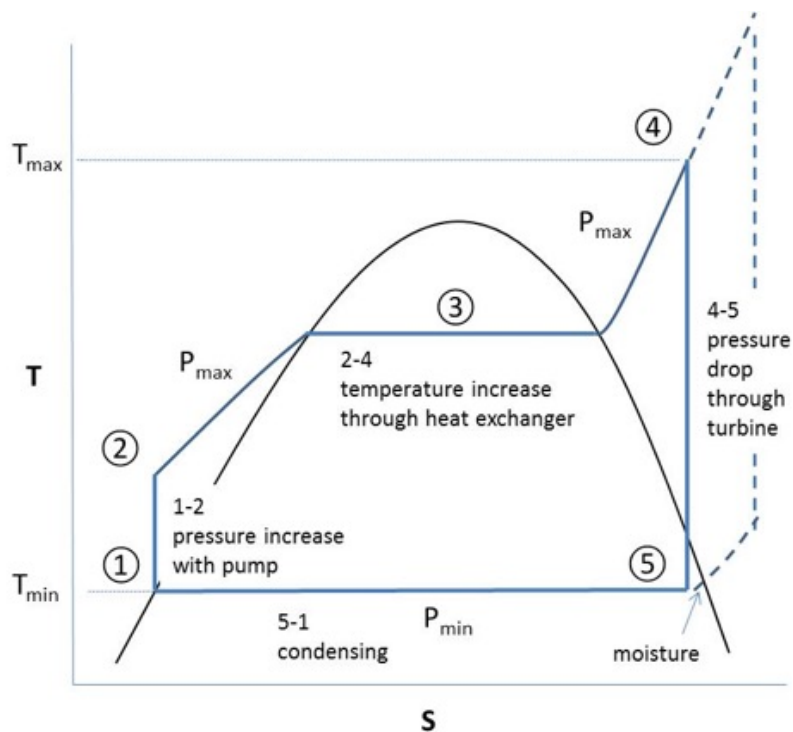
The Rankine cycle consist of 4 cycles, referring to Figure 6.7,

①-② adiabatic pumping of the saturated liquid from  $P_{\min}$  to  $P_{\max}$

②-④ isobaric heat addition in heat exchanger to  $T_{\max}$ ,

④-⑤ adiabatic expansion in the turbine from  $P_{\max}$  to  $P_{\min}$  producing power with the possibility of wet steam exiting the turbine, and

⑤-① isobaric heat loss in the condenser.



**Figure 6. 7 Rankine Cycle (Temperature and Entropy)**

The isothermal segments represent moving from saturated liquid to saturated vapor in the case of ③ in the heater and the reverse in the condenser between ⑤-①. The key thermodynamic properties of the working fluid(s) are determined using REFPROP (Lemmon and Huber 2013). The basic features of the problem: there are four decision variables, one linear constraint, nine nonlinear inequality constraints, and six nonlinear goals. The cDSP is given as follows.

**GIVEN**

**Parameters including dependent system variables P1-P51 (units in abbreviated SI<sup>28</sup>)**

<i>CARNOT</i>	Carnot cycle efficiency (%)	Parameter 1 (P1)
<i>CPEE</i>	Specific heat value for input line in exchanger	P2
<i>CPRE</i>	Specific heat value for Rankine (output) feedline in exchanger (J/(kg·K))	P3
<i>DBTMNR/DBTMNE</i>	= 273.16	P4/P5
Lower temperature limit (freezing point) in the Rankine cycle ( <i>DBTMNR</i> ) and in the heat exchanger ( <i>DBTMNE</i> ) for every fluid (K)		
<i>DBTMXR/DBTMXE</i>	= 2000.0	P6/P7
Upper temperature limit in the Rankine cycle ( <i>DBTMXR</i> ) or in the heat exchanger ( <i>DBTMXE</i> ) for every fluid (K)		
<i>DELTLM</i>	Logarithmic main temperature difference (K)	P8
<i>DENSi</i>	$i = 1, 2, \dots, 5$ , Density at ①-⑤ (kg/m <sup>3</sup> )	P9, P10, P11, P12, P13
<i>EDIA/ELEN</i>	Diameter/length of heat exchanger (m)	P14/P15
<i>ENTHi</i>	$i = 1, 2, \dots, 5$ , Specific enthalpy at ①-⑤ (J/kg)	P16, P17, P18, P19, P20
<i>ENTHMX/ENTHMN</i>	Enthalpy at TMINE/TMAXE in exchanger	P21/P22
<i>FLOWR/FLOWE</i>	Mass flow rate of Rankine cycle / exchanger (kg/s)	P23
<i>FRMXR</i>	The upper limit of Rankine cycle mass flow rate (kg/s)	P24
<i>HTEFF</i>	Heat transfer effectiveness (%)	P25
<i>PPUMP/PTURB</i>	Power of the pump/turbine (W)	P26
<i>PRESi</i>	$i = 1, 2, \dots, 5$ , Pressure at ①-⑤ (kPa)	P27, P28, P29, P30, P31
<i>QINR</i>	Heat transfer in the heat exchanger (W)	P32
<i>QOUTE</i>	Exchanger heat transfer (W)	P33
<i>QUALi</i>	$i = 1, 2, \dots, 5$ , Quality of stream at ①-⑤ (%)	P34, P35, P36, P37, P38
<i>RCEFF</i>	Rankine cycle efficiency (%)	P39
<i>RCMIT</i>	Rankine cycle moisture in turbine (%)	P40
<i>REQPOW</i>	Required power at the Rankine cycle (kW)	P41
<i>RFEEDL</i>	Calculated Rankine cycle length required given diameter (m)	P42
<i>SAREAE</i>	Surface area of the heat exchanger (m <sup>2</sup> )	P43
<i>SYSEF1</i>	System efficiency 1 (%)	P44
<i>STSEF2</i>	System efficiency 2 (%)	P45
<i>TDELE=10</i>	Requirement of minimum temperature change in the heat exchanger (K)	P46
<i>TDELC</i>	Minimum temperature gap between the minimum temperature in the heat exchanger and the temperature at ②	P47
<i>TEFFEX</i>	Temperature exchanger efficiency (%)	P48
<i>TEMPi</i>	$i = 1, 2, \dots, 5$ , Temperature at ①-⑤ (K)	P49
<i>TMINE</i>	Minimum temperature in exchanger (K)	P50
<i>UHTC</i>	Overall heat transfer coefficient (W/(m <sup>2</sup> ·K))	P51

**Functional relationship between parameters and system variables F1-F14**

$CARNOT = 1.0 - \frac{TEMP1}{TEMP4}$	Function 1 (F1)
$CPEE = \frac{ENTHMX - ENTHMN}{TMAXE - TMINE}$	F2

<sup>28</sup> International System of Units

$$\begin{aligned}
CPRE &= \frac{ENTH4-ENTH2}{TEMP4-TEMP2} & F3 \\
DELTLM &= \frac{(TMINE-TEMP2)-(TMAXE-TEMP4)}{\ln\left(\frac{TMINE-TEMP2}{TMAXE-TEMP4}\right)} & F4 \\
FLOWE \cdot (ENTHMX - ENTHMN) &= FLOWR \cdot (ENTH4 - ENTH2) & F5 \\
FLOWR &= \begin{cases} FRMXR & \text{if } EMP4 - TEMP5 = TEMP2 - TEMP1 \\ \frac{REQPOW}{(TEMP4-TEMP5)-(TEMP2-TEMP1)} & \text{otherwise} \end{cases} & F6 \\
HTEFF &= 1.0 - e^{-\frac{UHTC \cdot SAREAE}{FLOWR \cdot CPRE}} & F7 \\
PPUMP &= (ENTH2 - ENTH1) \cdot FLOWR & F8 \\
PPUMB &= (ENTH4 - ENTH5) \cdot FLOWR & F9 \\
QINR &= FLOWR \cdot CPRE \cdot (TEMP4 - TEMP2) & F10 \\
QOUTE &= FLOWE \cdot CPEE(TMAXE - TMINE) & F11 \\
RCEFF &= \frac{PTURB-PPUMP}{QINR} & F12 \\
RCMIT &= \begin{cases} 0 & \text{if } QUAL5 > 1 \\ 1 & \text{if } QUAL5 < 0 \\ 1 - QUAL5 & \text{if } 0 \leq QUAL5 \leq 1 \end{cases} & F13 \\
RFEEDL &= \frac{SAREAE}{\pi \cdot I} & F14 \\
SAREAE &= \pi \cdot EDIA \cdot ELEN & F15 \\
SYSEF1 &= \frac{PTURB-PPUMP}{QOUTE} & F16 \\
SYSEF2 &= RCEFF \cdot TEFEX & F17 \\
TEFFEX &= \frac{TMAXE-TMINE}{TMAXE-TEMP2} & F18 \\
UHTC &= \frac{QINR}{SAREAE \cdot DELTLM} & F19
\end{aligned}$$

## **FIND**

### ***x*, the decision variables (system variables) *x1-x4***

<i>P</i> MAX	Maximum pressure in the Rankine cycle	Variable 1 ( <i>x</i> 1)
<i>P</i> MIN	Minimum pressure in the Rankine cycle	<i>x</i> 2
<i>T</i> MAX	Maximum temperature in the Rankine cycle	<i>x</i> 3
<i>T</i> MAXE	Maximum temperature of the heating fluid in the exchanger	<i>x</i> 4

### **Deviation variables *d***

$d_k^-, d_k^+$	$k = 1, 2, \dots, 6$ , Under-achievement and over-achievement of Goal $k$	$d$
----------------	---	-----

## **SATISFY**

### **The system constraints**

#### **Linear constraints C1**

$$TMAXE - TMAX \geq DELTLM$$

Temperature delta (10 K) for maximums in exchanger    Constraint 1 (C1)

#### **Nonlinear constraints C2-C10**

$RCMIT \leq TMXL$	Moisture in turbine (RCMIT) less than upper limit (TMXL)	C2
$FLOWR \leq FRMXR$	Rankine cycle mass flow rate (FLOWR) less than upper limit (FRMXR)	C3

$TEMP4 \geq TEMP3$ (TEMP3)	Temperature at ④ (TEMP4) should be greater than or equal to temperature at ③ (TEMP3)	C4
-------------------------------	--	----

$QUAL4 \geq 1.0$	Quality at ④ (QUAL4) is superheated vapor	C5
$TMAXE - TMINE \geq TDELE$	TMAXE is greater than TMINE by at least TDELE	C6
$TMINE - TMEP2 \geq TDELC$	TMINE is greater than temperature at ② by at least TDELC	C7
$CARNOT \geq SYSEF1$	Ideal Carnot cycle efficiency is greater than System efficiency 1 (Sanity check 1)	C8
$CARNOT \geq SYSEF2$	Ideal Carnot cycle efficiency is greater than system efficiency 2 (Sanity check 2)	C9
$DBTMXE \geq TMAXE$	Temperatures within valid ranges for REFPROP fluid	C10
<u>The system variable bounds (<math>x_j^{min} \leq x_j \leq x_j^{max}</math>):</u>		
$500 \leq PMAX \leq 5000$ (kPa)		Bound 1,2 (B1, B2)
$350 \leq TMAX \leq 850$ (K)		B3, B4
$350 \leq TMAXE \leq 850$ (K)		B5, B6
<u>The system goals:</u>		
Goal 1: Achieve zero moisture in steam leaving the turbine (Minimize the moisture or maximize the steam quality of ①)		
$RCMIT + d_1^- - d_1^+ = 0$		Goal 1 (G1)
Goal 2: Maximize Rankine cycle efficiency		
$RCEFF + d_2^- - d_2^+ = 1$		G2
Goal 3: Maximize temperature exchanger efficiency		
$TEFFEX + d_3^- - d_3^+ = 1$		G3
Goal 4: Maximize system efficiency indicator 1		
$SYSEF1 + d_4^- - d_4^+ = 1$		G4
Goal 5: Maximize system efficiency indicator 2		
$SYSEF2 + d_5^- - d_5^+ = 1$		G5
Goal 6: Maximize heat transfer effectiveness in exchanger		
$HTEFF + d_6^- - d_6^+ = 1$		G6

## MINIMIZE

### The design scenario DS to be explored

$z_{DS}(d)$  DS

There are limitations of the method used in (Smith, Milisavljevic et al. 2015). Using Preemptive approach, the six goals are placed at six levels of priority. By prioritizing the goals differently, comparison may show competing goals driving the solution in different directions. Using Archimedean approach, the six goals are grouped at the same level and linearly combined using weights. There can be a mixture of Preemptive and Archimedean approach, but the authors of (Smith, Milisavljevic et al. 2015) did not explore the mixture form. With the proposed method in this dissertation, the ALWC algorithm, the knowledge on the mixture-form of the goals can be



explored, captured, and reused for other problems. We use this problem assuming there is no knowledge on the division and interrelationship of the goals or subsystems. We apply the ALWC algorithm to learn such knowledge and use the domain expertise in (Smith, Milisavljevic et al. 2015) to verify our findings.

### 6.3 The Adaptive Leveling-Weighting-Clustering (ALWC) Algorithm

#### 6.3.1 Clustering the Goals based on their Interrelationship

In managing a many-goal, concurrent design problem, to explore the effective combinations of the goals  $z(d)$ , the interrelationship of the goals, such as the correlation or orthogonality, should be identified, and the scalarization function or priori of the goals should be determined based on their interrelationship. Any way to combine the goals of a many-goal problem is a design scenario (DS), such as an Archimedean way using a weight vector, or a Pre-emptive way using an order to prioritize the goals, or any way to mix the two strategies. Under a design scenario,  $DS_\alpha$ , the merit function is denoted as  $z_{DS_\alpha}(d)$ , the corresponding *satisficing* solution is  $x_\alpha^s$ , and the deviation of Goal  $k$  is  $d_{\alpha k}$ . For a  $K$ -goal problem, if we solve it using  $A$  design scenarios,  $[DS_1, DS_2, \dots, DS_A]^T$ , where  $A$  is a positive integer, we get an  $A \times K$  matrix of deviations,  $\mathcal{D}$ ; see Figure 6.8. The interrelationship that we learn by analyzing Matrix  $\mathcal{D}$  can be interpreted as an indication of the conditional correlation or conditional orthogonality<sup>29</sup>, that is within the feasible space  $\mathcal{F}$  and under  $A$  DSs, the correlation or the orthogonality of the  $K$  goals; see Figure 6.9. The learning algorithms

---

<sup>29</sup> The correlation analysis and orthogonality analysis are two examples of the analyses illustrated in this chapter to learn the interrelationship among goals. There are other types of analysis that can be applied to capture the interrelationship among goals. Designers can select their own methods and make customization based on the characteristics of their problems and their preferences.

and cluster analysis methods can be expanded according to the requirements of the problems. In this chapter, the knowledge discovery and decision support relevant to the innovation, selection, and customization of learning algorithms and cluster analysis methods are not our focus, however, it can be addressed and improved using our ALWC algorithm.

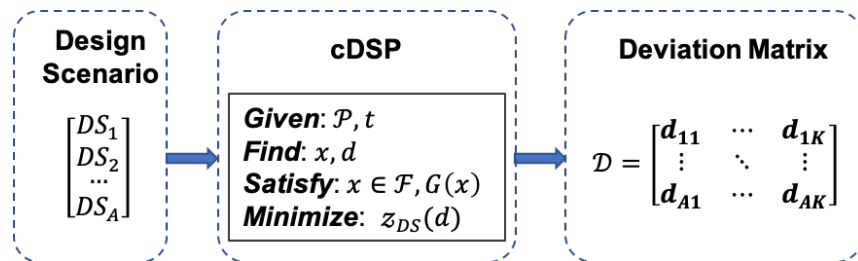


Figure 6. 8 Using Multiple Design Scenarios to Obtain a Deviation Matrix

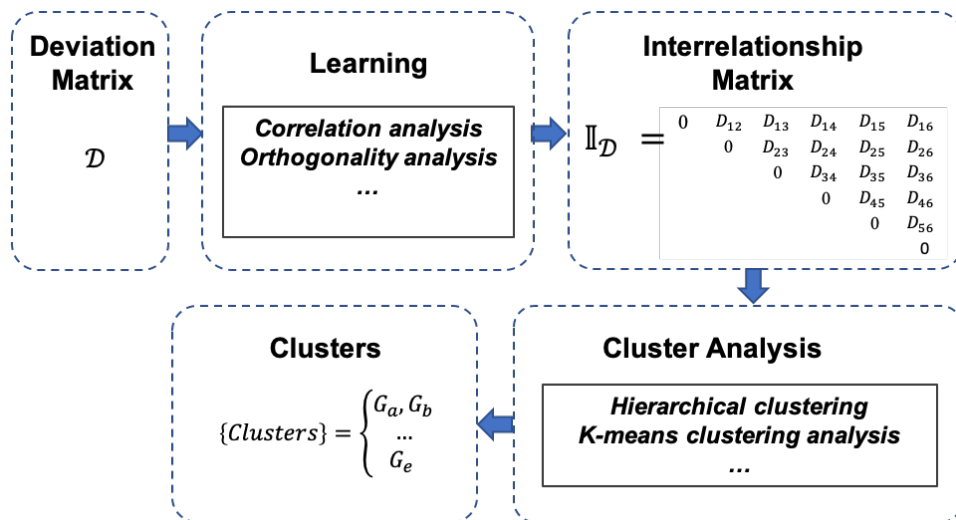


Figure 6. 9 Cluster Analysis Using a Deviation Matrix

*The adaptiveness.* To make the learning algorithm generic, we assume there is no domain knowledge as pre-knowledge on the interrelationship among the goals. In this chapter, as in each iteration, only limited number of design scenarios are generated to obtain solutions, hence, we learn and update the interrelationship by iterating. In this sense, our method is an adaptive method.

*Learning the correlation.* We learn the correlation among the deviation vector of the goals using angle-based distance. Suppose for a three-goal problem, we use two design scenarios DS1 and DS2 to learn the correlation of the goals, the scalarization of each design scenario is represented as  $z_{DS_i}(d)$ ,  $d = [d_k^-, d_k^+]^T$ , where  $i = 1, 2, k = 1, 2, 3$ . The *satisficing* solution and the deviation are represented as Equation 6.4 and 6.5.

$$x^{si} = \arg \left( \min \left( z_{DS_i}(d) \right) \right), \forall i = 1, 2 \quad \text{Equation 6. 4}$$

$$d_{ik} = |t_k - G_k(x^{si})|, \forall i = 1, 2, \forall k = 1, 2, 3 \quad \text{Equation 6. 5}$$

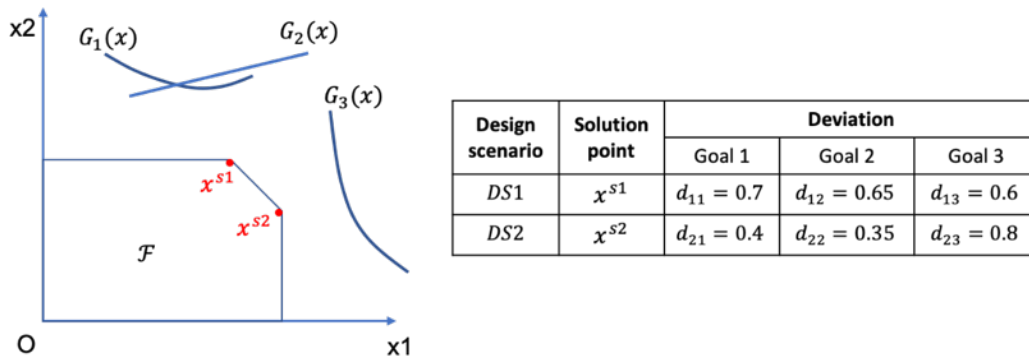
As in Figure 6.10, we use a two-dimensional solution space to illustrate the three goals  $G_k(x)$ ,  $\forall k = 1, 2, 3$ , and two *satisficing* solutions  $x^{si}$ ,  $\forall i = 1, 2$ , under two design scenarios  $DS_i$ ,  $\forall i = 1, 2$ . In Figure 6.11, we show the deviation points in two two-dimensional goal spaces. The coordinate origin O is the utopia point where the deviation of both goals is zero, whereas the Point I (1, 1) is the worst point where both goals are completed by 0%. When design scenario changes from DS1 to DS2, the deviation of Goal 1 and Goal 3 change in different directions, Figure 6.11 (a), whereas the deviation of Goal 1 and Goal 2 change in the same direction, Figure 6.11 (b). Therefore, based on the results of these two design scenarios, Goal 1 and Goal 3 are with a lower correlation, so they belong to two different clusters, whilst Goal 1 and Goal 2 have a higher correlation, so they belong to one cluster. As  $\alpha_{ij}$ , the acute angle between  $\overline{D_1^{ij} D_2^{ij}}$  and the diagonal  $\overline{OI}$  of the goal space, indicates the correlation between Goal i and Goal j, we use angle-based correlation analysis to obtain the interrelationship between the two goals (Equation 6.6). As more design scenarios being used, the correlation among goals can be better learned. For A number of design scenarios, if we use  $\alpha_{ij}(DS_\kappa, DS_l)$  to denote the acute angle between  $\overline{D_\kappa^{ij} D_l^{ij}}$ , the deviation coordinate of Goal i and Goal j under two design scenarios,  $DS_\kappa$  and  $DS_l$ , and we use the average

of the enumerated angle-based correlation under any two design scenarios as the correlation between Goal  $i$  and Goal  $j$ , between the two goals, it is shown as Equation 6.7.  $D_{ij\_cor}$  are elements of correlation matrix  $\mathbb{I}_{\mathcal{D}}$ , therefore, we obtain the matrix using angle-based correlation with  $A$  design scenarios as Equation 6.8. As we update the design scenarios in each iteration, we update  $\mathbb{I}_{\mathcal{D}}$  along iterating. Using  $\mathbb{I}_{\mathcal{D}}$  to cluster the goals, when clustering results converge, we can stop iterating.

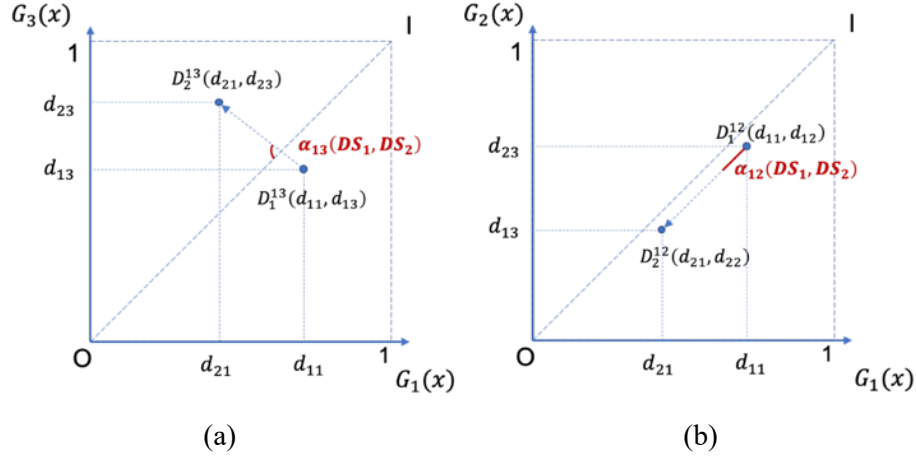
$$D_{ij\_cor}(DS_1, DS_2) = \text{arc}(\sin(\alpha_{ij})), \forall i, j = 0, \dots, K \quad \text{Equation 6.6}$$

$$D_{ij\_cor}(DS_1, \dots, DS_A) = \frac{1}{C_A^2} \sum_{\kappa=1}^{A-1} \sum_{l=\kappa+1}^A \text{arc}(\sin(\alpha_{ij}(DS_{\kappa}, DS_l))), \forall i, j = 0, \dots, K \quad \text{Equation 6.7}$$

$$\mathbb{I}_{\mathcal{D}}(\text{correlation}, [DS_1, \dots, DS_A]^T) = (D_{ij\_cor}(DS_1, \dots, DS_A)) \quad \text{Equation 6.8}$$



**Figure 6.10 The Satisficing Solutions to a Three-Goal cDSP under Two Design Scenarios Illustrated in a Two-Dimensional Solution Space**



**Figure 6. 11 The *Satisficing* Solutions to a Three-Goal cDSP under Two Design Scenarios Illustrated in Two Two-Dimensional Goal Spaces**

*Learning the orthogonality.* When learning the correlation of the goals, we treat the deviation vector of the goals as statistics variables and induce the correlation of the goals by analyzing the correlation among the statistic variables; whereas when learning the orthogonality among the goals, we obtain the relative position of goals in geometric space, under the constraints of the feasible region  $x \in \mathcal{F}$ , by using the dot-product of the deviation vector of any two goals; see Equation 6.9. In Figure 6.11 (a) and Figure 6.11 (b), we visualize the dot-product of the deviation vector of two goals, Goal  $i$  and Goal  $j$ , and Goal  $i$  and Goal  $j$  using two design scenarios, DS1 and DS2. The dot-product is calculated as Equation 6.10. For Euclidean distance,  $p = 2$ . If the dot-product is zero, the two goals are orthogonal, or perpendicular, see Figure 12 (a); if the dot-product is a large value, the two goals are more parallel, see Figure 12 (b).  $D_{ij\_ort}$  are elements of orthogonality matrix  $\mathbb{I}_D$ , therefore, we obtain the matrix using orthogonality with  $A$  design scenarios as Equation 6.11.

$$\{ \langle \mathbf{Goal } i, \mathbf{Goal } j \rangle \mid x \in \mathcal{F} \} = \int_{x \in \mathcal{F}} \overline{\mathbf{G}_i(x) \mathbf{G}_j(x)} dx \approx \mathbf{D}_{ij\_ort}(\mathbf{DS}_1, \dots, \mathbf{DS}_A) = [\mathbf{d}_{1i}, \dots, \mathbf{d}_{Ai}] \cdot [\mathbf{d}_{1j}, \dots, \mathbf{d}_{Aj}]^T$$

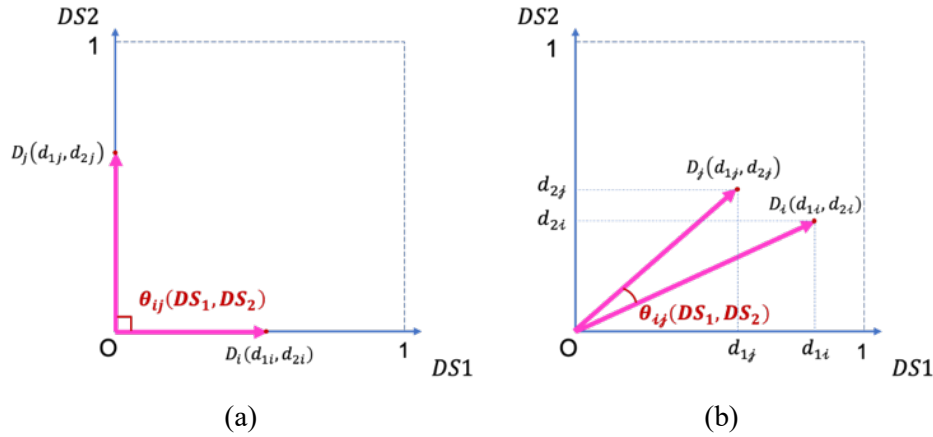
**Equation 6. 9**

$$D_{ij_{ort}}(DS_1, \dots, DS_A) = \left( \sum_{\kappa=1}^A d_{\kappa i}^p \right)^{\frac{1}{p}} \left( \sum_{\kappa=1}^A d_{\kappa j}^p \right)^{\frac{1}{p}} \cos \left( \theta_{ij}(DS_1, \dots, DS_A) \right)$$

Equation 6. 10

$$\mathbb{I}_{\mathcal{D}}(\text{orthogonality}, [DS_1, \dots, DS_A]^T) = \left( D_{ij_{ort}}(DS_1, \dots, DS_A) \right)$$

Equation 6. 11



**Figure 6. 12 The Orthogonality between the Deviation Vectors of Two Goals using Two Design Scenarios**

*Cluster analysis based on interrelationship matrix of the goals.* In this chapter, we use multiple clustering algorithms and apply cross validation to ensure the rationality of the clustering results. There are criteria for designers to choose appropriate clustering algorithms, such as the sensitivity of the clustering results to sample size (the number of design scenarios), the improvement of goal achievement and diversity of the solutions by leveling the goals using a clustering result, etc. The selection of the clustering algorithms is a part worthy of studying regarding improving the ALWC algorithm. We demonstrate the feasibility of further improving the selection of clustering algorithms.

### 6.3.2 A Schematic of the ALWC Algorithm

The ALWC includes two loops; see Figure 6.13. Leveling-Weighting-Clustering is the outer loop, and Weighting-Clustering is the inner loop. In the outer loop, leveling starts with the clustering result  $\{C, \text{Cluster}\}$  from the previous iteration<sup>30</sup>. In the very first iteration, leveling starts with the initialized single cluster so all goals are set to one level. In the later iteration, when there are more than one clusters return from the previous clustering, each cluster is set to a level. After running all processes in this leveling setting, including the weighting and clustering in the inner loop, the levels are alternated, that is the goals in each cluster are alternately set to a different level. When it enters the inner loop, the goals in each level are combined using weight vectors. After being solved using the ALP, the deviation matrix  $\mathcal{D}$  is obtained, so the interrelationship matrix  $\mathbb{I}_{\mathcal{D}}$  is obtained and used to do cluster analysis, and a new  $\{C, \text{Cluster}\}$  is returned to update the leveling for the next iteration. The inner loop stops generating more weight vectors when the diversity of  $\mathbb{I}_{\mathcal{D}}$  do not increase much, that is  $\sigma(\mathbb{I}_{\mathcal{D}}) \leq \varepsilon$ , where  $\varepsilon$  is a threshold determined with heuristics, as more weight vectors being used. The outer loop stops leveling the goals based on the latest clustering result  $\{C, \text{Cluster}\}$  if it does not change in the previous  $\eta$  iterations, where  $\eta$  is a positive integer determined with heuristics. When the outer loop stops, convergence is reached.

---

<sup>30</sup>  $C$  is the number of clusters. Cluster is a two-dimensional array containing the clusters of the goals. E.g., Cluster =  $[[G1, G2, G4], [G3, G5, G6]]$  means that there are two clusters: Cluster 1 include Goal 1, 2, and 4, whereas Cluster 2 include Goal 3, 5, and 6. Cluster[i][j] represent the  $j^{\text{th}}$  goal in the  $i^{\text{th}}$  cluster, so, Cluster[2][3] = G6. Array “Level” works in the same way.

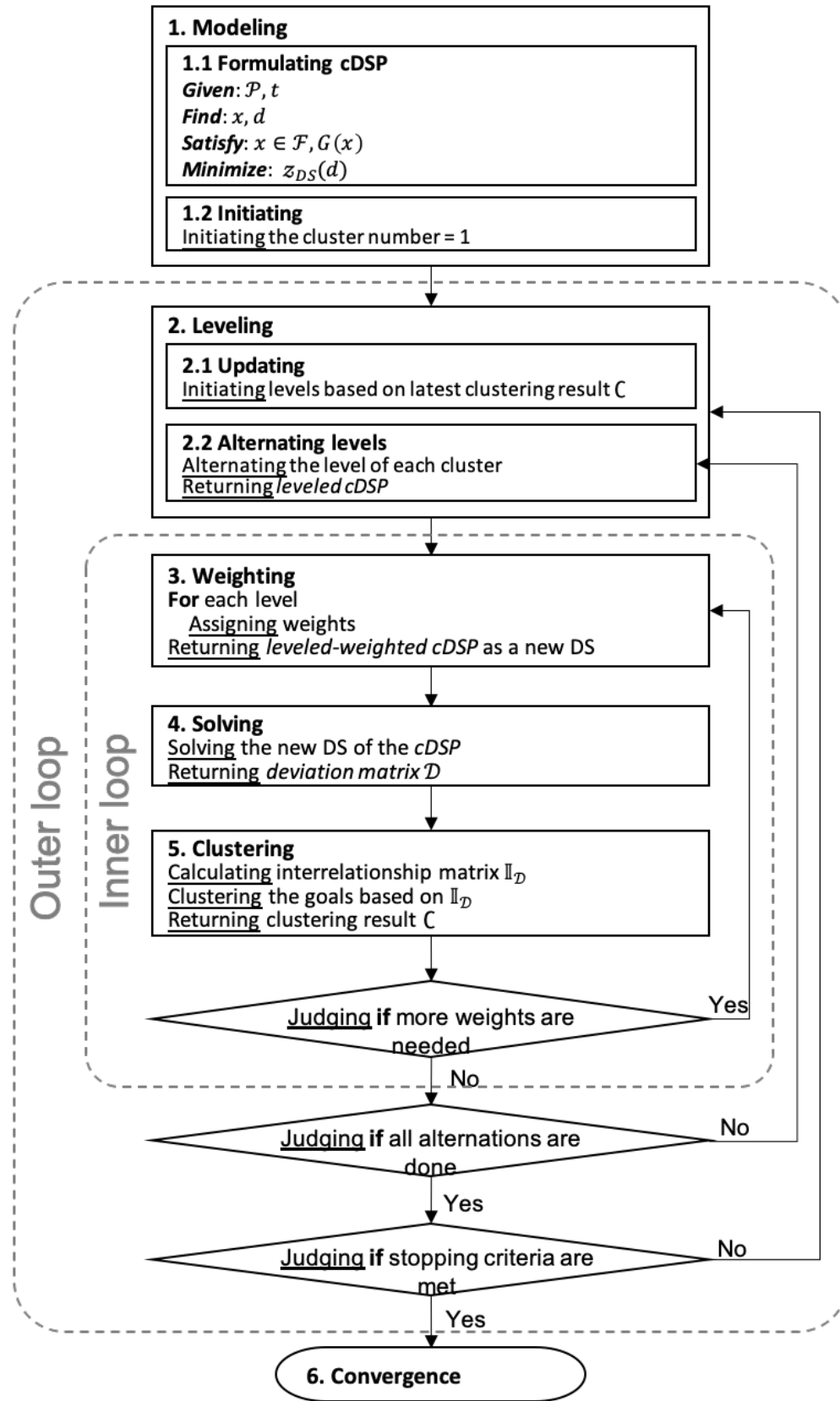


Figure 6. 13 The Flowchart of the Adaptive Leveling-Weighting-Clustering (ALWC) Loop



### 6.3.3 The Algorithms in the ALWC

There are six procedures in the ALWC. The model is formulated as a compromise Decision Support Problem (cDSP) (Mistree, Hughes et al. 1981, Mistree, Hughes et al. 1993) in Procedure 1.1. The number of clusters,  $C$ , is initiated as “1” in Procedure 1.2.

In Procedure 2 Levelling, starting from the second iteration, the goals are levelled based on clustering results – the output of Procedure 5 in the previous iteration. In Procedure 1, the number of levels,  $n$ , is updated. In Procedure 2.2, each cluster take turns to be set to Level 1 to Level  $n$ . The algorithm is given as follows.

---

#### Algorithm 1 Procedure 2 Levelling

---

##### 2.1 Updating

- 1: **Given** Cluster results  $\{C, \text{Cluster}\}$ , cDSP, Levelling information  $\{L, \text{Level}\}$  //  $C, L$  are integers representing the number of the clusters and the number of levels respectively. “Cluster, Level” are two-dimension arrays containing the clustering result<sup>31</sup> and leveling information.
- 2: integer  $L = C$
- 3: array  $\text{Level} = \text{Cluster}$
- 4: **return**  $\{L, \text{Level}\}$  and **go to** Procedure 3 Weighting

##### 2.2 Alternating levels

- 5: **if**  $L \geq 2$
  - 6: array  $\text{temp} = \text{Level}[L]$
  - 7: **for** integer  $n$  in range  $[2, L]$
  - 8:      $\text{Level}[n] = \text{Level}[n - 2]$
  - 9:      $\text{Level}[1] = \text{temp}$
  - 10: **return**  $\{L, \text{Level}\}$  and **go to** Procedure 3 Weighting
- 

---

<sup>31</sup> E.g.,  $\text{Cluster} = [[G1, G2, G4], [G3, G5, G6]]$  means that there are two clusters: Cluster 1 include Goal 1, Goal 2, and Goal 4, whereas Cluster 2 include Goal 3, Goal 5, and Goal 6.  $\text{Cluster}[2][3]$  represent the third goal in Cluster 2, so,  $\text{Cluster}[2][3] = G6$ . Array Level works in the same way.

After the goals are leveled, in each level, the goals are combined using weight vectors. It is proved that an even distribution of solutions in the solution space are not guaranteed by using an even distribution of weight vectors (Messac and Mattson 2002), therefore, in Procedure 3, Weighting, the weight vectors of the goals in each level are generated in a greedy manner, until the deviation matrix  $\mathcal{D}$  does not increase its diversity while adding more weight vectors. In this chapter, we use the standard deviation,  $\sigma(\mathbb{I}_{\mathcal{D}})$ , to evaluate the diversity. The solution points generation method in (Seada and Deb 2014) is improved and applied to generate weight vectors in this chapter. The algorithm of weighting is a fractal algorithm. For example, if there are three goals in a level, we can use a ternary plot to illustrate how the weights are generated, see Figure 6.14. Each edge of the triangle is an axis representing the weight of a goal. The coordinates of each spot in the triangle represents a weight vector. The sum of the coordinates of any spot is one. The algorithm of weighting is given as follows.

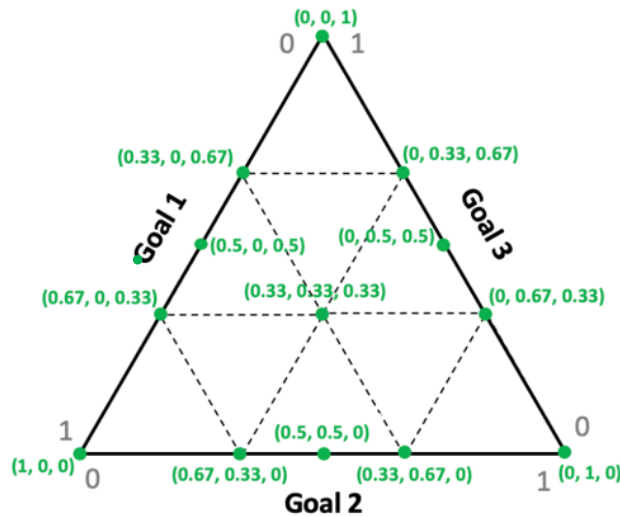


Figure 6. 14 Weight Vectors of a Three-Goal Problem with  $p = 3$

---

**Algorithm 2 Procedure 3 Weighting**

---

---

```

1:  Given {L, Level}, {C, Cluster}i-1, Weight // “Weight” is a three-dimension array containing
    weight vectors32
2:  Initiate parameters:
    integer p = 2, i = 1
    array F = [], wv = [], Weight = [wv],  $\sigma(\mathcal{D}_0) = 0$ ,  $\varepsilon = \mathcal{E}$ 
3:  for integer n in range [1, L]
4:      integer M = length(Level[n]) // Assign the number of goals in Level n as M
5:      while  $|\sigma(\mathcal{D}_i) - \sigma(\mathcal{D}_{i-1})| > \varepsilon$ 
6:          i = i + 1
7:          for integer q in range[1, p]
8:              for integer r in range[0, q]
9:                   $F = F \cup \frac{r}{q}$ 
10:         for I in product(* [F] * M) // “I” is a one-dimension array with three elements that are
            cross-combined by M number of F33
11:             if sum(I) == 1 // If the sum of the elements in “I” equals one.
12:                 wv = I
13:                 Weight[n] = Weight[n]  $\cup$  wv
14:         Call Procedure 4 and return  $\mathcal{D}_i$ 
15:         Calculate  $\sigma(\mathcal{D}_i)$ 

```

---

Weighting iterates while p is increasing and stops if the diversity of the deviation matrix  $\mathcal{D}_i$  is not improved. We use standard deviation to represent the diversity of  $\mathcal{D}_i$ . To remove the variety of the

---

<sup>32</sup> E.g.,  $\text{Weight} = \left[ \left[ [1,0], [0,1], \left[ \frac{1}{2}, \frac{1}{2} \right] \right], \left[ [1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1] \right] \right]$  means that there are two levels; Level 1 has two goals and there are three vectors for the two goals; Level 2 has four goals and there are four weight vectors for the four goals.  $\text{Weight}[n]$  represents the weight vectors of the goals in the  $n^{\text{th}}$  level;  $\text{Weight}[n][m]$  represents the  $m^{\text{th}}$  weight vector of the goals in the  $n^{\text{th}}$  level.  $\text{Weight}[n][m][k]$  represents the weight of the  $k^{\text{th}}$  goal of the  $m^{\text{th}}$  weight. vector in the  $n^{\text{th}}$  level. In this example,  $\text{Weight}[1][3][1] = \frac{1}{2}$ .

<sup>33</sup> E.g.,  $F = [0, \frac{1}{2}, 1]$ ,  $M = 2$ , “I” can be one of the nine arrays:  $[0,0], [0, \frac{1}{2}], [0,1], [\frac{1}{2}, 0], [\frac{1}{2}, \frac{1}{2}], [\frac{1}{2}, 1], [1,0], [1, \frac{1}{2}]$ . “wv” can be  $[0,1], [\frac{1}{2}, \frac{1}{2}]$ , and  $[1,0]$ .

ambitiousness of the target of the goals, that is the deviation range and distribution may vary from one goal to another, we normalize  $\mathcal{D}_i$  into the range  $[0, 1]$  using Equation 6.12.

$$d_{ak_{norm}} = \frac{d_{ak} - \min\{d_{1k}, \dots, d_{Ak}\}}{\max\{d_{1k}, \dots, d_{Ak}\} - \min\{d_{1k}, \dots, d_{Ak}\}} \quad \text{Equation 6.12}$$

**Table 6.3. A part of the normalized deviations of a six-goal cDSP with 81 Iterations with  $L = 3, p = 2$**

Iteration $a$	Leveling-Weighting Scenario			$d_{a1_{norm}}$	...	$d_{a6_{norm}}$
	Level 1 (Weight)	Level 2 (Weight)	Level 3 (Weight)			
$a = 1$	Goal 2, 4 (1,0)	Goal 3, 6 (1,0)	Goal 1, 5 (1,0)	1	...	1
$a = 2$	Goal 2, 4 (0,1)	Goal 3, 6 (1,0)	Goal 1, 5 (1,0)	1	...	1
$a = 9$	Goal 2, 4 $(\frac{1}{2}, \frac{1}{2})$	Goal 3, 6 $(\frac{1}{2}, \frac{1}{2})$	Goal 1, 5 (1,0)	1	...	0.62
$a = 19$	Goal 2, 4 (1,0)	Goal 3, 6 (1,0)	Goal 1, 5 $(\frac{1}{2}, \frac{1}{2})$	1	...	0.83
$a = 27$	Goal 2, 4 $(\frac{1}{2}, \frac{1}{2})$	Goal 3, 6 $(\frac{1}{2}, \frac{1}{2})$	Goal 1, 5 $(\frac{1}{2}, \frac{1}{2})$	0.45	...	0.64
$a = 54$	Goal 3, 6 $(\frac{1}{2}, \frac{1}{2})$	Goal 1, 5 $(\frac{1}{2}, \frac{1}{2})$	Goal 2, 4 $(\frac{1}{2}, \frac{1}{2})$	0.03		0
$a = 81$	Goal 1, 5 $(\frac{1}{2}, \frac{1}{2})$	Goal 2, 4 $(\frac{1}{2}, \frac{1}{2})$	Goal 3, 6 $(\frac{1}{2}, \frac{1}{2})$	0.01	...	0.68

Therefore, the number of leveling-weighting scenarios (or design scenarios, DS),  $A$ , corresponds to each clustering result is given in Equation 6.13.

$$A = \text{length}(\text{DS}) = n! \cdot n \cdot \text{length}(\text{Weight}[n]) \quad \text{Equation 6.13}$$

Essentially, the leveling-weighting is converting a multi-goal cDSP into multiple single-goal cDSPs. The conversion takes place as follows. The goals are grouped into multiple clusters based on their correlation or orthogonality, and the clusters are converted to constraints one-by-one, within a tolerance. It is assumed that the goals with stronger orthogonality or less correlation perform in different trajectory under multiple design scenarios, so the weighted sum of the highly

orthogonal goals to acquire compromise solutions may waste the potential of all subsystems. Therefore, to boost the performance of each subsystem in turn to *satisfice* various situations, we assign the highly orthogonal goals into different clusters and prioritize them alternatively, whereas we assign the more correlated goals in the same cluster and combined them using weight vectors.

To cluster the goals in Procedure 5, we use the interrelationship matrix  $\mathbb{I}_{\mathcal{D}}$  as the input. In this chapter, the correlation analysis (Equation 6.6 and 6.7) and orthogonality analysis (Equation 6.9 and 6.10) are two options for obtaining  $\mathbb{I}_{\mathcal{D}}$  for cluster analysis, and the methods in “hclust” in R, hierarchical clustering, are used, and obtained the clustering results  $\{C, \text{Cluster}\}$ . By applying all methods in hierarchical clustering, different results may be acquired, and the most popular one is returned as  $\{C, \text{Cluster}\}$  that is used in leveling of the next cycle. The algorithm of clustering is given as follows.

---

### Algorithm 3 Procedure 5 Clustering

---

```

1:   Given  $\mathcal{D}_i, \{C, \text{Cluster}\}$ 
2:   Initiate queue = [], temp_result = []
3:    $\mathcal{D}_{i\_norm} \leftarrow$  Normalize  $\mathcal{D}_i$  using Equation 12
4:   Calculate  $\mathbb{I}_{\mathcal{D}}$  based on their correlation (Equation 6, 7) or orthogonality (Equation 9, 10)
5:   for each method in hclust
6:      $\{C, \text{Cluster}\} \leftarrow$  hclust( $\mathbb{I}_{\mathcal{D}}, \text{method}$ )
7:     queue.append =  $\{C, \text{Cluster}\}$  // Add the latest result into an array named “queue”
8:      $\{C, \text{Cluster}\} =$  mode(queue) // Select the clustering result that appears most often
9:   return  $\{C, \text{Cluster}\}$ 

```

---

We use the test problem, Rankine cycle thermal system design, to testify the effectiveness of the ALWC algorithm.

## 6.4 Unsuperfized Learning Results and Discussions

### 6.4.1 Clustering result

By running the ALWC loop applying different interrelationship methods and clustering methods, we obtain clustering results iteratively and list them in Table 6.3. For each interrelationship method, although there is only one clustering result converged, during running the ALWC loop, different clustering results are found. There are three clustering results have ever been identified as a result to update the leveling and we summarize them in Table 4. Our aims of processing the ALWC loop include identifying the best clustering result, obtaining more solutions that better complete the goals or improve the diversity of the deviations, and discover the knowledge on method selection and reuse them. Therefore, we append all solutions and deviations in all iterations to enlarge the solution pool for decision support.

**Table 6. 4 The clustering results along iterations**

Interrelationship	Iteration 1: 1-level			Iteration 2	Iteration 3
Angle-based correlation	6 weight vectors	21 weight vectors	71 weight vectors	22 design scenarios by alternating levels {3, [[1, 6], [2, 4], [3, 5]]}	22 design scenarios by alternating levels {3, [[1], [2, 4], [3, 5, 6]]}
	{3, [[1], [2, 4], [3, 5, 6]]}	{3, [[1, 6], [2, 4], [3, 5]]}	{3, [[1, 6], [2, 4], [3, 5]]} (returned)	{3, [[1], [2, 4], [3, 5, 6]]}	{3, [[1], [2, 4], [3, 5, 6]]} (converged)
Orthogonality	6 weight vectors	21 weight vectors		22 design scenarios by alternating levels {3, [[1, 2, 4], [3, 5], [6]]}	22 design scenarios by alternating levels {3, [[1, 2, 4], [3, 5], [6]]}
	{3, [[1, 2, 4], [3, 5], [6]]}	{3, [[1, 2, 4], [3, 5], [6]]}		{3, [[1, 2, 4], [3, 5], [6]]}	{3, [[1, 2, 4], [3, 5], [6]]} (converged)

**Table 6. 5 The summary of the clustering results ever returned to update the leveling**

Clustering result 1	{3, [[1], [2, 4], [3, 5, 6]]}
Clustering result 2	{3, [[1, 6], [2, 4], [3, 5]]}
Clustering result 3	{3, [[1, 2, 4], [3, 5], [6]]}

### 6.4.2 Improvement in Goal Achievement along the Design Scenario Expansion

For a many-goal problem, because the tradeoffs between two goals often affect other goals, there are too many nondominated solutions and weak dominated solutions, therefore makes it ineffective to use the conception of “solution domination” to rank the solutions, therefore, we use statistics to reflect whether the results are improved and enriched along iterating. In Table 6.5, we show the mean, standard deviation, best (minimum), and worst (maximum) deviations of each of the six goals, under each clustering scenarios, and highlight the best case among all clustering scenarios. To make the results from different clustering scenarios comparable, we use the real deviation value instead of the normalized value. For the 1-level scenario, we use the results of the 71 weight vectors to obtain the statistics because these 71 weight vectors include the 6 and 21 weight vectors in previous iterations of the inner loop. For all the other three clustering scenarios, as each of them has 22 design scenarios, we use them to calculate the statistics respectively. For example, for Goal 1, among the means of each of the four clustering scenarios, the best (smallest) value 0.04 takes place in the third clustering scenario. The numbers in Table 6.5 are rounded but we use the longer floating numbers to do the comparison and highlight the best ones. The key messages from Table 6.5 are stated as follows.

**Table 6. 6 Statistics of the Results**

Statistics	Clustering scenario	$d_k = \max\{d_k^-, d_k^+\}$						Sum $\sum_{k=1}^6 d_k$
		$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$	
Mean	1 level	0.05	0.81	0.36	0.81	0.90	0.02	2.94
	{3, [[1], [2, 4], [3, 5, 6]]}	0.05	0.78	0.51	0.78	0.92	0.01	3.04
	{3, [[1, 6], [2, 4], [3, 5]]}	0.04	0.84	0.34	0.84	0.90	0.01	2.96
	{3, [[1, 2, 4], [3, 5], [6]]}	0.05	0.80	0.45	0.80	0.91	0.01	3.01
Standard deviation	1 level	0.05	0.08	0.41	0.08	0.05	0.04	0.31
	{3, [[1], [2, 4], [3, 5, 6]]}	0.05	0.08	0.38	0.08	0.05	0.01	0.26
	{3, [[1, 6], [2, 4], [3, 5]]}	0.05	0.05	0.24	0.05	0.04	0.01	0.19
	{3, [[1, 2, 4], [3, 5], [6]]}	0.04	0.08	0.37	0.08	0.05	0.01	0.25
Minimum (best case)	1 level	0.00	0.69	0.04	0.69	0.84	0.00	2.67
	{3, [[1], [2, 4], [3, 5, 6]]}	0.00	0.69	0.06	0.69	0.85	0.00	2.72
	{3, [[1, 6], [2, 4], [3, 5]]}	0.00	0.69	0.06	0.69	0.85	0.00	2.72
	{3, [[1, 2, 4], [3, 5], [6]]}	0.00	0.69	0.07	0.69	0.85	0.00	2.73

Maximum (worst case)	1 level	0.12	0.88	0.90	0.88	0.97	0.16	3.52
	{3, [[1], [2, 4], [3, 5, 6]]}	0.12	0.87	0.90	0.87	0.97	0.02	3.31
	{3, [[1, 6], [2, 4], [3, 5]]}	0.12	0.87	0.90	0.87	0.97	0.02	3.29
	{3, [[1, 2, 4], [3, 5], [6]]}	0.12	0.87	0.90	0.87	0.97	0.02	3.29

By leveling the goals using three clustering scenarios, the mean, standard deviation, and the worst-case result of the deviations of five goals but Goal 1 are improved. The worst-case of the sum of all goals is improved. These observations indicate that by clustering and leveling the goals using ALWC algorithm, we identify better design scenarios regarding reducing the deviation (or improving the achievement) of most goals.

There can be other statistics evaluate the iterative results from different perspectives. Designers may select or develop customized statistics based on the characteristics of their problems.

### 6.4.3 Reducing the Euclidean Distance to the Utopia Point

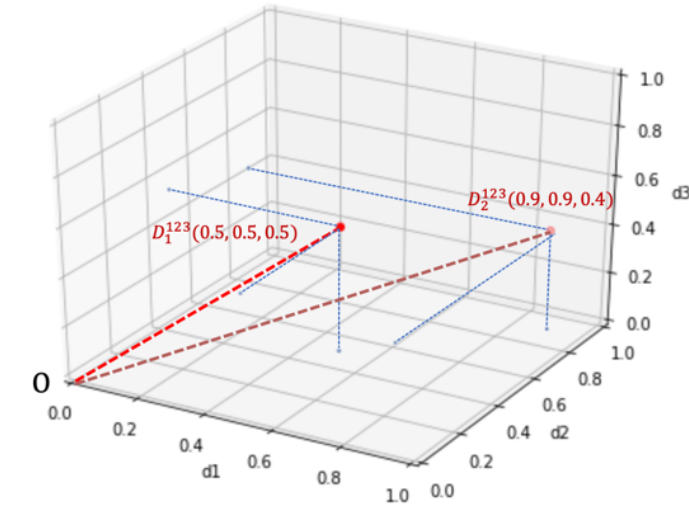
For a many-goal, concurrent design problem, goals may represent the performance of various subsystems of the design. It is possible that the improvement of one goal results in a bigger sacrifice of another goal or several other goals; see Figure 6.15, as Goal 3 is improved 20% at  $D_2^{123}$  versus  $D_1^{123}$ , Goal 1 and 2 get worse by 80%. This may be desired in some situation, but more often than not, designers would rather avoid it because the comprehensive performance of the design may not be practically enhanced by improving one subsystem. Therefore, we use the Euclidean distance to the Utopia point of the deviations, to evaluate the comprehensive performance of the achievement of all goals. Equation 6.14 is the Euclidean distance to the Utopia point of the result from Design Scenario  $a$  for a K-goal problem. We use statistics to evaluate the evolving of the Euclidean distance along the iterating and summarize them in Table 6.6. Using the clustering scenarios obtained along iterating, the mean, standard deviation and the worst case of the Euclidean distance to the Utopia point are improved, whereas the best case is not improved. All the design scenarios and corresponding results from all iterations are added to the solution pool



for designers to select. Designers may customize their post-solution analyses to acquire further decision support on design scenario generation and selection.

$$|OD_a| = \left( \sum_{k=1}^K d_{ak}^2 \right)^{\frac{1}{2}}$$

**Equation 6. 14**



$$|OD_1^{123}| = \left( (0.5)^2 + (0.5)^2 + (0.5)^2 \right)^{\frac{1}{2}} = 0.87$$

$$|OD_2^{123}| = \left( (0.9)^2 + (0.9)^2 + (0.4)^2 \right)^{\frac{1}{2}} = 1.33$$

**Figure 6. 15 An Example of Improving Goal 3 by 20% While Worsening Goal 1 and Goal 2 by 80% Respectively**

**Table 6. 7 Statistics of the Euclidean Distance to the Utopia Point of the Results under Each Clustering Scenario**

	Mean	Standard deviation	Minimum (best case)	Maximum (worst case)
1 level	1.56	0.08	1.45	1.70
{3, [[1], [2, 4], [3, 5, 6]]}	1.57	0.08	1.46	1.66
{3, [[1, 6], [2, 4], [3, 5]]}	1.54	0.06	1.46	1.64
{3, [[1, 2, 4], [3, 5], [6]]}	1.56	0.07	1.46	1.64

#### 6.4.4 Reducing Computational Complexity

Smith and coauthors of (Smith, Milisavljevic et al. 2015) explore the leveling of all six goals, and the theoretical number leveling scenario is  $6! = 720$ . In this chapter, as the goals are leveled based on clustering results, for each clustering scenario, the number of leveling scenarios is reduced to

$3! = 6$ . Within each level, the goals are combined using weight vectors. The stopping criteria of weight vectors generation (Line 5 of Algorithm 2) help prevent designers from using too many unnecessary weight vectors. Using angle-based correlation method to calculate the interrelationship matrix, we explore 142 design scenarios in three iterations and converge; using orthogonality method we explore 71 design scenarios in three iterations and converge. If we use both methods, that is 213 design scenarios. Therefore, the number of design scenarios are reduced from 720 to 213.

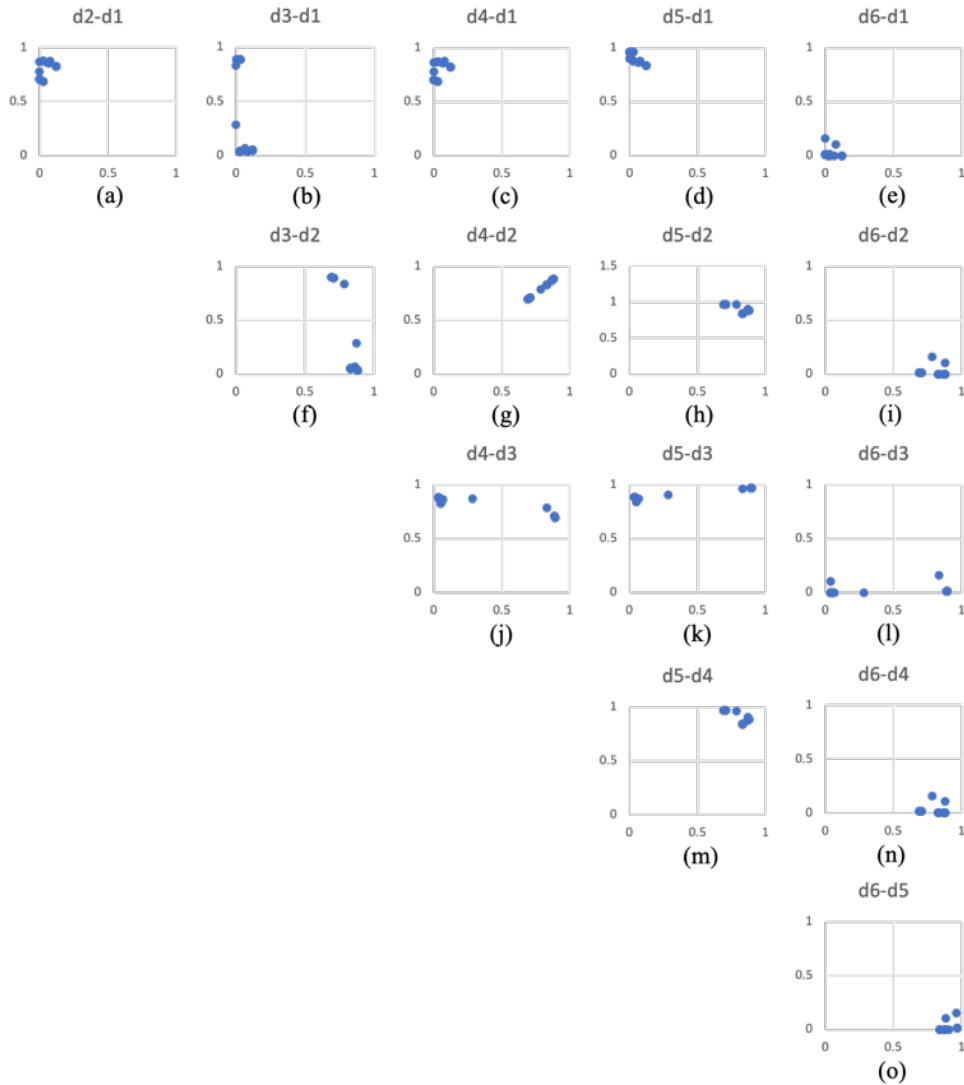
#### ***6.4.5 Verification of the Results***

**The results are verified by domain knowledge.** Among all three clustering scenarios, Goal 2 and Goal 4 are always in one cluster, Goal 3 and Goal 5 are always in one cluster, whereas the clustering result of Goal 1 and Goal 6 are changed along iterating. This means that Goal 2 and Goal 4 are strongly correlated, or weakly orthogonal, and so do Goal 3 and Goal 5; the relationship between Goal 1, Goal 6 and other goals are not significant. The clusters represent the subsystems, which verify the clustering result, see Table 6.8. Goal 2 and Goal 4 represent the Rankine cycle efficiency. It is also verified by the fact that the system efficiency indicator 1 is to boost the Rankine cycle efficiency. Goal 3 and Goal 5 represent the heat exchange efficiency, and the temperature exchanger and the heat transfer work have synergetic effect during system working. Goal 1 is about the moisture in the turbine, and it sometimes forms a single cluster and sometimes clustered with Goal 2 and Goal 4, which is verified by the knowledge that when the moisture in the turbine is low, the Rankine cycle has a high efficiency, but the turbine is a relatively isolated subsystem. Goal 6 is sometimes formed as a single cluster and sometimes clustered with Goal 3 and Goal 5. It is understandable that they all deal with the efficiency of the heat exchanger; Goal 3 and Goal 5 are more about the temperature, whereas Goal 6 deals with the liquid flow. However,

in one clustering scenario, Goal 1 and Goal 6 are in one cluster. The scatter plots of any two goals facilitate (Figure 6.16) us to interpret this phenomenon. The deviations of Goal 1 and Goal 6 are relatively small values and close to the Utopia point O, Figure 6.16 (e), comparing with other two-goal plots. Even after normalizing the deviations in the range [0, 1], it shows that Goal 1 and Goal 6 have a weak correlation. However, after such a clustering scenario being used in the next iteration, the clustering results are “back to normal.” This indicates that the iteratively clustering and updating is necessary because it can expand the sample size and remove bias.

**Table 6. 8 Meaning of the Three Clusters**

<b>Meaning</b>	<b>Representative Goals</b>
Rankine cycle efficiency	<u>Goal 2</u> : Maximize Rankine cycle efficiency <u>Goal 4</u> : Maximize system efficiency indicator 1
Heat exchange efficiency	<u>Goal 3</u> : Maximize temperature exchanger efficiency <u>Goal 5</u> : Maximize system efficiency indicator 2
Moisture in turbine	<u>Goal 1</u> : Moisture in steam leaving the turbine
Heat transfer effectiveness in exchanger	<u>Goal 6</u> : Heat transfer effectiveness in exchanger



**Figure 6. 16 Scatter plots of any two goals using deviations of 1-level, 21 weight vectors**

With domain knowledge, the clustering result is verified for the thermal system design problem. As to the many-goal problems that the domain knowledge or expertise is missing, the ALWC algorithm can facilitate designers identify the interrelationship between the goals. Hence, more design scenarios regarding the combination form of goals are explored based on their interrelationship. The design scenarios and deviations of the goals are added to the solution pool for designers to select to satisfy different requirements, to improve the problem formulation, and to do further analysis to better understand the interrelationship among subsystems. The ALWC

algorithm can be used as a tool to partition a concurrent design problem, especially when information is incomplete.

#### ***6.4.6 Closing Remarks on Using ALWC to Speed up Learning***

In this chapter, we propose a method, the Adaptive Leveling-Weighting-Clustering (ALWC), for exploring multiple design scenarios. We use a test problem (thermal system design) to illustrate the effectiveness of the ALWC algorithm. One question is addressed, and the answer is summarized as follows.

- *What is the domain-independent method to capture and reuse the knowledge of a many-goal, concurrent-engineering-design problem to facilitate the exploration and selection of design scenarios?*

Using the ALWC algorithm, with increasing weight vectors, the interrelationship among goals based on their deviations, or achievement rates are evolved and converged. Based on their interrelationship, goals are grouped into clusters to represent different subsystems. The combinations of the goals are explored iteratively, by using either the Pre-emptive or Archimedean strategy. This facilitates assigning each cluster a different level (leveling) and combining the goals in each level using weight vectors (weighting). Through iteration more design scenarios are identified and the corresponding solutions are obtained for designers to choose the appropriate design scenario and thence improve the design. As a tool to acquire insight when domain knowledge is lacking, the combination of the goals is explored so that better solutions regarding the average deviations, standard deviations, worst case, and Euclidean distance to the Utopia point are identified, whilst the computational complexity is reduced.

The algorithms embodied in the ALWC algorithm can be extended, modified, or customized to specific requirements of a design problem. When there is insufficient expertise in the problem domain to support decision making on the goals, subsystem division and tradeoffs, and design improvement, the knowledge discovered using the ALWC algorithm can be used to explore the ways that may contribute to design improvement.

## **6.5 Role of Chapter 6 in this Dissertation**

### ***6.5.1 Summarizing How We Finish Task 3: Connecting Formulation, Exploration, and Evaluation***

For a complex system composed of subsystems, when designers lack knowledge of subsystem division, the structure of the subsystems, combination status, mutual relationship, etc., the interrelationship among the subsystems cannot be effectively leveraged to improve the efficiency entire system. For the decision model of a concurrent engineering-design problem, the awareness of subsystems and the interrelationship among the subsystems are hidden information that can be learned by analyzing the decision model, the solution space, and the association between design scenarios and the solutions.

The essence of Specific Hypothesis 3, “*learn system nature such as interrelationship among subsystems and reorganize them based on it*” is, to explore multiple design scenarios and obtain a dynamic dataset for analysis, through which the correlation, orthogonality, or other indicators that characterize the degree of contradiction or synergy among the different driving forces in the system, gradually emerge and eventually form a trend. Learning such a trend is helpful in reorganizing the subsystems or selecting the most reasonable and representative design scenarios that maximize the

usage of the interrelationship among the subsystems. Such learning process is to improve the formulation through exploration and evaluation of the model and solution space.

In this chapter, we divide the methods that deal with multi-goal (multi-objective) problems into two categories, focusing on the performance of the solution algorithms and focusing on design improvement. Based on the limitations of both categories – the limitations of Pareto front, especially in engineering-design problems, and, relying on domain knowledge heavily, we propose a method, the adaptive leveling-weighting-clustering (ALWC) algorithm (Figure 6.9 and 6.13) to use the dataset consists of (but not limited to) design scenarios, goal achievement, and solutions to learn the system nature and obtain useful information to improve the formulation. Information such as subsystems and their interrelationships are helpful in identifying the most appropriate and representative design scenarios to reorganize the subsystems and speed up the learning process.

In other words, we realize the connections among formulation, deduction, decision, and action through the activities in the circles, as shown in Figure 6.17. The overview and flowchart of the method is illustrated in Figure 6.9 and 6.13. Through realizing the algorithms 1-3 in Section 6.3.3, we implement the proposed method in the context of the Rankine cycle thermal system problem.

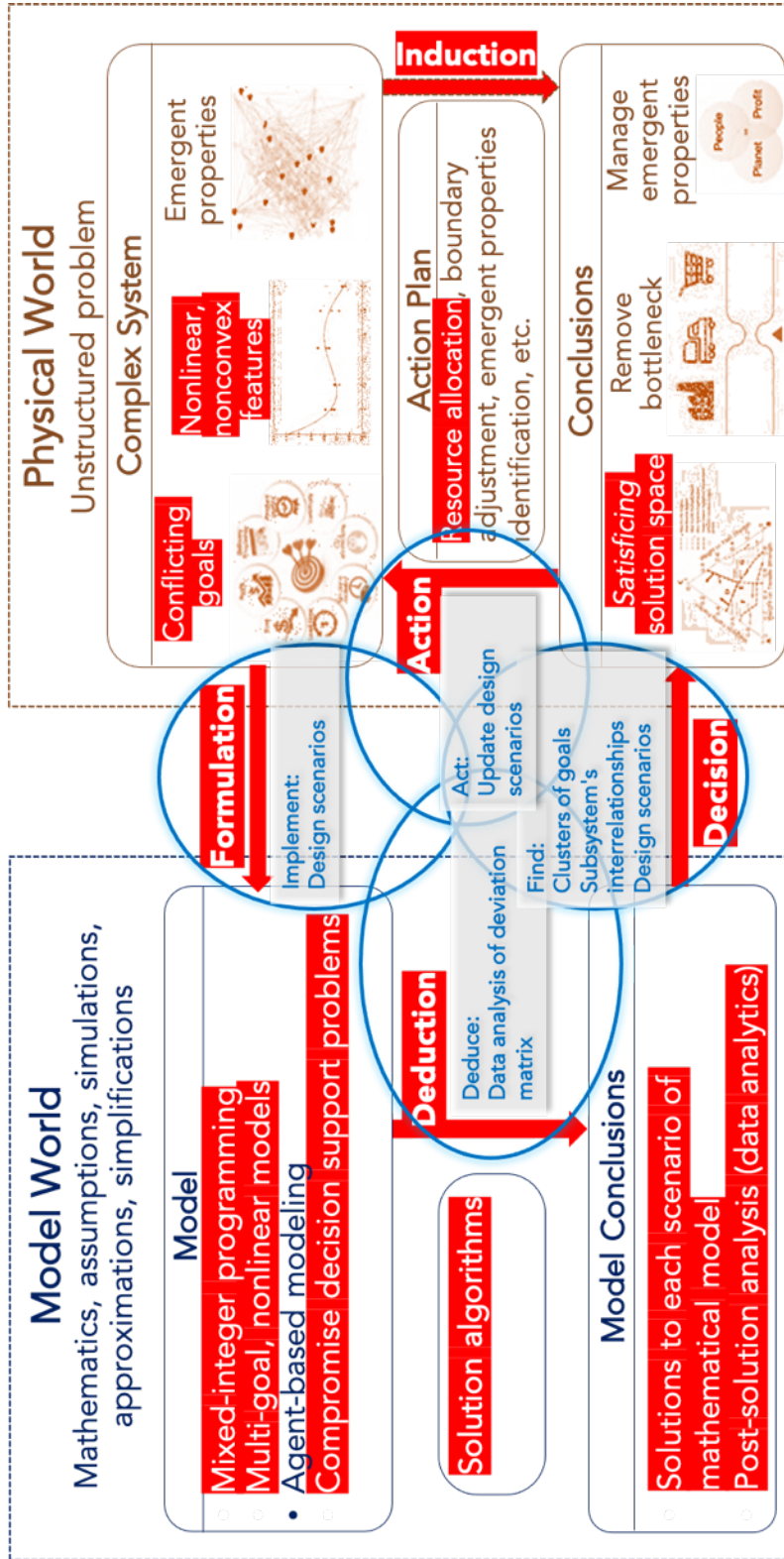


Figure 6. 17 The Procedures Involved in Formulation-Exploration-Evaluation – Establish the information exchange, knowledge awareness, and instructions sharing among formulation deduction, decision, and action



### ***6.5.2 Summarizing How We Realize Type I, III, & IV Robust Design***

For the test problem, designing the Rankine cycle thermal system, Type I uncertainty is identified as the various results due to the starting point changing, thus we call “XPLORE” module in DSIDES (details are introduced in Section 2.2.3) to identify a local area that is feasible and with relatively good goal achievement to start searching.

Type III uncertainty, defined as the uncertainty in model approximation (ways of combining multiple goals), is managed using the three algorithms in adaptive leveling-weighting-clustering (ALWC) algorithm, Algorithm 1 Leveling, Algorithm 2 Weighting, and Algorithm 3 Clustering; see Figure 6.13. By exploring different leveling and weighting formats to combine the multiple goals that represent the interest of subsystems, the goals can be clustered and the subsystems can be identified and their interrelationship can be learned, therefore, the combination of the goals can be updated and the most appropriate and representative design scenarios (which in this test problem is the combination of goals) can be selected. Using the selected design scenarios instead of enumerating all of them can speed up the learning and reduce the computational complexity.

Type IV uncertainty, the uncertainty in using domain knowledge to simplify the model, for the Test Problem 3, has two manifestations, fixing decision variables based on domain expertise and selecting design scenarios using experience. The first uncertainty is managed using the “fixing variables” and “XPLORE” modules. In DSIDES, when applying both modules to a variable, we first using interior-point searching to identify the best value for a variable and then fix it, which means treating it as a deterministic parameter. If we fix more than one variable, then the best combination of the variables is returned as a vector and those variables’ values are fixed by implementing the vector. This significantly reduces the computational complexity especially when there are discrete variables. The second uncertainty is managed through the leveling and clustering

algorithm in the ALWC algorithm. By clustering the goals and leveling them based on their clusters, we select the design scenarios that allow boosting the system performance.

By implementing the ALWC algorithm as an extension of DSIDES, we can identify the solution space that is relatively insensitive to the Type I, III and IV uncertainty that we need to manage in a specific problem. In this way, we realize Type I, III & IV robust design. see the summary in Table 6.9 as the closing remarks of Table 3.2 regarding the robust design realization and uncertainty management for Test Problem 3.

**Table 6.9 Summary of Test Problems 3 regarding Type I, II, & IV Uncertainty Management**

RD Type	RDI-II			RDIII		
				RDIV		
Method	M1: Formulation-Exploration Framework		M2: Adaptive Linear Programming with Parameter Learning (ALPPL)	<b>M3: Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)</b>		M4: Scenario Planning in Agent-Based Modeling
Chapter	Ch 4		Ch 5	Ch 6		Ch 7
Uncertainty Test Problem	T1: Dam network	T2: Supply chain	T3: Hot rolling process chain	<b>T4: Thermal system</b>		T5: Promoting second-season farming
Type I	<i>Uncertainty in timing and amount of inflow – Table 4.7</i>	<i>Uncertainty in demand side – Figure 4.25</i>	<i>Uncertainty in hyper parameter setting – Table 5.7, Figure 5.16</i>	<b><i>Uncertainty in parameter setting in solution algorithm (Starting point of searching) – call XPLORE in DSIDES</i></b>		Uncertainty in price (Price of agriculture products)
Type II	<i>Uncertainty in outflow (water release target) – Table 4.5</i>	<i>Uncertainty in supply side - Table 4.15</i>	<i>Uncertainty in user preferences – Table 5.6</i>			<i>Promotion effort and timing</i>
Type III			<i>Uncertainty in model approximation due to heuristics in approximation – Table 5.5</i>	<b><i>Uncertainty in model approximation (ways of combining multiple goals) – leveling-weighting-clustering algorithms (Figure 6.13)</i></b>		

Type IV				<i>Uncertainty in using domain knowledge to simplify the model (fixing decision variables and selecting design scenarios) – “fixing variable” and “XPLORE” modules in DSIDES, and leveling and clustering algorithm in the ALWC</i>	<i>Interventions that change the mathematical relation among promotion and result (developing local market)</i>
RD – robust design M – method EVe – empirical verification of the method T – test problem					

### 6.5.3 Role of Chapter 6

In Chapter 6, given the frame of references on the modeling constructs and solution algorithms for multi-goal or multi-objective problems, especially focusing on the exploration of the interrelationship among subsystems and the combination of goals, which is an extension of the frame of references in Chapter 2. A method, the Adaptive Leveling-Weighting-Clustering (ALWC) algorithm, is proposed to aware and explore the subsystems of the system and the design scenarios. A test problem, designing a Rankine cycle thermal system, is used to verify the proposed methods. It is proved that using the ALWC algorithm, the interrelationship among the subsystems can be learned, the most representative design scenarios that can reasonably organize the subsystems to boost the system’s efficiency can be identified and applied. Research Question 3 is addressed.

# CHAPTER 7 TYPE I, II, & IV ROBUST DESIGN THROUGH EMERGENT PROPERTIES IDENTIFICATION AND INTERPRETATIONS

## – EXPLORING CRITICAL FACTORS THROUGH SCENARIO PLANNING IN AGENT-BASED MODELING

### *The new knowledge in Chapter 7:*

*A framework to identify and leverage the critical factors to reach simulation goals*

In Chapter 7, see Figure 7.1: in Section 7.1, the reference on designing promotions using agent-based modeling is framed and identified the limitations; in Section 7.2, the test problem, the motivation and challenges in learning emergent properties in promoting the second-season cultivation in an island village in India are introduced; in Section 7.3, based on the research gaps described in Section 1.5 and the Method 4 proposed in Section 3.3.4, we introduce the framework of scenario planning in agent-based modeling for exploring critical factors in details; in Section 7.4, the proposed method is applied to the second-season cultivation promotion simulation problem for critical factor identification and emergent property learning; in Section 7.5, summarized the role of Chapter 7.

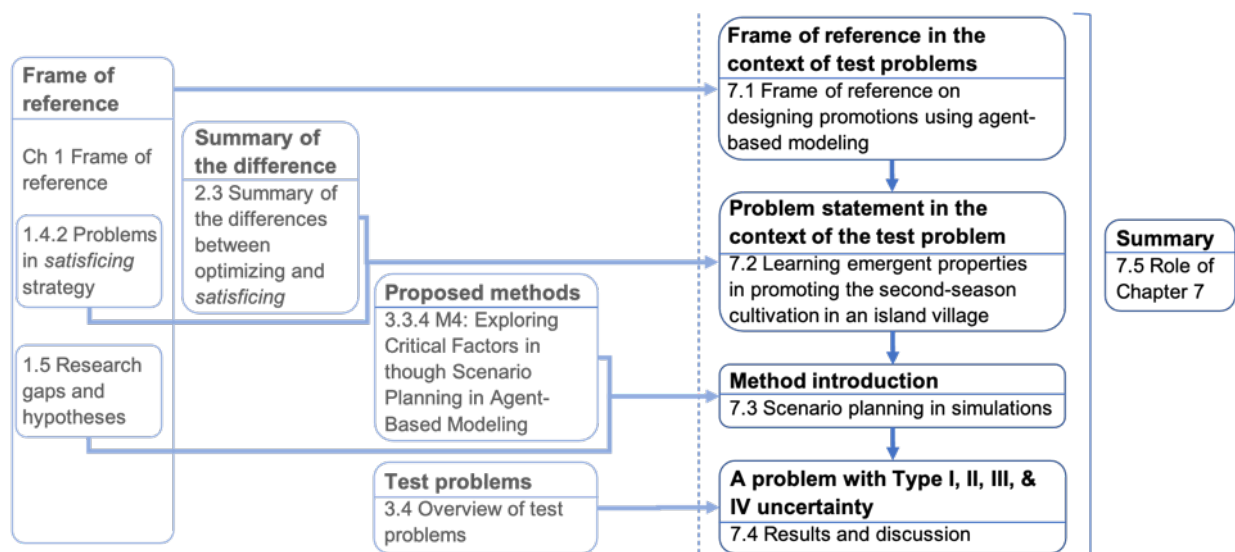


Figure 7. 1 Organization of Chapter 7

The plan of specifying and answering Research Question 4 in the context of the test problems is shown in Table 7.1. In Chapter 7, the Proposed Method 4 (M4), the scenario planning framework in agent-based modeling, is empirically verified (EVe4) using a test problem, designing the promotion of the second-season cultivation in a rural community (T4). Research Question 4 (RQ4) is specified in the context of the test problem (SQT4) and answered (AQ4) by testifying M4. The empirical validation and theoretical validation are in Chapters 8 and 9.

**Table 7. 1 Plan of Specifying Research Question 4 (RQ4) and Empirically Verifying the Scenario Planning Framework in Agent-Based Modeling (M4)**

Chapte	Ch1	Ch 2	Ch 3	Ch 4-7		Ch 8	Ch 9
				Ch 4-6	Ch 7		
Actions	RG H	RD RQ SH	TVe M	EVe 1-3 SQT 1-3 AQ 1-3	<i>EVe4: use a test problem on designing the promotion in a social network to verify SH4 and demonstrate M4.</i>	CQ EVa	TE
					<i>SQT4: What is the method that facilitates recognizing emergent properties in a social network and identifying critical factors to reach promotion goals under the emergent properties?</i>		
					<i>SQT4.1: What are the critical factors that affect the collective behaviors under interventions?</i>		
					<i>SQT4.2: How can we identify the critical factors and select the appropriate scenario to reach an expected result?</i>		
Nomenclature	RG – give research gaps						
	H – give hypotheses						
	RD – tie to roust design						
	RQ – pose research questions						
	SH – specify hypotheses						
	TVe – theoretically verify hypotheses						
	M – introduce methods						
	EVe – empirically verify hypotheses						
	SQT – specify research questions in the context of test problems						
	AQ – answer research questions						
	CQ – closure the answers to research questions						
EVa – empirically validate hypotheses							
TE – theoretically extend the research							

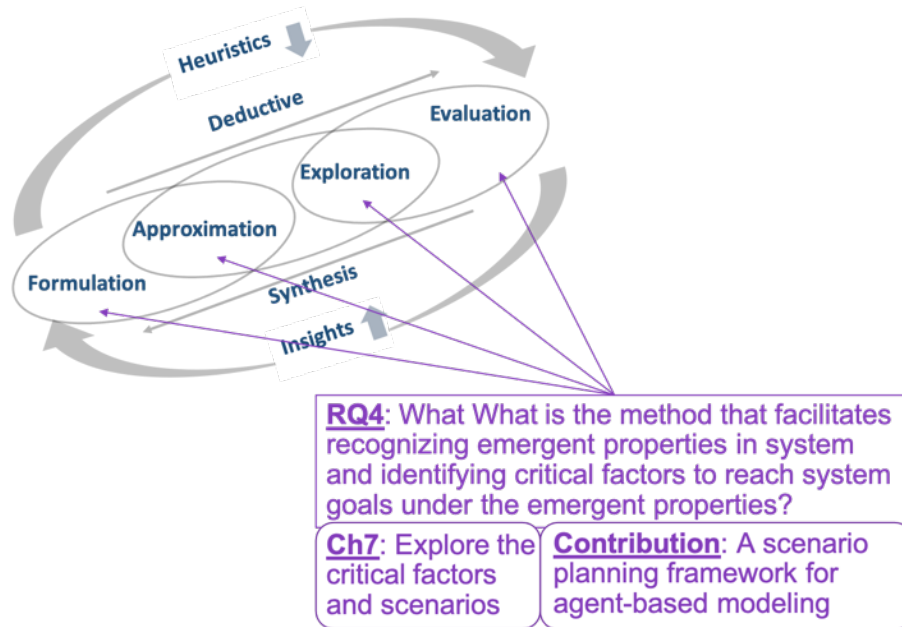
In this chapter, the Research Question 4 (RQ4) “*What is the method that allows passing the information through multiple scales of a system?*” is specified regarding the test problem, designing the promotion of the second-season cultivation in a rural community (T4), as follows (SQT4) and then further specified into two sub-questions indicating the tasks and answered.

***SQT4: What is the method that facilitates recognizing emergent properties in system and identifying critical factors to reach system goals under the emergent properties?***

***SQT4.1: What are the critical factors that affect the collective behaviors under interventions?***

***SQT4.2: How can we identify the critical factors and select the appropriate scenario to reach an expected result?***

It is hypothesized that interactions among the formulation, exploration, approximation, and evaluation of a design problem should be studied and the mechanisms of information sharing and intervention between the four procedures are established. See Figure 7.2.



**Figure 7. 2 Specified Research Question 3 and the Relevant Stages to be Connected in Design Evolution Cycle**

### 7.1 Frame of Reference on Designing Promotions using Agent-Based Modeling

It is challenging to change people’s mode of making a living and lifestyle when a new technology is available, especially in relatively closed and underdeveloped areas (Wilhelm 2000, Stewart 2016). Modeling methods and simulations are used to predict people’s acceptance and adoption of a new lifestyle or technology, but often the verification of the simulations and the validation of the utility of the method are missing (Parker, Manson et al. 2003, Chitungo and Munongo 2013). Because of unavoidable errors and flaws in modeling (Box 1976) and due to the complexity of the model environment, methods such as sensitivity analysis (Abreu and Ralha 2018, Guo, Chen et al. 2019) and Monte Carlo analysis (Lumbroso and Davison 2018) are often used to manage uncertainties and provide decision support in a changing environment. However, for the design of sociotechnical systems, where the initial data is lacking or difficult to verify and quantify (Afshari and Peng 2015), and when the future is uncertain, we need a method to explore the variability of

the model results and identify sensitive factors that contribute to variabilities (Abreu and Ralha 2018).

From the literature on simulating social behaviors and providing decision support for policy-making, agent-based modeling (ABM) is a practical tool (Qiu, Xu et al. 2018). There are many examples of the use of ABM to observe patterns in collective behavior, identify critical factors, and intervene in the system by changing critical factors, including the exploration of the extent of influence, or radius of influence, among neighbors with respect to promotion actions (Opiyo 2019), the identification of critical factors that contribute to population dynamics (Qiu, Xu et al. 2018), the estimation of the effects of policy interventions on the investment in new equipment (Al Irsyad, Halog et al. 2019), and decision support for managing the potential labor reproduction (Rossoshanskaya 2019). In Table 7.2, we list some representative publications on capturing social behavior and leveraging critical factors to serve a promotion goal. However, there is less literature on decision support based on planning for various scenarios. Therefore, here, we provide scenario-planning-based decision support to social entrepreneurs, SEs, with respect to reaching social-economic goals in various situations. We use a test problem of promoting second-season cultivation, that is, growing a second crop each year in an underdeveloped, rural Indian village.

**Table 7. 2 Some Representative Applications of Agent-Based Modeling (ABM) for New Technology Acceptance and Policy Impact**

<b>Author and Year</b>	<b>Problem Description</b>	<b>Method</b>	<b>Results</b>	<b>Contribution</b>
<b>Opiyo, 2019 (Opiyo 2019)</b>	Study neighborhood influence and social pressures on temporal diffusion of solar home systems	Agent-based modeling with survey data	Visibility of newly installed SHS and increasing influence radius leads to growth in SHS installations	The survey method is helpful to acquire relatively quantifiable data for a social problem



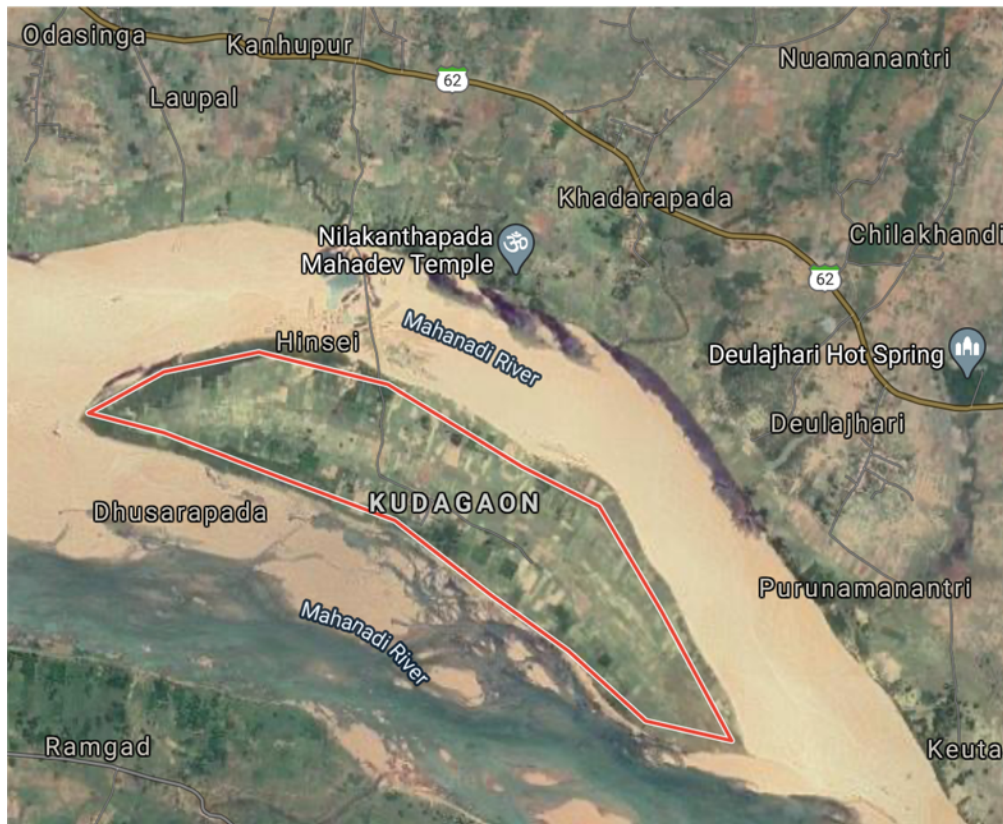
<b>Qiu, 2018 (Qiu, Xu et al. 2018)</b>	Simulate urban land development and population dynamics	An agent-based and spatial genetic algorithm framework (PDULD)	Government policies dominate the process of land development	Community cohesion theory is introduced into the model; historic data are used to verify the results
<b>Irsyad, 2019 (Al Irsyad, Halog et al. 2019)</b>	Estimate the effects of four solar energy policy interventions on photovoltaic (PV) investments, government expenditure, economic output, etc.	Uses hybrid energy - agent-based modeling	Results call for PV donor gift policy, the improvement of production efficiency, after-sales services and rural financing institutions	Integrate input-output analysis, environmental factors and socioeconomic characteristics of households in Indonesia
<b>Rossoshanskaya, 2019 (Rossoshanskaya 2019)</b>	Simulate labor potential reproduction; among the scenario forecasts, provide decision support on management actions	Agent-based modeling with multi-agents and multi-scenarios	The integrated agent-based model of labor potential reproduction at the municipal level	The model is filled with sociological and statistical data and has a user-friendly interface

## 7.2 Problem Statement – Promoting the Second-Season Cultivation in an Island Village in India

### *- Test Problem 4: apply scenario planning in agent-based modeling to a social system design problem*

Our objective is to give decision support to social entrepreneurs (SEs) promoting second-season cultivation in Kudagaon, a relatively isolated and underdeveloped village in Odisha, India. Kudagaon is a village on an island surrounded by a river (Figure 7.3). There are 85 households in the village and each household has some farmland. There is a rainy season (or monsoon season) and a dry season. Most households do one-season cultivation, growing rice or vegetables in the rainy season using the water from the river. In the dry season, the water level of the river drops significantly, so the majority of the families cannot farm because of the scarcity of water. Therefore,

two-season cultivation is not possible; hence, the villagers cannot increase their family savings by two-season cultivation. It has been determined that Kudagaon has sufficient underground water for farming in the dry season, but villagers need to purchase or rent equipment to pump the underground water and transport it to their farmland. Since most families do not have adequate savings, they cannot afford the equipment. This is a dilemma.



**Figure 7.3 The satellite map of Kudagaon**

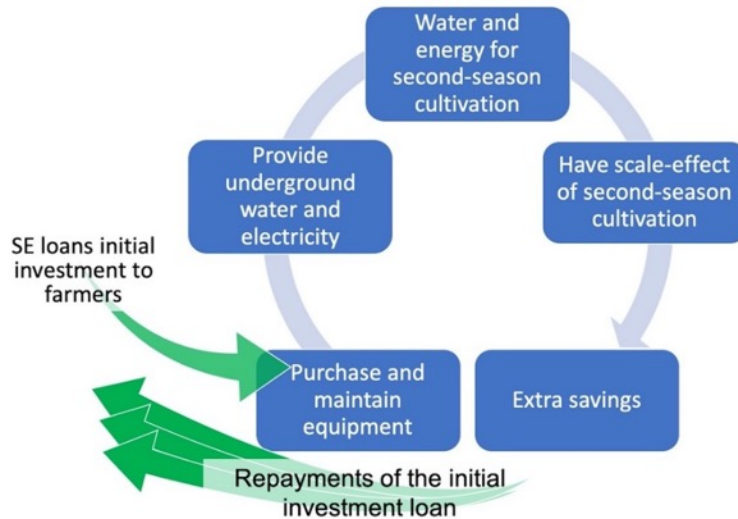
When a household (family) grows crops only once a year, during the dry season, the main laborer(s) of the family migrate so that they can obtain daily wages. These daily wages are their families' only source of income in the dry season. We call this "migration income." Migration income is usually less than the income which could be garnered from second-season cultivation, assuming sufficient water. In addition, the wellbeing of families with migration worker(s) (these are "migration families") regarding family stability and social status is worse than that of the families

who do two-season cultivation (these are “cultivation families”). However, during the rainy season, families who do one-season cultivation when their farmlands are away from the river may still consider growing crops in the dry season based on the specific climate and the market situation each year. They can lease equipment and water pumps to get underground water for second-season cultivation, but the cost is high, and the risk is high. However, if they anticipate that they may gain more profit through second-season cultivation than doing migration work, they may stay and grow second-season crops instead of migrating. Currently, the probability of a household which does one-season cultivation to switch to two-season cultivation is as low as 5%. This number has been provided by our colleagues, the social entrepreneurs, SEs, at SunMoksha, based on historic data and from interviews of the villagers.

The role of a social entrepreneur (SE) in this project is to help villagers in Kudagaon improve their social and economic status by promoting second-season cultivation. The SE plans to provide farmers the initial investment as loans to construct public infrastructure so that electrical power can be generated, and underground water may be obtained and transporting the water and electricity to make farming possible in the dry season. The underground water and electricity are provided to farmers as utilities using affordable, tiered pricing. After the farmers profit from the second-season cultivation and have savings, they repay the loans within several years. In Figure 7.4, we illustrate how the initial investment from the SE helps to make second-season cultivation feasible.

The acceptance of second-season cultivation is crucial for the success of this project. Hence, in addition to loans and technical support, it is important to promote second-season cultivation among farmers. The benefits from second-season cultivation include improving a family’s economic and social status with a higher and more stable income and keeping the family together. However, not

all villagers realize these benefits, and there is a considerable reluctance to change one's mode of production, lifestyle, and source of income.



**Figure 7. 4 The SE's plan for facilitating second-season cultivation**

Based on the dilemma and the difficulties that the SE encounters, we summarize the SE's possible actions and targets in Table 7.3. Each target either improves the villagers' social status, or improves their economic status, or both. We hypothesize that by increasing the two-season cultivation households, there can be a scale effect due to stable market demand and lowering the unit cost of storage and transportation; thus, villagers' income can be increased, and their economic status can be enhanced. An SE's job is to boost villagers' economic and social status in both the short term and long term. In the short term, the SE wants to improve the two-season cultivation rate from  $\beta_1$  to  $\beta_2$ , and, in the long term, the SE wants to improve  $\beta_1$  to  $\beta_3$ .  $\beta_1$  is a given static rate based on historical data.  $\beta_2$  and  $\beta_3$  are targets that the SE wants to reach. There are different scenarios of  $\beta_2$  and  $\beta_3$ . In this chapter, we offer suggestions on promotion efforts and their durations to reach each scenario of  $\beta_2$  and  $\beta_3$ .

**Table 7. 3 The SE's Target based on the Current Situation**

Current Problem	Reason or Dilemma	SE's Action	Target	Category
Only $\beta_1$ of the households anticipate a better profit of second-season cultivation so they grow twice a year $\beta_1 \approx 5\%$	Farmers cannot afford the equipment to acquire underground water for farming in the dry season	Having promotion activities <sup>34</sup> ; providing underground water;	Raising the second-season cultivation rate from $\beta_1$ to $\beta_2$ in the promotion year	<b>Short-term economic status</b> – increase profit level
			Maintaining the second-season cultivation rate of $\beta_3$ after two years following the promotion year	<b>Long-term economic status</b> – stability and reliability of income sources
Reducing the migration rate from $(1 - \beta_1)$ to $(1 - \beta_2)$ in the promotion year			<b>Short-term social status</b> – the confidence to make an improvement	
Maintaining the migration rate at $(1 - \beta_3)$ two years after the promotion year.			<b>Long-term social status</b> – the family stability and sense of security and self-sufficiency	
$(1 - \beta_1)$ of the households do not anticipate a better profit of the second-season cultivation so they grow only once a year and migrate in the dry season				

We introduce modeling and scenario development in Section 7.3, discuss the results of the scenario planning in Section 7.4, and summarize the role of Chapter 7 in this dissertation in Section 7.5.

## 7.3 Modeling and Scenario Development

### 7.3.1 Build the Architecture and Set the Baseline Scenario of the Agent Based Model

To capture the factors that impact the promotion effects, we simulate the households' behavior using agent-based modeling (ABM). The simulation is performed using AnyLogic 8 PLE software. Because Kudagaon is a single community with a relatively flat social hierarchy, we define each household as an agent and use a single type of agent. Social influence and influence of each

---

<sup>34</sup> The promotion activities include holding community discussions, visiting each household in person, giving training on second-season cultivation and strategies to conserve water, etc. As Kudagaon is a small village with only 85 households, SEs are able to visit and interview each household.

family's neighbors are randomized in the same numerical range, although they vary from household to household.

The project's duration is three years. We simulate the households' behavior for four years. To establish a baseline, there is no intervention in the first year to simulate the villagers' behavior before the project launches. This pre-project simulation allows us to verify the model using current actual data. The SEs' promotion starts in the second year which takes place during the rainy season. The third and fourth years are post-promotion. This allows us to track the long-term effects of the SE's promotion.

**States and transitions between states.** An agent (household) has 11 possible states, as follows.

<i>S<sub>1</sub>for1S</i>	Growing once a year before SEs' promotion
<i>MigrationWork1</i>	Migrating during the dry season, before the promotion
<i>S<sub>1</sub>ToBePromoted</i>	Growing once a year in the SEs' promotion year
<i>BeingPromoted</i>	Being promoted to grow twice a year by SEs
<i>MigrationWork2</i>	Migrating during the dry season after being promoted
<i>S<sub>1</sub>for1SAfter</i>	Growing once a year after the SEs' promotion year
<i>S<sub>1</sub>for2Ss</i>	Growing the first season crops for a two-season cultivation year
<i>Season2</i>	Doing a second season of growing in the two-season cultivation year
<i>Harvest</i>	Harvesting the second-season crops
<i>Profit</i>	Gaining at least the expected profit or even more from second-season cultivation
<i>NoProfit</i>	Gaining less profit than expected from second-season cultivation

The transitions between states are described in Table 7.4. The flowchart of the agents' state transition is in Figure 7.5. The trigger condition and/or the duration of each transition is illustrated in Figure 7.6.

**Table 7. 4 Transitions between Different States for an Agent**

Transition	Meaning
$S_{1for1S} \xrightarrow{\beta_1} Season2$	Before SEs' promotion, with probability $\beta_1$ , a one-season cultivation household will change to two-season cultivation. $\beta_1 = 5\%$
$S_{1for1S} \xrightarrow{(1-\beta_1)} MigrationWork1$	Before SEs' promotion, with probability $(1 - \beta_1)$ , a one-season cultivation household will do migration work during the dry season
$BeingPromoted \xrightarrow{\beta_2} Season2$	During the SEs' promotion year, with probability $\beta_2$ , a household being promoted will change to two-season cultivation. $\beta_2 \gg \beta_1$
$BeingPromoted \xrightarrow{(1-\beta_2)} MigrationWork2$	During the SEs' promotion year, with probability $(1 - \beta_2)$ , a household being promoted will migrate during the dry season
$Harvest \xrightarrow{\alpha} Profit$	With probability $\alpha$ , a household will reach the anticipated profit through second-season cultivation and decide to do two-season cultivation next year
$Harvest \xrightarrow{(1-\alpha)} NoProfit$	With probability $(1 - \alpha)$ , a household will not reach their anticipated profit through second-season cultivation and decide to do one-season cultivation next year
$S_{1for1SAfter} \xrightarrow{\beta_3} Season2$	After the SEs' promotion year, with probability $\beta_3$ , a one-season cultivation household will change to do two-season cultivation. $\beta_3 = \beta_2 \cdot \alpha$ (1)
$S_{1for1SAfter} \xrightarrow{(1-\beta_3)} MigrationWork2$	After the SEs' promotion year, with probability $(1 - \beta_3)$ , one-season cultivation household will migrate during the dry season

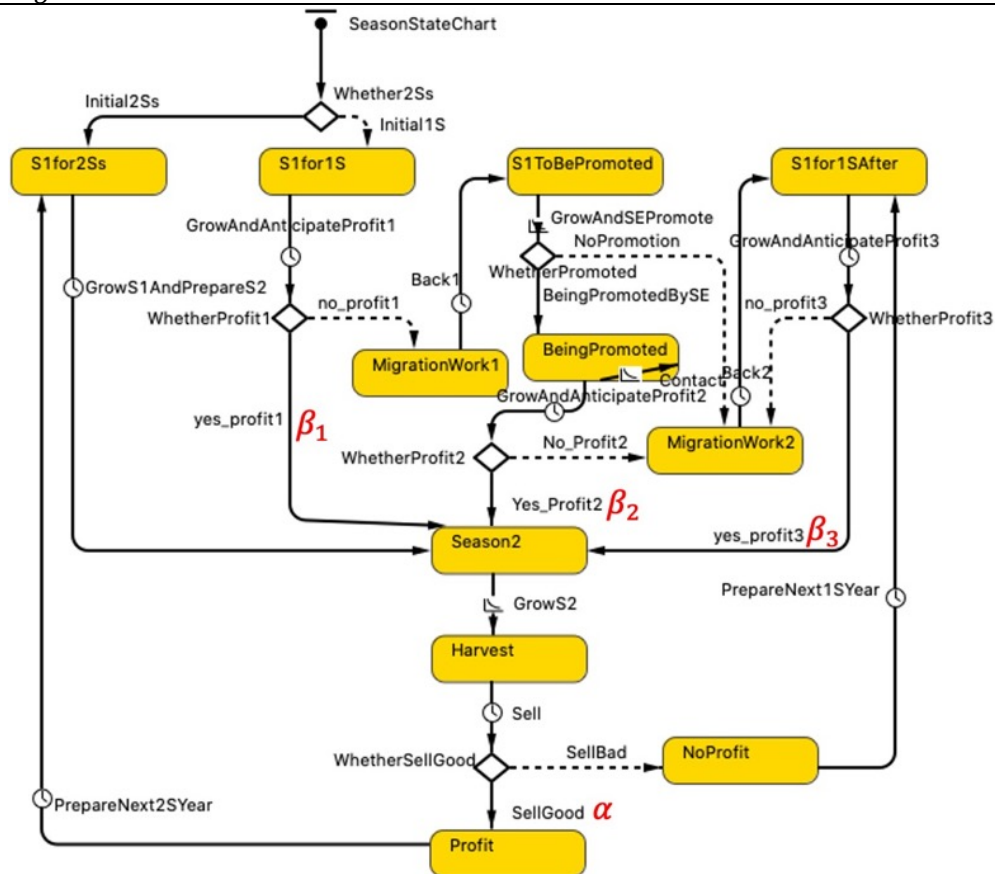


Figure 7.5 The Flowchart of the Agents' State Transitions

For Agents grow twice/year (every year)	For Agents grow once/year without SEs' promotion (1 <sup>st</sup> and 3 <sup>rd</sup> + year)	For Agents grow once/year with SEs' promotion (2 <sup>nd</sup> year)
Season 1 } Grow: 5 months	Season 1 } Grow & anticipate profit: 5 month	Season 1 } Grow & be promoted: 1 month
Season 2 } Grow: 1-4 months	If anticipate profit go to "grow twice/year, Season 2" Prepare 2nd season: 0 month Else Migration work Prepare migration: 0 month	Being promoted } Grow & anticipate profit: 4 month
Harvest } Sell: 1 month		If anticipate profit go to "grow twice/year, Season 2" Prepare 2nd season: 0 month Else Migration work Prepare migration: 0 month
Profit (sell good) or no profit (sell bad)	If Migration work Next year still once/year  Migrate and back: 7 months	If Migration work Next year still once/year  Migrate and back: 7 months
If profit } Prepare next year: 2 months next year still grow twice Else } Prepare next year: 2 months next year switch to grow once		

**Figure 7. 6 The Trigger Conditions and/or the Duration of the Transitions between the States of the Agents**

### 7.3.2 Scenario Development

A factor is any source of uncertainty source, including the model structure, initial conditions, and input parameters. In this project, based on the basic knowledge from the SE, we define the network structure, SE's promotion effort, promotion duration, villagers' anticipation of profit, and actual profit as factors to be explored. Different scenarios are developed for each factor by brainstorming between an employee of SunMoksha, Ayushi Sharma, and research assistants in Systems Realization Laboratory, Lin Guo and Vishnu Kamala, based on their experience, domain knowledge, and assumptions. Through scenario planning, the sensitivity of the results to each factor is analyzed, and critical factors are identified. In Table 7.5, we list the scenarios related to each factor. The expected outcome of scenario planning and the way to obtain these outcomes are given in Table 7.6.

**Table 7. 5 Scenarios for Testing Each Factor**



Factors	Scenarios
Network structure	Distance-based Scale-free
Promotion effort	Different percentages of households being promoted – 10%, 50%, 75%, 100%
Promotion duration	Different promotion durations – one month, two months, three months, four months
Anticipation	Different positive anticipations of second-season profit – 75%, 95%
Profit	Different percentages of households who actually obtain higher profit – 75%, 95%

**Table 7. 6 The Expected Outcome of the Scenario Planning**

Expected outcome	The way to obtain the outcome		
		Short-term effect	Long-term effect
Simulation results	Improved economic state	The number of households that gain expected profit by growing twice a year in the <i>promotion year</i>	The number of households that gain expected profit by growing twice a year in the <i>fourth year</i>
	Improvement of social status	The number of households that migrate during the dry season during the <i>promotion year</i>	The number of households that are directly or indirectly promoted
Critical factors	Identify whether the simulation results are sensitive to scenario changes for each factor		
Variability of the simulation results	By changing the scenarios of the critical factors, obtain the range of the output of the model		
Identify how the critical factors affect the simulation results	Capturing qualitative and quantitative relationship among critical factors		

## 7.4 Results and Discussions

We explore the network type, promotion effort and duration, and anticipation level of profit and actual profit level. The results indicate that the acceptance of second-season cultivation is insensitive to network type but is sensitive to promotion effort. The factors that particularly affect the short-term result are promotion duration and villagers' anticipation; the factor that affects the long-term result is the real profit that second-season cultivation produces. In Table 7.7, we summarize the results of the scenario planning. We describe our exploration process and observations in detail in Sections 7.4.1-7.4.3.

### 7.4.1 Exploring the Network Type and Promotion Effort and their Interaction Effects

The SE wants to reduce the migration population whereas and increase the profit population. In Figures 7.7 and 7.8, we present the results of two network types when the SE's promotion reaches every household and 50% households, respectively. In each graph, the horizontal axis represents time (unit: month), and the vertical axis represents the number of households. The smooth thick line represents the number of migration households during the dry season in the first year, the thin line with dots represents the households that gain expected profit from second-season cultivation, and the thin line with squares represents the number of migration households during the dry season in and after the promotion year (the second, third and fourth year).

**Table 7. 7 The Summary of the Results of the Scenario Planning**

Factor	Controllable or not	Scenario	Meaning	Observation	Whether critical or not
Network type <sup>35</sup>	No	Distance-based	One community with strong neighborhood influences but weak or zero distant interactions	The scale-free network has slightly better results in the promotion year, but in the long-term, network type does not affect the promotion result; see Figures 5 and 6.	No
		Scale-free	One community with asymmetric influences between any two connected households and the influence does not depend on distance		
Promotion effort	Yes – SEs can control the promotion effort by reaching out to different numbers of households	Reaching different numbers of house-holds: 10%, 30%, 50%, 75%, 100%	The promotion condition and target can be different from village to village, so searching for an appropriate rate for each village is necessary to reduce the promotion cost	The promotion result is always improved by increasing the promotion effort. The short-term effects are more sensitive to the promotion effort than the long-term effect; see Figures 7.8 and 7.9, and Table 7.8.	Yes, more critical for short-term effects
Promotion duration	Yes – SEs can perform the promotion for different	The promotion duration may last from one	A short promotion means relatively short direct promotion but long indirect promotion. This is suitable	For short-term effects, long promotion has much better results; for long-term effect, short promotion and long	Only critical for short-term effects

<sup>35</sup> A SE needs to determine the network type of a village through observation and statistics. The specific indicators used to determine the network type may vary from village to village.

	lengths of time	month to four months during the rainy season	for a village with a strong mutual influence among households and vice versa.	promotion show similar results; see Figure 7.10 and Table 7.9	
Villagers' anticipation and real profit – $\beta_2$ of households anticipates a better profit through second-season cultivation, $\alpha$ of them gain real profit as anticipated	Initially, $\alpha$ and $\beta_2$ are uncontrollable, but as the project goes on, SEs can control them by improving productivity, developing more market demand, etc.	$\beta_2 = 95\%$ $\alpha = 95\%$	Optimistic profit anticipation; good economy, weather condition, or soil fertility	The short-term result is more sensitive to profit anticipation; The long-term result is more sensitive to actual profit, See Figure 7.11	Profit anticipation is critical for short-term effects; real profit is critical for long-term effect
		$\beta_2 = 95\%$ $\alpha = 75\%$	Optimistic profit anticipation; acceptable economy, weather condition, or soil fertility		
		$\beta_2 = 75\%$ $\alpha = 95\%$	Conservative anticipation; good economy, weather condition, or soil fertility		

We explore different influence radius of the distance-based network, 50 meters (Figure 7.7-a) and 100 meters (Figure 7.7-b). We explore different average degrees of the scale-free network, 5 (Figure 7.7-c) and 10 (Figure 7.7-d), that is the average number of connections of each household. We also explore two different promotion efforts, reaching all households and reaching 50% households (the households can promote each other through the network), with one-month duration.

**Observations from exploring the type of network**, comparing Figure 7.7 (a) and Figure 7.7 (b) with Figure 7.7 (c) and Figure 7.7 (d): neither the short-term nor the long-term results are sensitive to network types.

**Observations from exploring the influence radius**, comparing Figure 7.8 (a) with Figure 7.8 (b): if the village is a distance-based network, when the influence radius is larger, the short-term effect is slightly better, whereas the long-term effect is insensitive to the influence radius.

**Observations from exploring the average degree of the network**, comparing Figure 7.8 (c) with Figure 7.8 (d): when a village is a scale-free network, neither the short-term nor the long-term results are sensitive to the average degree, the average number of connections of each household.

**Observations from exploring interaction effect of network type and promotion effort:** only the short-term results are sensitive to promotion effort, whereas the long-term results are insensitive to the promotion effort because in long-term, the households promote each other sufficiently.

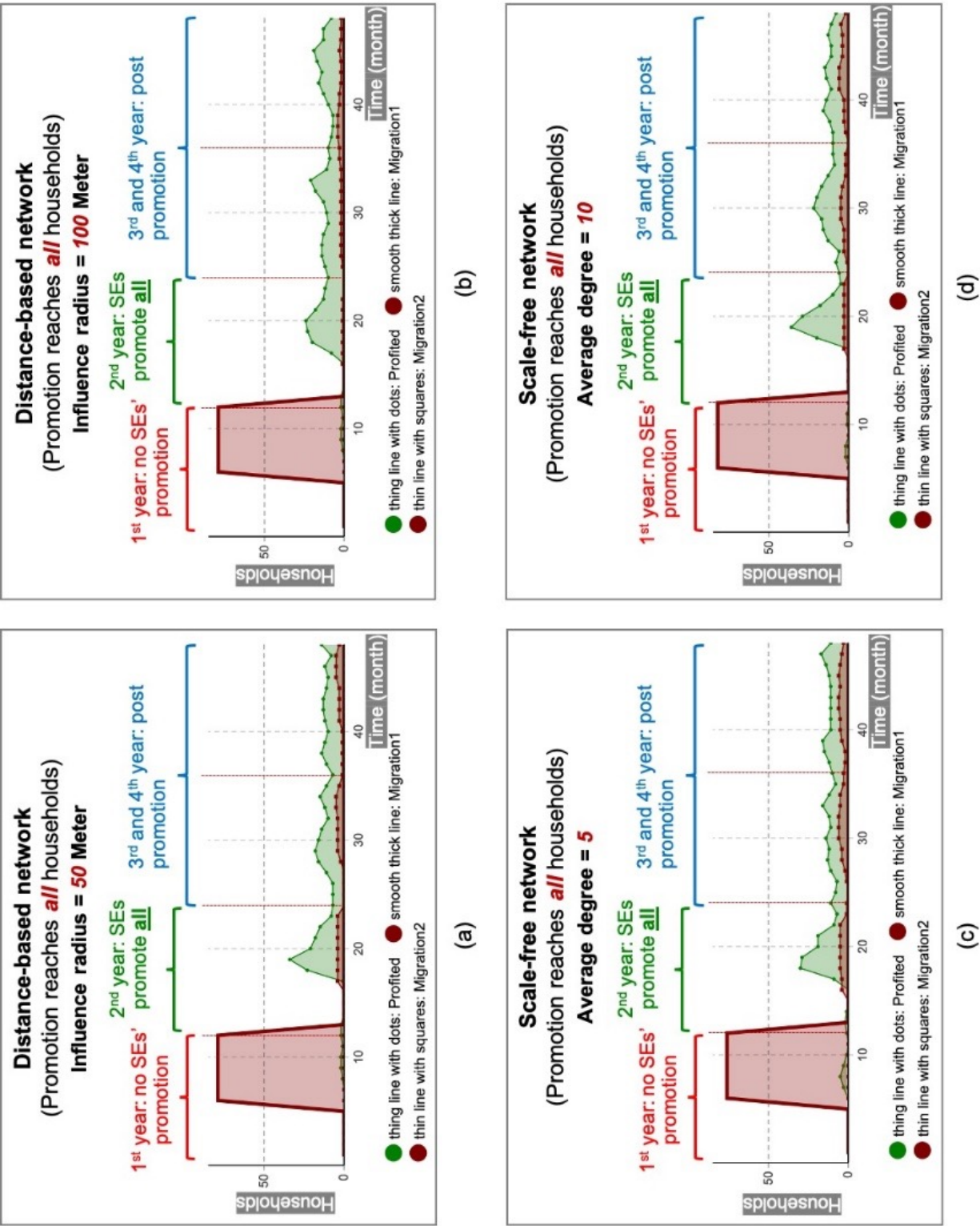
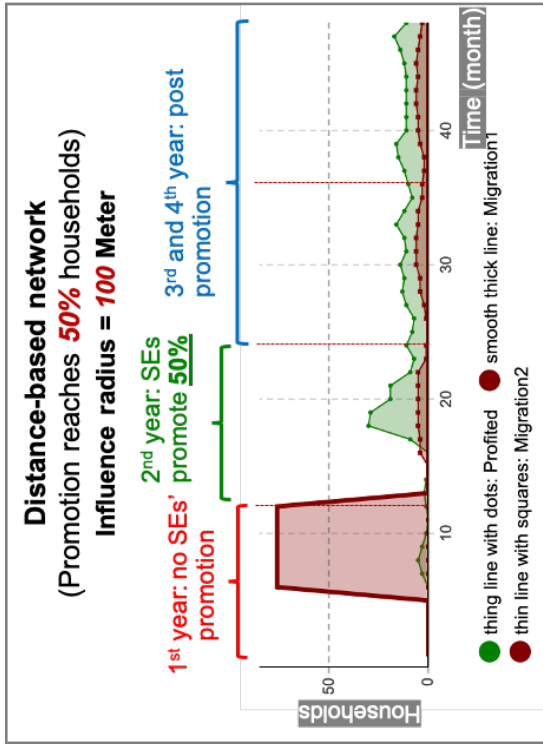
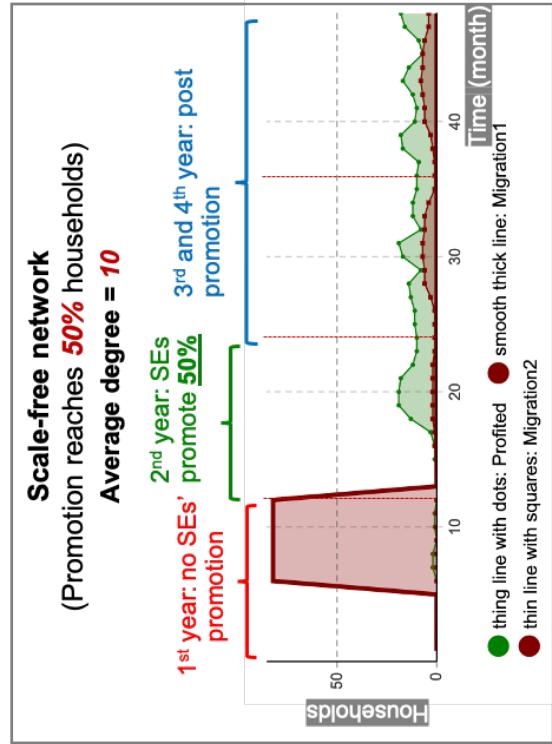


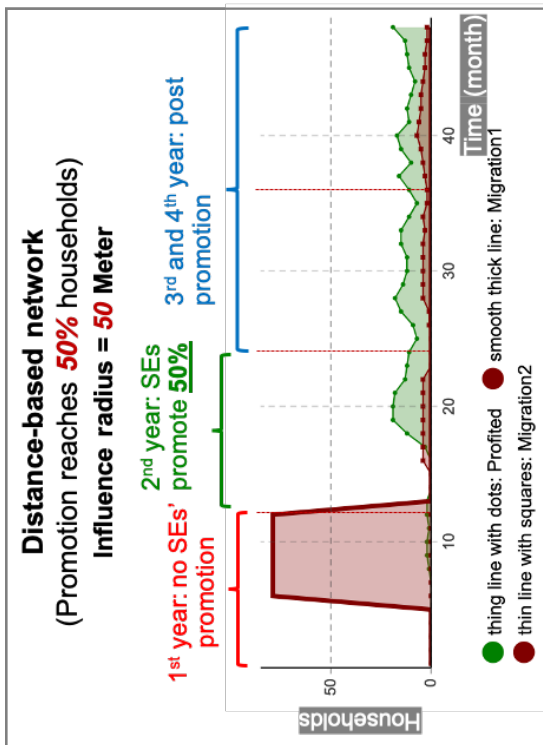
Figure 7. 7 Results for Two Network Types When Promotion Reaches All Households



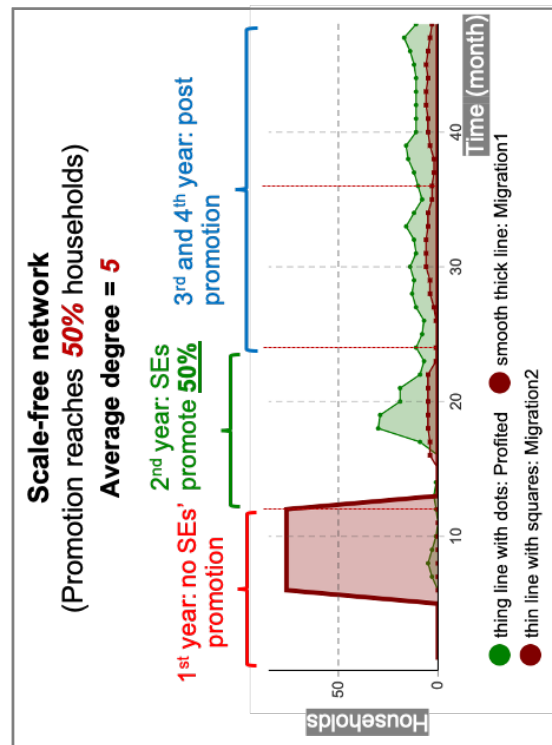
(b)



(d)

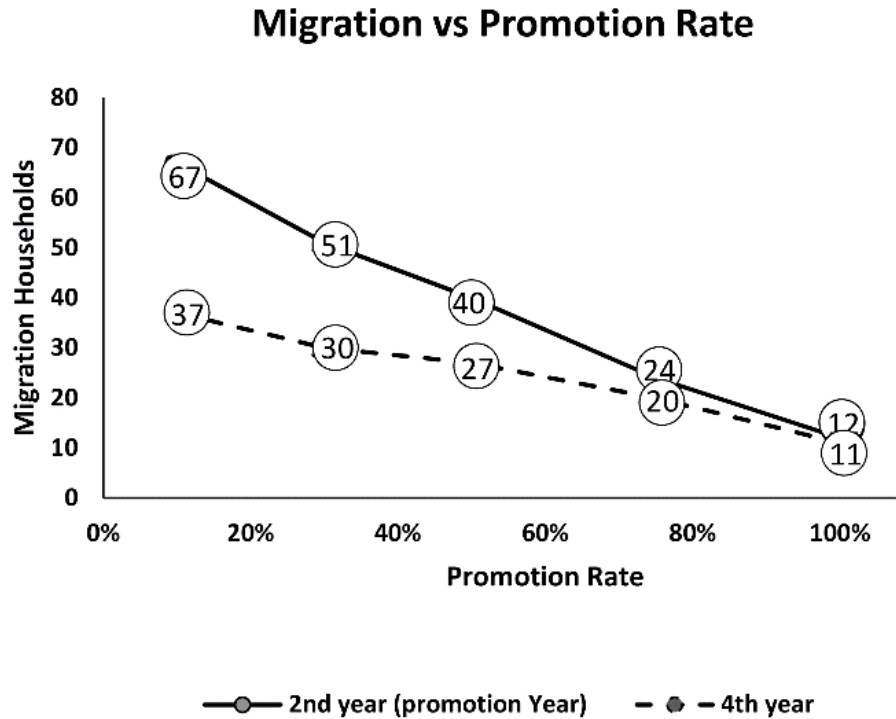


(a)



(c)

Figure 7. 8 Results for Two Network Types When Promotion Reaches 50% of Households



**Figure 7.9 The Migration Households with Different Promotion Effort**

In Figure 7.9, we illustrate the number of migration households during the promotion year (the short-term effect) and two years after the promotion year (the long-term effect). As the promotion rate increases, the marginal improvement in the short-term is greater than that in the long-term. In Table 7.8, we list the interaction effects of influence radius and promotion effort. Even if the SE promotes every household, there will still be 11-13 households migrating during the dry season.

**Table 7.8 Promotion Effort Exploration – Migration Household in the Promotion Year and in the End-of-Project Year with Different Network Scenarios**

Type and setting of the network	Promotion Effort									
	10%		30%		50%		75%		100%	
	2 <sup>nd</sup> year	4 <sup>th</sup> year	2 <sup>nd</sup> year	4 <sup>th</sup> year	2 <sup>nd</sup> year	4 <sup>th</sup> year	2 <sup>nd</sup> year	4 <sup>th</sup> year	2 <sup>nd</sup> year	4 <sup>th</sup> year
Distance-based network With influence radius: 50 m	67	37	51	30	40	27	24	20	12	11

Distance-based network With influence radius: 100 m	66	37	51	28	40	27	26	20	9	13
Scale-free network With the average number of connections for a household being 10	68	40	62	37	44	26	27	22	12	12

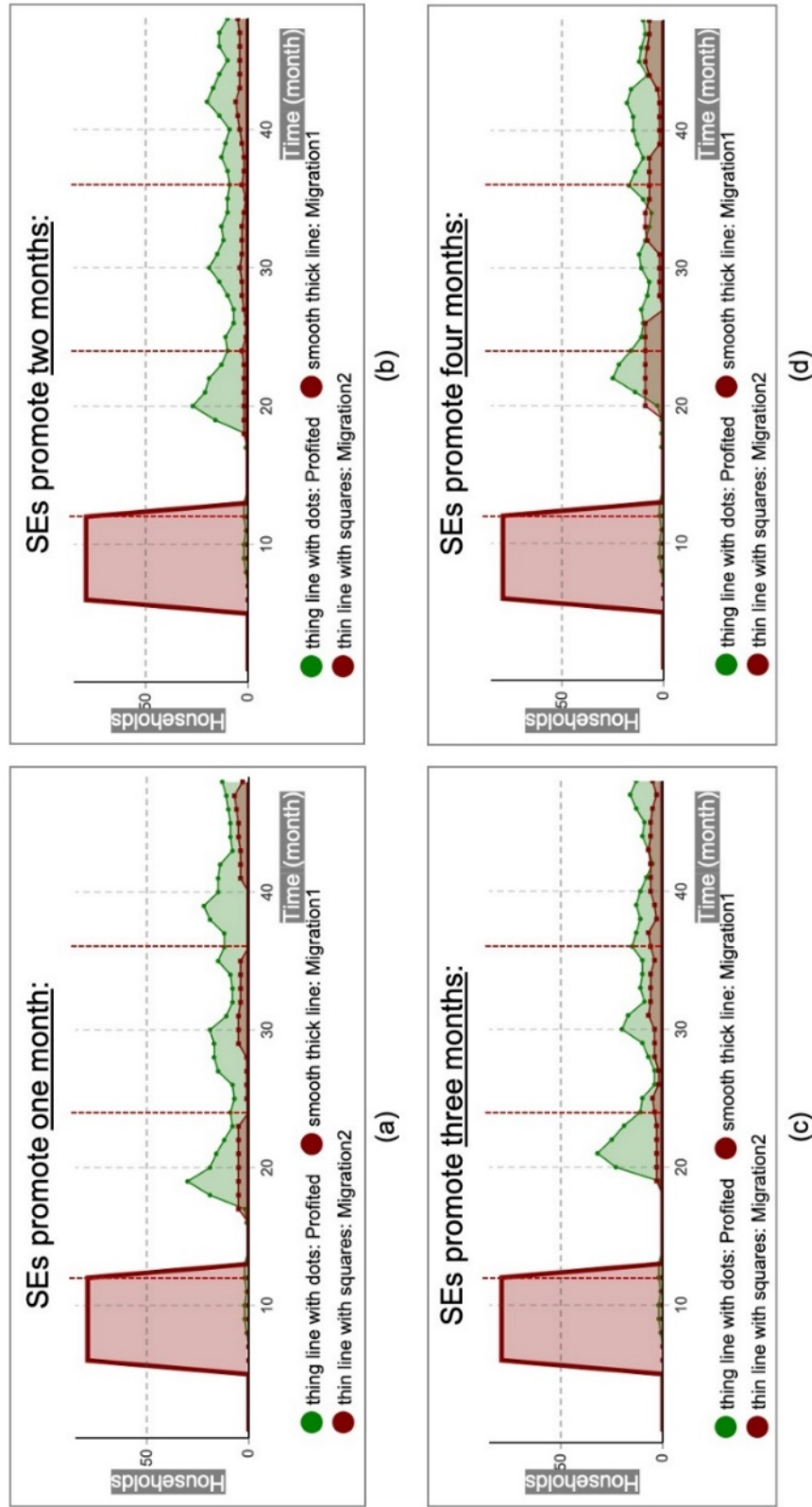
**Exploring promotion effort and its interaction with influence radius:** with the consideration of various promotion efforts, the simulation results are still insensitive to the types of network and are not sensitive to the network setting either. The migration population is reduced with the increase of promotion effort. To save cost, A SE can select the appropriate promotion effort (given a certain target for population migration) instead of promoting every household and the economic and the social status can be increased.

#### ***7.4.2 Exploring the Promotion Duration***

In Figure 7.10, we show the promotion results with different durations. One month – Figure 7.10 (a), two months – Figure 7.10 (b), three months – Figure 7.10 (c), and four months – Figure 7.10 (d). As we have shown that the results are insensitive to the type and setting of the network, we select a distance-based network and a 75-meter influence radius. The migration population under different scenarios is summarized in Table 8. The results reveal that prolonging promotion results in larger profit in both the promotion year and the following years. The reason is that a longer promotion results in a larger population being promoted both directly (by the SE) and indirectly (by villagers themselves). Direct promotion together with the indirect promotion reinforces the villagers’ acceptance of second-season cultivation. However, when prolonging the promotion from three months to four months, the results do not improve. This indicates that the villagers’ capacity for accepting an idea through promotion has an upper limit; therefore, overwhelming promotion



does not bring a higher social acceptance and a three-month promotion gives the best results for this village.



**Figure 7. 10 Simulation Results for Different Promotion Durations – Using a Distance-Based Network with a 75-Meter Influence Radius**

**Table 7. 9 Migration Population (Households) during the Four Years with Different Promotion Durations**

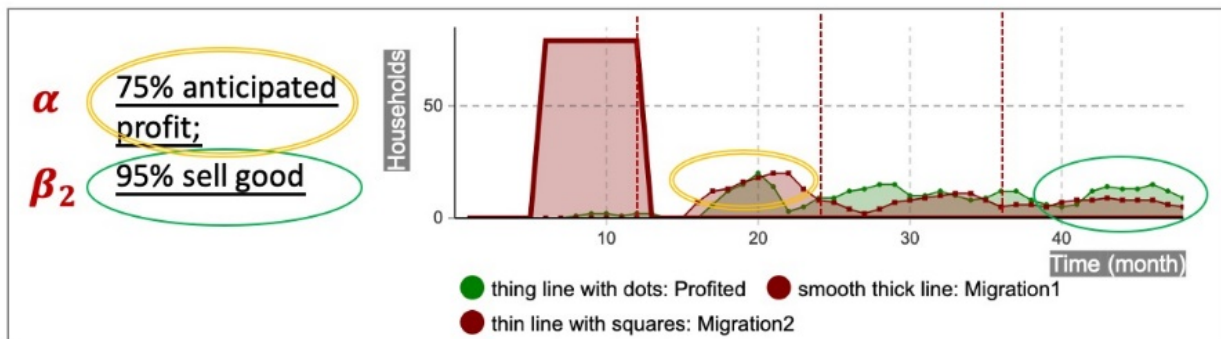
<b>Promotion Duration</b>	<b>1<sup>st</sup> year</b>	<b>2<sup>nd</sup> year (promotion year)</b>	<b>3<sup>rd</sup> year</b>	<b>4<sup>th</sup> year</b>
<b>1 month</b>	80	1-10	1-11	1-9
<b>2 months</b>	80	10	1-9	1-10
<b>3 months</b>	80	0-1	1-9	2-9
<b>4 months</b>	80	0-2	1-8	2-9

**Exploring promotion duration:** prolonging the SE’s promotion allows more interactions among the SE and the villagers and also triggers more indirect promotion among villagers. Therefore, it enhances the acceptance of second-season cultivation. Thus, the value of increasing the duration of promotion has an upper limit. In this project, this limit is three months – after three months, additional promotion does not help.

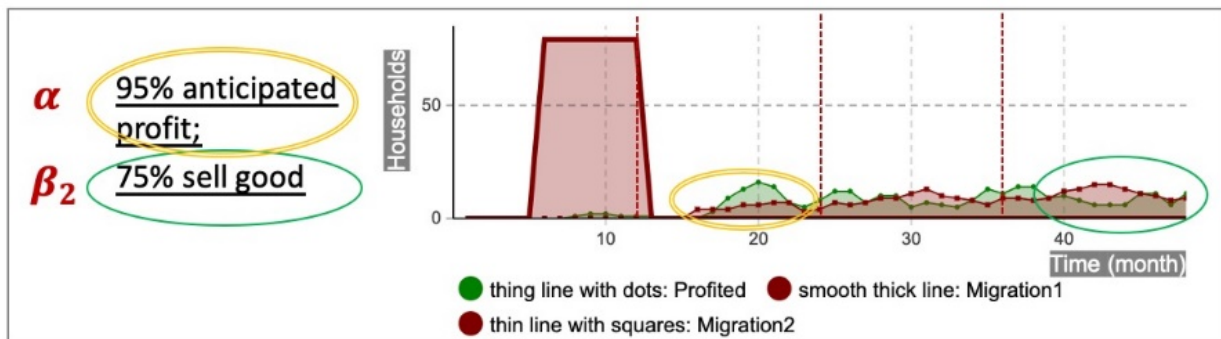
### ***7.4.3 Anticipation and Profit Exploration***

Another two factors are the villagers’ anticipation of profit and the actual profit they gained in the previous year. These are important because the SE needs to set a target for the two factors when doing promotion. Therefore, we need to identify the relationship: 1) between the villagers’ anticipation and their actual improvement of actual economic and social status, and 2) between the villagers’ profit and their decision on whether to do two-season cultivation the next year.

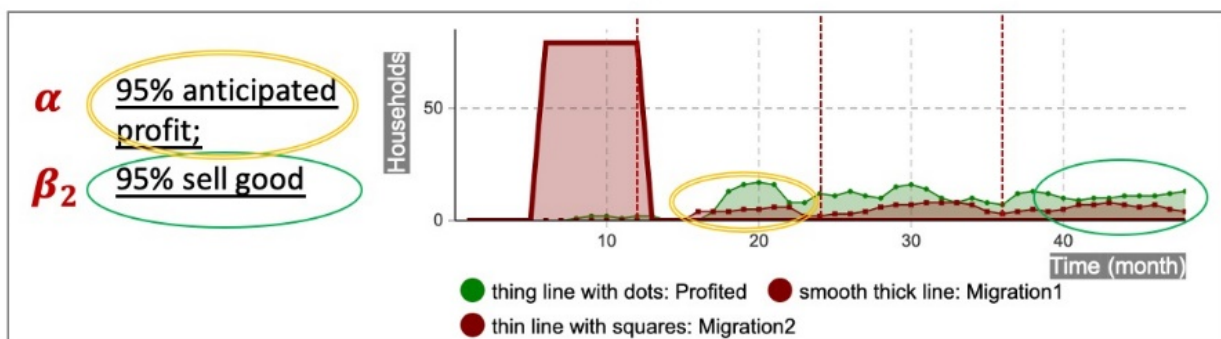
We explore three combinations of two values of  $\alpha$  and  $\beta_2$ . Because  $\beta_3 = \beta_2 \cdot \alpha$ , we do not need to set a value for  $\beta_3$ . We use a distance-based network with influence radius as 75 meters and the SE promotes 50% households. The long-term effect is determined by the short-term effect and the real gain. In Figure 7.11, we show the results from the three scenarios for  $\alpha$  and  $\beta_2$ .



(a)



(b)



(c)

**Figure 7. 11 Simulation Results of Three Scenarios of Anticipation  $\beta_2$  and Profit  $\alpha$**

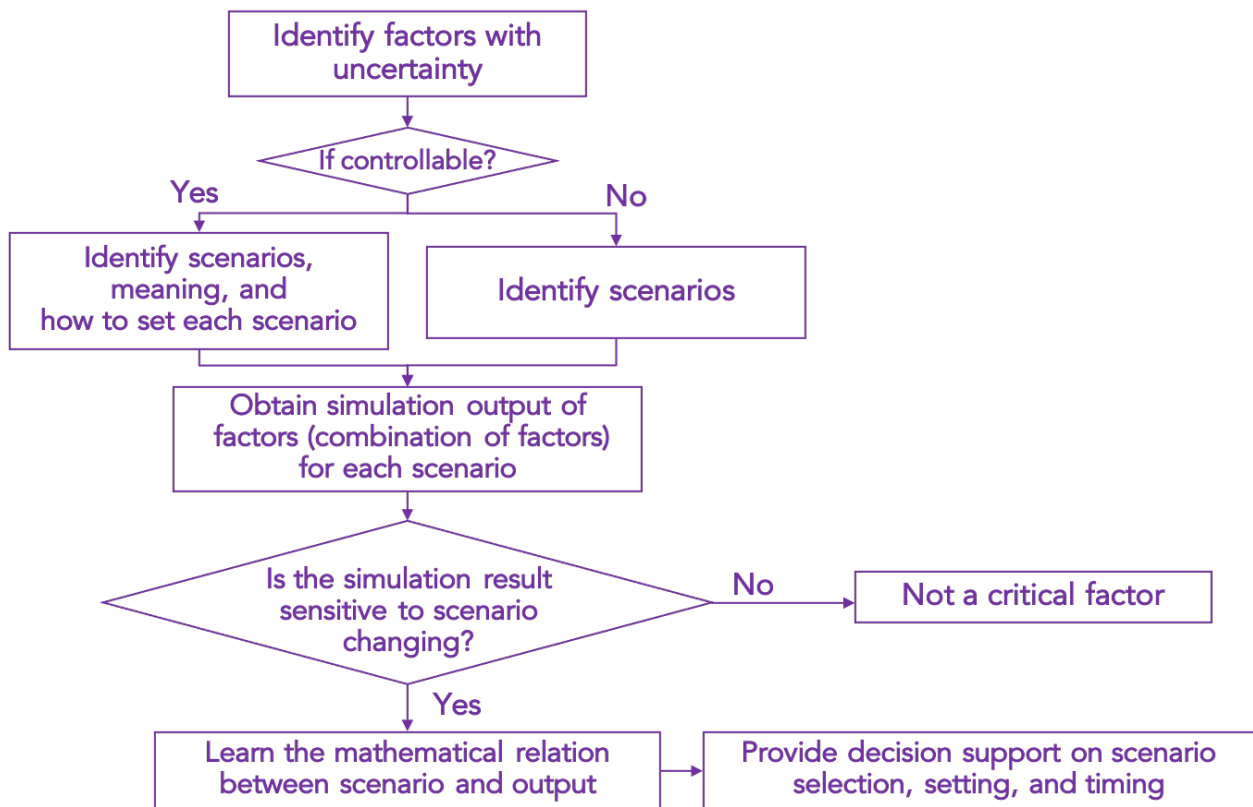
*Exploring the relationship between the anticipation of profit and the actual profit:* the short-term result is more sensitive to anticipated profit; the long-term result is more sensitive to actual profit. A SE can select the appropriate target for anticipated profit when promoting, and take other actions such as improving farmland productivity, expanding market share, or reducing inventory and logistics costs to improve farmers' actual profit to reach the desired migration rate target.

#### ***7.4.4 Closing Remarks***

In this project, agent-based modeling is used to simulate villagers' acceptance of second-season cultivation, and scenario planning is used to identify critical factors that significantly affect the results. We explore scenarios for four factors: network type of the village, the social entrepreneurs (SE's) promotion efforts, the SE's promotion duration, and the villagers' anticipated profit and actual profit, with respect to the short-term and long-term effects on villagers' economic and social status. We observe that among all the explored factors, the SE's promotion duration and villagers' anticipation are critical to the short-term effects, whereas the villagers' real profit is critical for the long-term effect. A SE can select the appropriate scenario to reach their economic and social goals. To make this scenario planning process adaptable for other social-technical-system design projects, we summarize the process in Figure 7.12 and described as follows.

- Identify the factors in the system with uncertainties.
- Determine whether each factor is controllable or uncontrollable.
- For controllable factors, identify their scenarios and the meaning of each scenario based on data, domain expertise, or assumptions, and identify or suggest the ways of setting each scenario.
- For uncontrollable factors, identify possible scenarios, and connect each scenario with system performance, and predict the impact of each scenario on system performance.
- Analyze the sensitivity of the simulation output to each factor and the necessary combinations of multiple factors.

- Identify critical factors – if the simulation output is sensitive to a factor or a combination of multiple factors, then the factor or the combination of multiple factors is a critical factor.
- Identify the quantitative relations among each scenario of the critical factors and the simulation output.
- Provide decision support to the system designer by giving all the scenario-output relations.



**Figure 7. 12 Scenario Planning for Identifying Critical Factors in Simulation**

## 7.5 Role of Chapter 7 in this Dissertation

### 7.5.1 *Summarizing How We Connect Formulation, Approximation, Exploration, and Evaluation*

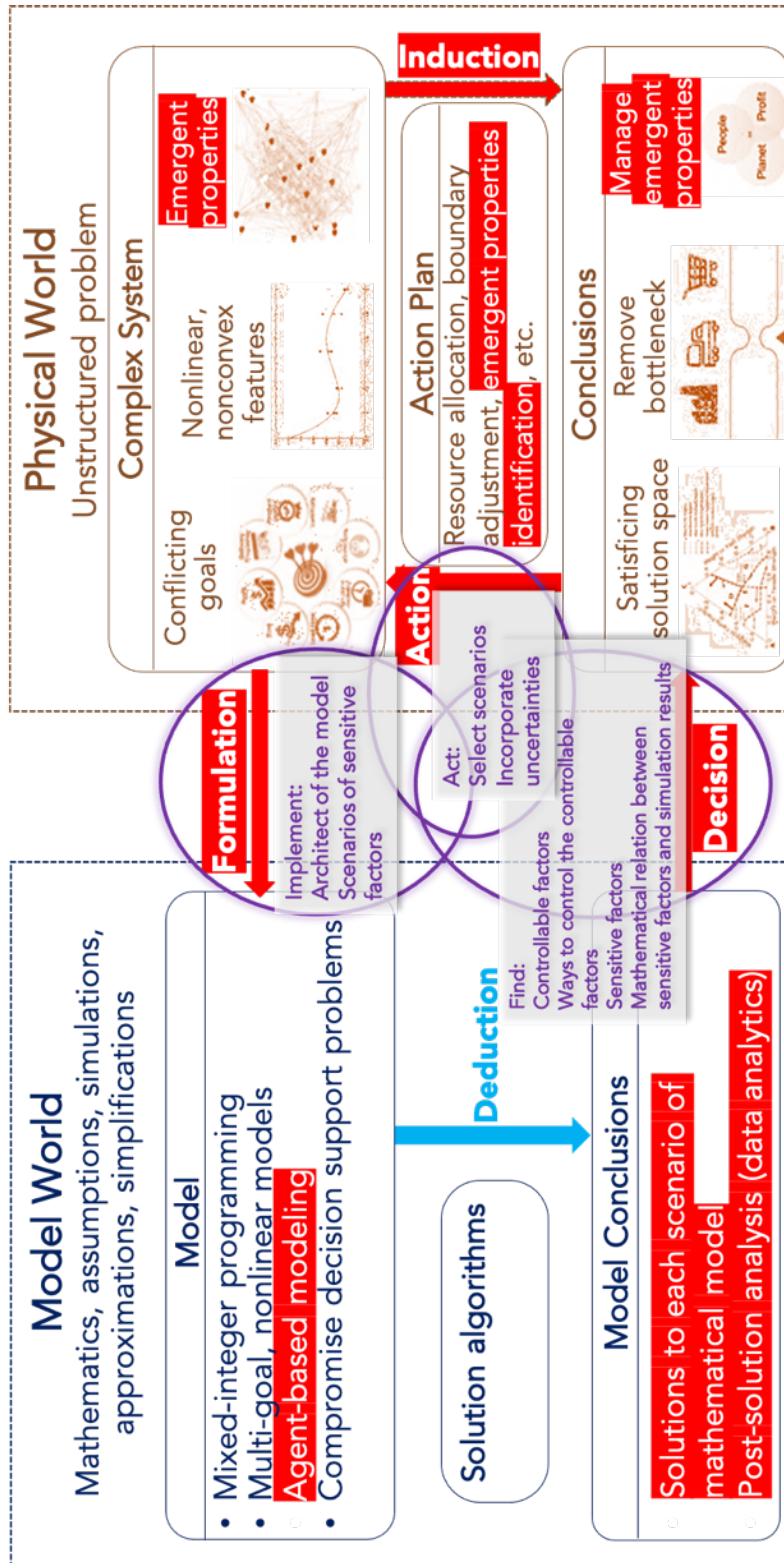
For a wicked problem, such as designing interactions with a social system to get a certain result, designers lack the knowledge and data to formulate the problem. To deal with the problem, we need to answer these questions: what is the boundary of the problem, what are things that can be controlled and leveraged whereas what is not, what factors should be designed as decision variables and what are parameters with uncertainty, what functional relation or cause-and-effect relations among variables and how they evolve over time or with other factors changing, etc. For a social system, even know what each individual's preference and behavior rules, designers may have no idea of the collective behavior of the community because the collective property is nonlinear with the individual property. Hence, in this chapter, we use scenario planning in simulations to identify the critical and controllable factors, the mathematical relationship among the factors, and use it to capture the emergent properties (Figure 7.12). With such knowledge, the appropriate scenarios and interventions are selected to reach the simulation goals.

The essence of Specific Hypothesis 4, “*capture and quantify emergent properties through scenario planning in simulations*” is, to explore the critical and sensitive factors that significantly affect the simulation results, quantify the relationship between the setting of those factors and the results regarding simulation goals, such learned mathematical relations to be updated in the model formulation and approximation are the emergent properties. By evaluating whether some interactions imposed on the system can control the controllable factors to achieve specific simulation results, either in the short term or in a longer time frame, designers gain the knowledge on how to further improve the formulation (such as the architect of the simulation model) and

approximation (ways to control the controllable factors and incorporate uncertainties of sensitive factors).

In other words, we realize the connections among formulation, decision, and action through the activities in the circles, as shown in Figure 7.13. The flowchart of the method is illustrated in Figure 7.12. Through realizing the proposed method for the test problem, promoting the second-season cultivation in a village in India, we capture and model the emergent properties of the social system through managing the critical and sensitive factors.





**Figure 7. 13 The Procedures Involved in Formulation-Approximation-Exploration-Evaluation – Establish the information exchange, knowledge awareness, and instructions sharing among formulation, decision, and action**

### ***7.5.2 Summarizing How We Realize Type I, II, & IV Robust Design***

For the test problem, promoting the second-season cultivation in a village, Type I uncertainty is identified as the price of the agricultural products, and it affects the real profit  $\alpha$ , so we incorporate the uncertainty as the different scenarios of the value of  $\alpha$  (Table 7.7 and Figure 7.5). As a result, the simulation results vary with  $\alpha$ ; see Figure 7.11 (a) and (c). In the short term, price is uncontrollable by farmers or social entrepreneurs, but in the long term, they can intervene through market development, technology adoption, or budget control. As the time frame for this project is four years, we take product price as a factor that cannot be directly controlled, so it is a parameter, and the price fluctuation is Type I uncertainty.

Type II uncertainty is represented in our test problem as the unknown and uncertain impact of different promotion efforts and promotion durations on the short-term and long-term promotional effects. The scenarios that we explore and their physical meaning are given in Table 7.7. The results of various scenarios of promotion effort are shown in Figure 7.8 and 7.9 and summarized in Table 7.8. The results of various scenarios of promotion duration are shown in Figure 7.10 and summarized in Table 7.9.

Type IV uncertainty is recognized as the unknown feature and the uncertainty in the villagers' estimation management. Villagers' original estimation of gaining more profit by growing the second season crops than migrating (profit estimation) without underground water,  $\beta_1$ , is known, but their profit estimation with underground water,  $\beta_2$ , depends on the real profit  $\alpha$  and their original profit estimation  $\beta_1$ . In other words, villagers' real profit of the previous year and the promotion results comprehensively impact their estimations of the next year. So, the uncertainty of  $\beta_2$  is the result of managing Type I and II uncertainty. That is why we define it as Type IV

uncertainty. The uncertainty is described in Table 7.7. The results are shown in Figure 7.5 and Figure 7.11 (b) and (c).

By implementing the proposed scenario planning method (Figure 7.12), we can identify the solution space that is relatively insensitive to the Type I, II and IV uncertainty that we need to manage in a specific problem. In this way, we realize Type I, II & IV robust design. see the summary in Table 7.10 as the closing remarks of Table 3.2 regarding the robust design realization and uncertainty management for Test Problem 4.

**Table 7. 10 Summary of Test Problems 4 regarding Type I, II, & IV Uncertainty Management**

RD Type	RDI-II			RDIII	
				RDIV	
Method	M1: Formulation-Exploration Framework		M2: Adaptive Linear Programming with Parameter Learning (ALPPL)	M3: Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)	<b>M4: Scenario Planning in Agent-Based Modeling</b>
Chapter	Ch 4		Ch 5	Ch 6	Ch 7
Test Problem	T1: Dam network	T2: Supply chain	T3: Hot rolling process chain	T4: Thermal system	<b>T5: Promoting second-season farming</b>
Type I	<i>Uncertainty in timing and amount of inflow – Table 4.7</i>	<i>Uncertainty in demand side – Figure 4.25</i>	<i>Uncertainty in hyper parameter setting – Table 5.7, Figure 5.16</i>	<i>Uncertainty in parameter setting in solution algorithm (Starting point of searching) – call XPLORE in DSIDES</i>	<b><i>Uncertainty in price (Price of agriculture products) – <math>\alpha</math>, Table 7.7, Figure 7.11 (a) and (c)</i></b>
Type II	<i>Uncertainty in outflow (water release target) – Table 4.5</i>	<i>Uncertainty in supply side - Table 4.15</i>	<i>Uncertainty in user preferences – Table 5.6</i>		<b><i>Promotion effort and timing – Table 7.7-7.9, Figure 7.8-7.10</i></b>

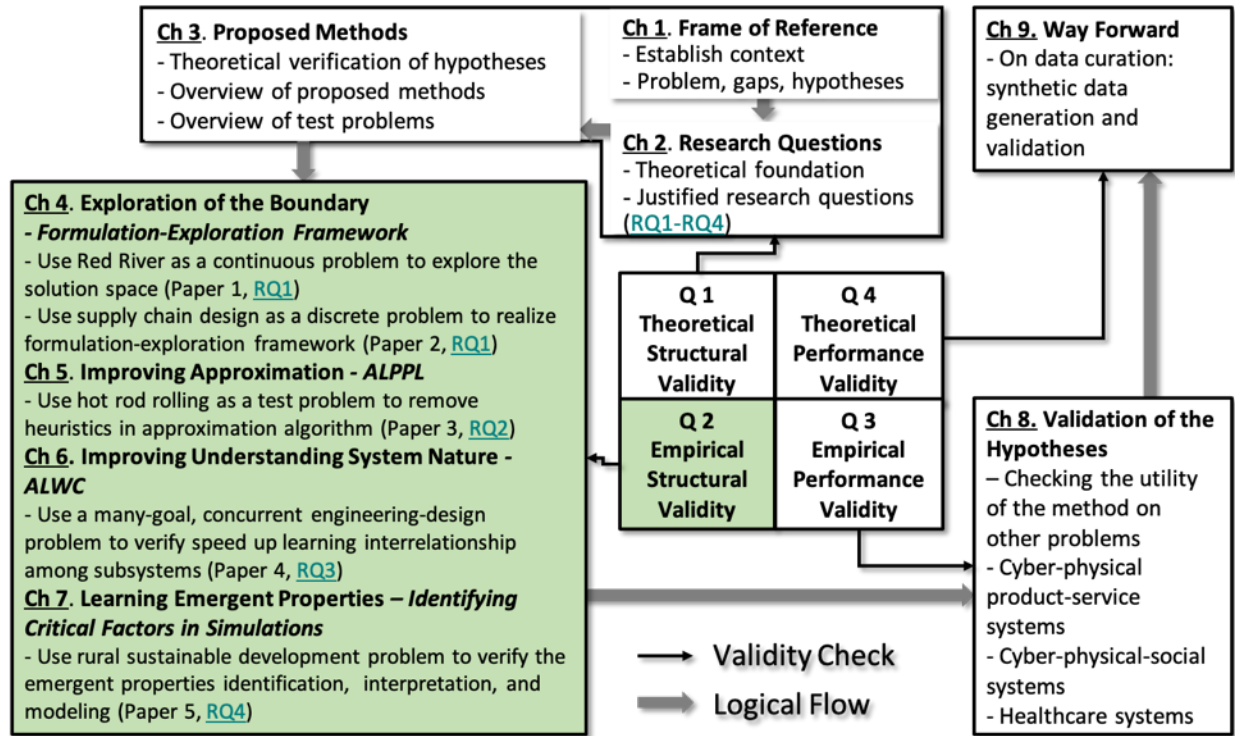
Type III			<i>Uncertainty in model approximation due to heuristics in approximation – Table 5.5</i>	<i>Uncertainty in model approximation (ways of combining multiple goals) – leveling-weighting-clustering algorithms (Figure 6.13)</i>	
Type IV				<i>Uncertainty in using domain knowledge to simplify the model (fixing decision variables and selecting design scenarios) – “fixing variable” and “XPLORE” modules in DSIDES, and leveling and clustering algorithm in the ALWC</i>	<b><i>Interventions that change the mathematical relation among promotion and result (developing local market) – <math>\beta_2</math>, Table 7.7, Figure 7.5, Figure 7.11 (b) and (c)</i></b>

RD – robust design  
M – method  
EVe – empirical verification of the method  
T – test problem

### 7.5.3 Role of Chapter 7

In Chapter 7, given the frame of references on designing promotions using agent-based modeling, which is an extension of the frame of references in Chapter 2. A method, a framework for identifying critical factors through scenario planning in agent-based modeling, is proposed to identify critical factors and quantify the scenario associated with simulation results for modeling emergent properties. A test problem, promoting the second-season cultivation in an island village in India, is used to verify the proposed methods. Different scenarios are identified and explored so that decision support can be provided to social entrepreneurs (SEs). Agent-based modeling (ABM) is used to simulate villagers’ acceptance of second-season cultivation, growing two crops a year instead of one. We explore the possibility of second season cultivation to improve the villagers’ social-economic status in both the short-term and the long-term. The proposed method of capturing and making use of critical factors in influencing individuals’ behavior in a community can be used in other projects. Research Question 4 is addressed.

**Role of Chapter 4 to 7:** From Chapter 4 to Chapter 7, Quadrant 2 of the Research Questions are addressed; see Figure 7.13. The empirical structural validity of the research questions is answered by testifying the hypotheses and proposed methods using test problems.



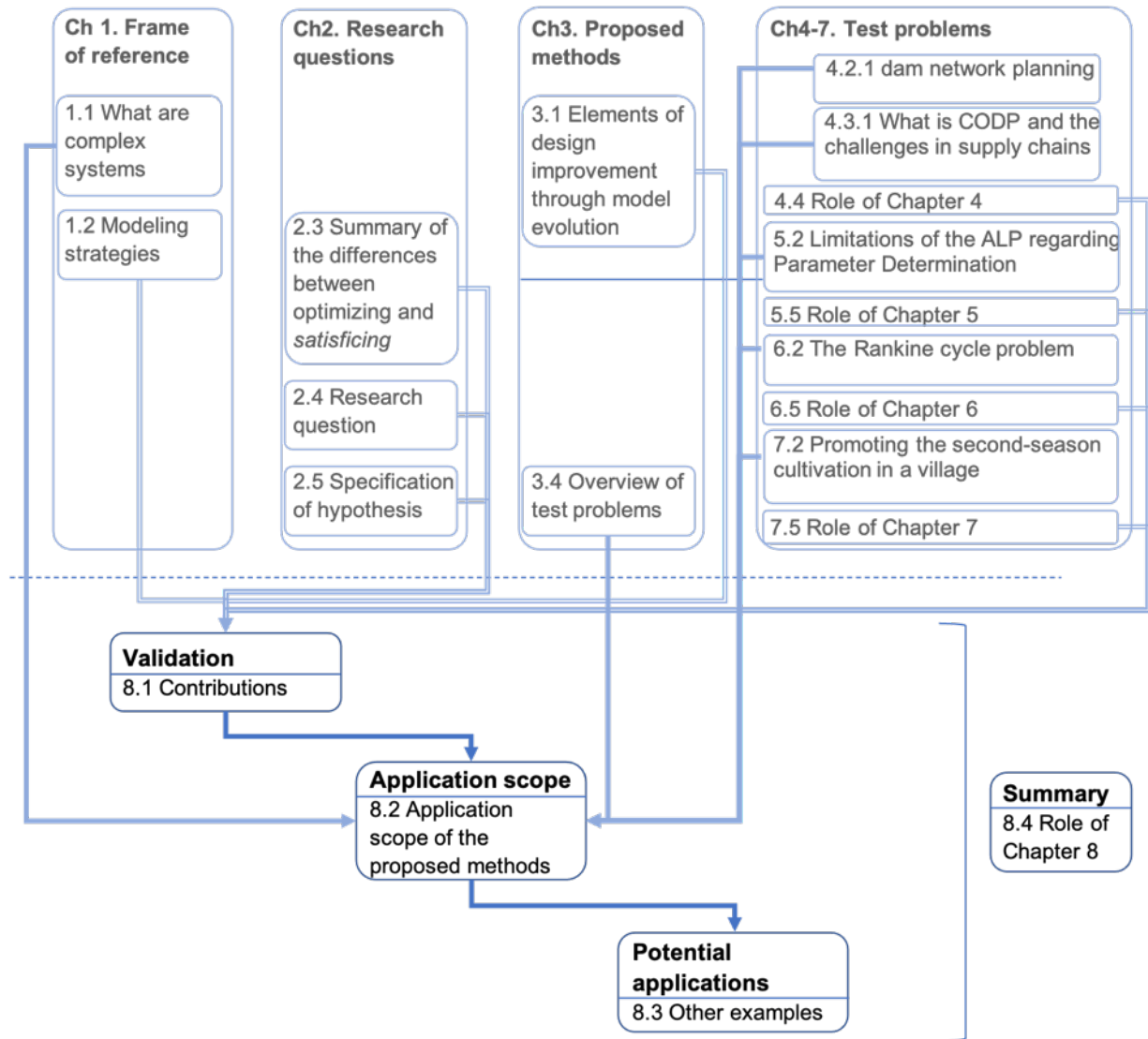
**Figure 7. 14 Finishing Empirical Structural Validity in Chapter 4, 5, 6, and 7**

## CHAPTER 8 VALIDATION OF THE HYPOTHESES IN REALIZING MODEL EVOLUTION

### *– SUMMARY AND VALIDATION OF MODEL EVOLUTION*

*In Chapter 8, summarize the empirical validation of the model evolution methods. We give closing remarks of the answer to the research questions and validate the hypotheses. In this chapter, we recap the contributions, the application scope of the proposed methods, and introduce more examples.*

In Chapter 8, see Figure 8.1, the contributions are summarized in Section 8.1, the application scope of each method is clarified in Section 8.2, other examples are introduced in Section 8.3, and the role of Chapter 8 is concluded in Section 8.4.



**Figure 8. 1 Organization of Chapter 8**

In Table 8.1, we illustrate how the earlier sections support Chapter 8 and how Chapter 8 recap the previous parts of this dissertation, which are the descriptions of how each arrow in Figure 8.1 works. In Section 8.1, we review the strength of the *satisficing* strategy (the strategy selected as the baseline method in this dissertation, through cDSP as the model construct, the ALP as the solution algorithm, implemented in DSIDES), summarize the answer to the research questions, validate the hypotheses, and reflect the tasks we finish to create knowledge. In Section 8.2, we

review the characteristics and challenges of complex systems design and give the application scope of the proposed methods, which lead to more applications in Section 8.3.

**Table 8. 1 The Support for Chapter 8 in Previous Sections**

Ch1 Sections	Ch2 Sections	Ch3 Sections	Ch4-7 Sections	Supporting Relation	Ch8 Sections
1.2				Select the appropriate strategy – <i>satisficing</i>	8.1
	2.3			The uniqueness and benefits of our strategy	
	2.4			Answer research questions	
	2.5			Validate hypotheses	
		3.1		Finish tasks	
			4.4 5.5 6.5 7.5	Contributions of the proposed methods	
1.1				Characteristics of complex systems and challenges in design	8.2 8.3
		3.4		Features of examples of complex system	
			4.2.1 4.2.3 5.2 6.2 7.2	Problem statement in the context of test problems	

## 8.1 Contributions

In this dissertation, we deal with the difficulties in designing complex systems. The model evolution loop is a concept and a tool facilitating designers to manage design complex systems. The model evolution is achieved by connecting the multiple stages in the complex-system design. Such connections include passing through information among stages to make or change decisions, establish mathematical relationships among parameters, actions, or heuristics of different stages, etc. We prove that by connecting formulation, approximation, exploration, and evaluation in designing a complex system, designers can manage four types of uncertainty. The four types are noise factors (Type I), variations in design variables (Type II), variations in mathematical models (Type III), and uncertainties caused by managing the previous three types of uncertainty (Type IV).



What has been done regarding addressing the research questions and verifying the hypotheses are summarized in Table 8.2. In Table 4.1, 5.5, 6.1, and 7.1, we give full versions of Table 8.2 with how we address Research Question 1, 2, 3, and 4, respectively. In Table 8.3, we summarize the research gap, hypothesis, four types of robust design, etc., in Chapter 1 to 3.

**Table 8. 2 Summary of Addressing the Research Questions and Verifying the Hypotheses – with section number**

Chapter	Actions	
Ch1	RG – 1.5.1	
	H – 1.5.2	
Ch2	RDIV – 2.4.1	RDIII – 2.4.1
	RQ4 – 2.4.2	RQ3 – 2.4.2
	SH4 – 2.5	SH3 – 2.5
	TVe4 – 3.2.4	TVe3 – 3.2.3
	M4 – 3.3.4	M3 – 3.3.3
		M2 – 3.3.2
Ch3		MI – 3.3.1
		EVe1 – 4.2.3/4.3.3/4.4.1
		SQT1 – 4.1/4.3.2
		AQ1 – 4.2.6/4.3.2
Ch4		VRDI-II – 4.4.2/5.5.2
Ch5		EVe2 – 5.2/5.5.1
		SQT2 – Table 5.1
		AQ2 – 5.5.1
Ch6		VRDIII – 5.5.2/6.5.2
		EVe3 – 6.3/6.5.1
		SQT3 – Table 6.1
Ch7		AQ3 – 6.5.1
		VRDIII – 6.5.2/7.5.2
Ch8		EVe4 – 7.3/7.5.1
		SQT4 – Table 7.1
		AQT4 – 7.5.1
Ch9		CO – 8.1/8.4
		EVa – 8.2/8.3
		TE

Nomenclature
RG – give research gaps
H – give hypotheses
RD“X” – tie to Type “X” robust design
RQ“X” – pose Research Question “X”
SH“X” – specify hypothesis “X”
TVe“X” – theoretically verify hypothesis “X”
M“X” – introduce method “X”
EVe“X” – empirically verify hypothesis “X”
SQT“X” – specify research question “X” in the context of test problems “X”
AQ“X” – answer research question “X”
VRD“X” – validate Type “X” robust design
CQ – closure the answers to research questions
EVa – empirically validate hypotheses
TE – theoretically extend the research

**Table 8.3 Addressing the Research Question 1 and Verifying the Hypotheses**

Chapter	Actions				Ch1	Ch2	Ch3	Ch4-7	Ch8	Ch9
<p>RG: How can designers realize model evolution using satisficing strategy so that they can manage chaos in the physical world, reduce the risk of losing an optimal solution, and discover domain-independent knowledge to update metaheuristics?                      H: by connecting the multiple stages of design and passing information through them, designers can improve their decision models in iterations.</p> <p>RDIV: What is the method that allows sensing and managing Type IV uncertainty?                      RDIII: What is the mathematics in the design method that allow the exploration of the solutions relatively insensitive to Type III uncertainty?                      RQ2: What is the method to evolve model to update metaheuristics?                      RQ3: What is the method to speed up learning the system nature?                      RQ4: What is the method that allows passing the information through multiple scales of a system?                      SH4: Capture and quantify emergent properties through scenario planning in simulation.                      TVe4: Learn emergent property by connecting formulation, approximation, and exploration, and evaluation.                      M4: Scenario planning in agent-based modeling</p>	<p>RDII: What are the mechanisms and procedures that enable the exploration of the solutions relatively insensitive to Type I and Type II uncertainty?</p>				<p>RQ1: What is the method to evolve model boundary?                      SH1: Explore the sensitivity of the segments of the model boundary and improve accordingly.                      TVe1: Explore the boundary of the system by connecting formulation with exploration.</p>	<p>M1: Formulation-Exploration framework</p>				
					<p>SH2: Explore the sensitivity of the segments of the model boundary and improve accordingly.                      TVe2: Improve the approximation by connecting approximation, exploration, and evaluation.</p>	<p>M2: Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL)</p>				
					<p>SH3: Learn system nature such as interrelationship among subsystems and reorganize them based on it.                      TVe3: Speed up learning system nature by connecting formulation, exploration, and evaluation.</p>	<p>M3: Adaptive Leveling-Weighting-Clustering algorithm (ALWC)</p>				
<p>RG – give research gaps                      H – give hypotheses                      RD“X” – tie to Type “X” robust design                      RQ“X” – pose Research Question “X”                      SH“X” – specify hypothesis “X”                      TVe“X” – theoretically verify hypothesis “X”                      M“X” – introduce method “X”                      EVe“X” – empirically verify hypothesis “X”                      SQT“X” – specify research question “X” in the context of test problems “X”                      AQ“X” – answer research question “X”                      VRD“X” – validate Type “X” robust design                      CQ – closure the answers to research questions                      EVa – empirically validate hypotheses                      TE – theoretically extend the research</p>										
<p>EVe, SQT, AQ</p>										
<p>C1, EVa</p>										
<p>TE</p>										

### ***8.1.1 Summarizing the Theoretical Foundation***

In Chapter 1, we frame the reference from 1) design strategies – Table 1.3-1.5, and 2) challenges in the model-based realization of complex systems – Figure 1.7. In the optimizing literature, assuming models are complete and accurate, the authors seek optimal solutions, meeting both the necessary and sufficient Kuhn-Tucker conditions, which are sensitive to errors, incompleteness, and variations embodied in the decision model. In the *satisficing* literature, with the awareness that models can be wrong and with various fidelity, the authors seek good enough solutions that are relatively insensitive to errors and uncertainties, but they have to rely on domain knowledge and metaheuristics, which make design knowledge irreproducible. Therefore, we identify the research gap to be addressed in this dissertation – *How can designers realize model evolution using satisficing strategy so that they can manage chaos in the physical world, reduce the risk of losing an optimal solution, and discover domain-independent knowledge to update metaheuristics? We hypothesize that by connecting the multiple stages of design and passing information through them, designers can improve their decision models in iterations, which in this dissertation is defined as “model evolution.”*

In Chapter 2, we pose research questions and specify the hypothesis. By analyzing the assumptions of using the Kuhn-Tucker conditions to seek optimal solutions, we conclude that using optimizing strategy, designers have to accept at least three assumptions (or meet these three requirements) to guarantee the optimal solution works, 1) models are perfect abstraction of the physical world, 2) all equations of a decision model should be continuous and differentiable, and 3) the convexity degree of at least one non-zero linear combination scenario of all constraints should be higher than the convexity degree of the objective function. Whereas using *satisficing* strategy, only the second assumption is required. Through five toy problems, we illustrate how the results of using two

strategies turn out different. The differences are caused in all four stages of the design loop – Formulation (using goals with target values as the right-hand side parameters), Approximation (second-order sequential linearization and accumulated linearization), Exploration (combining interior-point searching and vertex searching), and Evaluation (allowing violations). Based on all these discussions, we choose *satisficing* to be our strategy to manage complex-system design problems and improve it by connecting multiple stages. Therefore, we justify the research questions regarding requirements (four types of robust design, Table 2.14) and tasks (four stages of design loop, Table 2.15).

In Chapter 3, to address the research questions, we theoretically verify the specified hypotheses. We demonstrate the feasibility of finishing the research tasks of connecting multiple stages of the design loop through information sharing and performance evaluation. The purpose of running the design evolution loop and connecting the stages is to update the metaheuristics and manage more types of uncertainty in designs. We propose the methods in Chapter 3 (Figure 3.2). To explore the boundary of the design problems, we propose to connect the formulation and exploration using the Formulation-Exploration framework (M1, Figure 3.12). To improve the robustness of the approximation method under the *satisficing* strategy, we propose to use parameter learning to improve the adaptive linear programming algorithm (M2, Figure 3.13). To aware subsystems and explore the interrelationship among them, we employ unsupervised learning to learn the orthogonality and correlation among the goals and update the goal combination based on it (M3, Figure 3.14). To learn the emergent property of a design, such as getting to know the critical factors in scenario planning of a decision model, we standardize the process of scenario planning in simulations (M4, Figure 3.15).

### ***8.1.2 Summarizing the Test Problems***

From Chapter 4 to 7, we demonstrate the proposed methods M1-M4 using five test problems – T1.1, T1.2, T2, T3, T4, and T5.

***T1.1 – Dam network planning.*** We propose a three-step method to plan the monthly water release from each reservoir for a 14-dam network to satisfy three user-groups. The boundary of the reservoirs is evolved to enable the system to be less sensitive to uncertainties in water inflows.

***T1.2 – CODP positioning for a supply chain.*** We expand the three-step method into the formulation-exploration framework and use it to position the customer order decoupling point (CODP) for a single-product supply chain regarding different phases of the product life cycle (PLC). We remove the bottleneck in iterations by evolving the CODP and relevant parameters.

***T2 – Improving the hot rod process chain.*** We incorporate parameter learning in the adaptive linear programming algorithm. The improved algorithm, the adaptive linear programming algorithm with parameter learning (ALPPL), facilitates designers to evaluate the performance of the approximation regarding multicriteria and learn the association between parameter setting and approximation performance so as to update the metaheuristics applied in parameter setting.

***T3 – Rankine cycle thermal system design.*** Unsupervised learning is used to identify subsystems based on the interrelationship among the goals and reorganize them to boost the system performance. Learning the tradeoffs among subsystems and providing scenarios for users to realize certain tradeoffs can make the system more robust.

***T4 – Learning emergent properties of a social system under interventions.*** Scenario planning in agent-based modeling is processed to identify the critical and sensitive factors in the interactions

with the system and the interactions among the factors in the system so corresponding actions can be taken to reach the target of the system goals.

### ***8.1.3 Summarizing the Answer to the Research Questions***

In summary, the answer to the research questions is new knowledge: connecting multiple stages in the design evolution cycle. In Table 8.4, we summarize the new knowledge, how does it help answer each research question, what differences do we make, and the relevant publications. Table 8.4 is another format of the illustration of Figure 3.2. The differences between “before the new knowledge” and “after the new knowledge” is the major contributions in this dissertation. The essence of why we can make the difference is that we choose the *satisficing* strategy and fill in the research gaps.

**Table 8. 4 New Knowledge in this Dissertation**

Publication	After the New Knowledge	Before the New Knowledge	Answer to RQ – new knowledge	Research Question (RQ)	Chapter
Guo, L., Zamanisabzi, H., Neeson, T.M., Allen, J.K., Mistree, F., 2018, “ Managing Conflicting Water Resource Goals and Uncertainties in a Dam-Network by Exploring the Solution Space,” Journal of mechanical design, 141(3): 031702.	Solutions are on the boundary of the mathematical model but away from the physical boundary, so they are relatively insensitive to the variation in the boundary (formed by constraints or bounds); Designers can give suggestion on model improvement regarding both goal completion and robustness of the solution; Designers can improve the goal completion without sacrificing the model robustness, and vice versa.	Solutions are on the boundary of the mathematical model (formed by the constraints or bounds) as well as the boundary of the physical system, so they are sensitive to the variation in the system boundary; No suggestion on model improvement regarding both goal completion and robustness of the solution; Improving the goal completion results in the sacrificing the model robustness, and vice versa.	The Three-Step Exploration Method	RQ1: What is the method to evolve model boundary?	<b>Ch4</b>
Guo, L., Chen, S., Allen, J.K., Mistree, F., 2019, “ Designing the Customer Order Decoupling Point to Facilitate Mass Customization,” <i>ASME Design Automation Conference</i> , Anaheim, CA, USA, Paper Number DETC2019-97379.	Heuristics in algorithm are evaluated using criteria with consideration of goal completion, robustness and computational complexity; Heuristics are replaced with insight obtained from post-solution analysis.	Heuristics in the algorithm (Golden Section Search in ALP) have not been evaluated and improved; No criteria other than the goal completion for solution quality evaluation; No mechanism of replacing heuristics with insight obtained from calculation and analysis.	The Adaptive Linear Programming Algorithm with Parameter learning (ALPPL)	RQ2: What is the method to evolve model to update metaheuristics?	<b>Ch5</b>
Guo, L., Milisavljevic-Syed, J., Allen, J.K., Mistree, F., “ Formulation-Exploration Model for Managing Many-Goal Engineering Design Problems,” in preparation, target journal: Engineering Optimization.	Decisions support on the priority and combination of the multiple goals can be provided to designers even without any domain knowledge; Visualization of more than three goals is possible; More tradeoff scenarios are	The priority and combination of multiple goals rely on domain expertise; It is difficult to visualize the results for models with more than three goals; The tradeoff scenarios are limited.	The Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)	RQ3: What is the method to speed up learning the system nature?	<b>Ch6</b>
Guo, L., Mohebbi, S., Das, A., Allen, J. K., & Mistree, F. (2020). A Framework for the Exploration of Critical Factors on Promoting Two-Season Cultivation in India. Journal of Mechanical Design, 142(12).	Designers can capture and incorporated emergent properties of a system under interventions into the model; designers can identify critical and sensitive factors and quantify them with respect to system goals.	The emergent property of a system under interventions are not captured and incorporated into the model; No mechanism to identify critical and sensitive factors and quantify them with respect to system goals.	Scenario planning workflow for Agent-Based Modeling	RQ4: What is the method that allows passing the information through multiple scales of a system?	<b>Ch7</b>



#### ***8.1.4 Summarizing the Four Types of Robust Design***

In summary, to realize the four types of robust design, we manage the four types of uncertainty in designing complex systems. In different systems, each type of uncertainty may have various representations. The four types of uncertainty in each test problem, the new knowledge to realize the four types of robust design, and the number of the corresponding tables and figures containing more details are summarized in Table 8.5, which is another format of the illustration of Figure 3.17.

**Table 8. 5 Summary of the Realization of the Four Types Robust Design**

Questions regarding RD Type		RDI-II: What are the mechanisms and procedures that enable the exploration of the solutions relatively insensitive to Type I and Type II uncertainty?		RDIII: What is the mathematics in the design method that allow the exploration of the solutions relatively insensitive to Type III uncertainty?		RDIV: What is the method that allows sensing and managing Type IV uncertainty?			
		M1: Formulation-Exploration Framework		M2: Adaptive Linear Programming with Parameter Learning (ALPPL)		M3: Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)		M4: Scenario Planning in Agent-Based Modeling	
Realization of RD – Proposed Method		Ch 4		Ch 5		Ch 6		Ch 7	
Chapter		Ch 4		Ch 5		Ch 6		Ch 7	
Uncertainty Test Problem		T1.1: Dam network planning T1.2: CODP positioning for a supply chain		T2: Improving the hot rod process chain		T3: Rankine cycle thermal system design		T4: Learning emergent properties of a social system under interventions	
Type I		Uncertainty in timing and amount of inflow – Table 4.7 Uncertainty in demand side – Figure 4.25		Uncertainty in hyper parameter setting – Table 5.7, Figure 5.16		Uncertainty in parameter setting in solution algorithm (Starting point of searching) – call XPLORE in DSIDES		Uncertainty in price (Price of agriculture products) – $\alpha$ , Table 7.7, Figure 7.11 (a) and (c)	
Type II		Uncertainty in outflow (water release target) – Table 4.5 Uncertainty in supply side - Table 4.15		Uncertainty in user preferences – Table 5.6				Promotion effort and timing – Table 7.7-7.9, Figure 7.8-7.10	
Type III				Uncertainty in model approximation due to heuristics in approximation – Table 5.5		Uncertainty in model approximation (ways of combining multiple goals) – leveling-weighting-clustering algorithms (Figure 6.13)			
Type IV						Uncertainty in using domain knowledge to simplify the model (fixing decision variables and selecting design scenarios) – “fixing variable” and “XPLORE” modules in DSIDES, and leveling and clustering algorithm in the ALWC		Interventions that change the mathematical relation among promotion and result (developing local market) – $\beta_2$ , Table 7.7, Figure 7.5, Figure 7.11 (b) and (c)	

RD – robust design

M – method

T – test problem

## **8.2 Application Scope of the Proposed Methods**

### ***8.2.1 Application Scope of the Design Evolution Loop***

For designing any complex system with multiple types of uncertainty, using the design evolution loop, designers can evolve the boundary of the system, deal with discrete variables as well as continuous ones, use heuristics to proceed designing and improve the design by updating the heuristics, reorganize the subsystems based on their interrelationship, and learn and manage emergent properties.

The design evolution loop is an open framework. Connecting multiple stages in the design evolution loop may have different representations. The methods proposed in this dissertation are four examples of many implementations of connecting the stages and evolve the design. The contributions, potential contribution, and scope of application of each method (M1-M4) are summarized hereafter.

### ***8.2.2 Application Scope of M1 – Formulation-Exploration Framework***

The three-step method can be applied to improve the robustness of a mechanical engineering system. In mechanical design, when multiple goals conflict with each other and design preferences evolve with time, designers can use step 1 to explore the weight space and give rules on weight selection dynamically.

To improve the robustness of the system, a reasonable buffer is added to ensure that the solution does not approach the physical boundary too closely, so it can be relatively insensitive to the uncertainties, and our three-step method is a way of doing this. The advantage of our method is that we boost the potential of the physical system while improving its robustness; hence, we neither sacrifice system robustness for a better performance nor do the opposite.

In addition, the satisficing space allows designers to have a relatively insensitive design space and options with awareness of the system output, which is useful in concurrent design and multistage design.

The Formulation-Exploration framework can be used in designing mass-customized products at mass production costs. It is especially useful for continuously improving a system with conflicting goals, evolving design preferences, and multiple players who pursue common interests.

### ***8.2.3 Application Scope of M2 – Adaptive Linear Programming Algorithm with Parameter Learning (ALPPL)***

ALPPL can be applied to multi-goal engineering-design problems, especially when goals conflict with one another, the priority of the goals evolves with the environment changes, and the output of the model must be insensitive to model errors and variations.

The rule-based parameter learning can be used to improve other algorithms, especially when there are no customizable criteria for the evaluation of the algorithm performance, or the algorithm performance is highly sensitive to some critical parameters that are determined with heuristics or human intuition while the critical parameters are not updated based on the algorithm performance during the design iterations.

### ***8.2.4 Application Scope of M3 – Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)***

For concurrent engineering design problems with conflicting and evolving requirements in multiple disciplines that are not clear to the designers in the modeling stage, it is difficult to manage the interactions and coupling effects between subsystems using the combination forms of the goals. Using the ALWC algorithm, designers can capture the knowledge on the division, interaction, and

coupling scenarios among the subsystems, therefore, they can leverage the tradeoffs among conflicting requirements concurrently.

### ***8.2.5 Application Scope of M4 – Scenario-Planning for Simulations***

This proposed scenario-planning process allows designers in various fields to perform simulations and identify critical factors in their systems and select specific scenarios that accommodate different site-specific input values or domain-dependent knowledge to reach their goals.

## **8.3 Other Examples**

### ***8.3.1 Network Planning for Improving Hospital Visiting Process***

#### **- AN EXPANSION OF DAM NETWORK PLANNING**

When patients visit hospitals, they can be subject to long wait times due to operational inefficiencies and bottlenecks. Moreover, long wait times decrease patient satisfaction and patient happiness. There are many ways to model healthcare systems, including agent-based models, which can track individual patients and their movements through a hospital, and network flow models, which can model larger groups of patients and their interaction within a hospital. While agent-based models can detect bottlenecks by evaluating the patients flow speed, a network flow model can easily detect bottlenecks by applying a layer of abstraction to the healthcare network. In the network flow model, patients flow along the capacitated edges of a network while receiving treatment at the nodes, wherein it becomes trivially easy to identify bottlenecks by looking at the flow in and flow out of nodes. These bottlenecks manifest themselves in metrics used to evaluate the model run, including efficiency and wait times. Data regarding capacities of the edges for a network flow model are taken from an agent-based model of a case study of a primary care clinic.

Data describing patient flow through network facilities is difficult to find, so synthetic data is generated using an adaptation of the random subspace method. Ensemble runs of the network flow model are created to account for uncertainty in the synthetic data, culminating in a distribution analysis for the various metrics. By changing the topology of the network flow model, bottlenecks are removed, increasing efficiency in the model and decreasing patient wait times. Furthermore, the network flow model is sensitive to the constraints of the random subspace, and care should be taken when initializing the model.

There are several potential contributions to this test problem. Synthetic data is generated via the Random Subspace Method, and the data populates the various ensemble models. Bottlenecks readily appear in network flow models, and they are identified and assessed for potential removal. Moreover, modifying the topology of the model when wait times were long can partially remove a bottleneck at a node, leading to shorter wait times and increased efficiency. After borrowing some model parameters from an agent-based model, the average patient wait times remained similar between the two models, indicating that network flow models can represent the same system effectively.

An opportunity for future work is to apply the network flow model to a scenario where only the topology and the number of patients flowing into the model are known. The goal of the model is to reach a certain level of efficiency, and the idea shares several principles with bagging and training neural networks. The efficiency of the model is a function that depends upon all of the underlying variables while being subject to certain constraints, and taking the gradient of this function yields the direction in which to shift the underlying variables to reach a sufficient level of efficiency. This more general solution to the problem considers a larger solution space and can reach a satisficing solution to the given problem.

### ***8.3.2 Leveraging Social Drivers in Rural Development***

- AN EXPANSION OF LEARNING EMERGENT PROPERTIES OF A SOCIAL SYSTEM UNDER INTERVENTIONS

Development in rural regions is often dependent on policy actions by the government or social entrepreneurs. Generally, these kinds of policies are designed to motivate change through economic incentives, but this is only a short-term solution. For policies to be sustainably successful, long-term results must be produced, and the progress should not always depend on continuous funding. This poses a challenge of how to measure the long-term effects of short-term policies on individual behavior. In this research project, it is hypothesized that qualitative factors such as social drivers are the primary motivators for long term change and that Agent Based Modeling can be used to simulate and predict villager behavior to model the effects of policy on social interaction and behavioral patterns.

This project effectively serves as a proof of concept that social and environmental drivers should be considered in modeling behavioral reactions to policy, not just economic drivers, and that agent-based modeling effectively simulates the propagation of their effects. Most critically, the difference between the socio-economic impact of a driver such as access to electricity, which only alters productivity, versus a driver like family life that influences adoption of those in a villager's social network show that social impacts not only affect behavior, but changes in them across network interactions can have a more drastic impact than a purely economic factor as is normally considered, which is vital shift for development of rural communities.

Therefore, the methods and conclusions of this project can be applied in a myriad of applications where economic considerations have been the primary concern but have underlying social

implication. Further work in this project can be applied in modeling social drivers within networks across disciplines, from how people react in healthcare, to how education can more effectively engage students. Even complex systems such as welfare and policing or the judicial system, which generally focus on cost and data driven deployment of resources can be evaluated from this perspective considering the social network roles of both those administering these programs and systems, and those using or impacted by them.

### ***8.3.3 Knowledge Management in Designing Cyber-Physical Product-Service Systems***

#### **- AN EXPANSION OF POSITIONING CODP IN A SUPPLY CHAIN**

The automation and intelligence highlighted in Industry 4.0 put forward higher requirements for reasonable trade-offs between humans and machines for decision-making governance. However, in the context of Industry 4.0, the vision of decision support for design engineering is still unclear. Additionally, the corresponding methods and system architectures are lacking to support the realization of value-chain centric complex engineered systems design lifecycles. Hence, we identify decision support demands for complex engineered systems designs in the Industry 4.0 era, representing the integrated design problems at various stages of the product value chain. As a response, in this research project, the architecture of a Knowledge-Based Design Guidance System (KBDGS) for cloud-based decision support is presented that highlights the integrated management of complexity, uncertainty, and knowledge in designing decision workflows, as well as systematic design guidance to find satisfying solutions with the iterative process “formulation-refinement-exploration-improvement”. The KBDGS facilitates diverse multi-stakeholder collaborative decisions in end-to-end cloud services. The contribution of this project is to provide design guidance to facilitate knowledge discovery, capturing, and reuse in the context of decision-centric digital design, thus improving the efficiency and effectiveness of decision-making, as well as the



evolution of decision support in the field of design engineering for the age of Industry 4.0 innovation paradigm.

#### **8.4 Role of Chapter 8 in this Dissertation**

In this chapter, the contributions in this dissertation are summarized in Section 8.1, in which, we review the theoretical foundation, recap the test problems, and conclude the answer to the research questions and how we realize the four types of robust; the application scope of the design evolution loop and each method is discussed in Section 8.2; other examples which are extensions of the methods and test problems in this dissertation are stated in Section 8.3.

In summary, in Chapter 8, we provide the empirical performance validity; see Figure 8.2. The empirical performance validity is done by reviewing what differences are made by using the proposed methods and clarifying the scope of application of the methods.

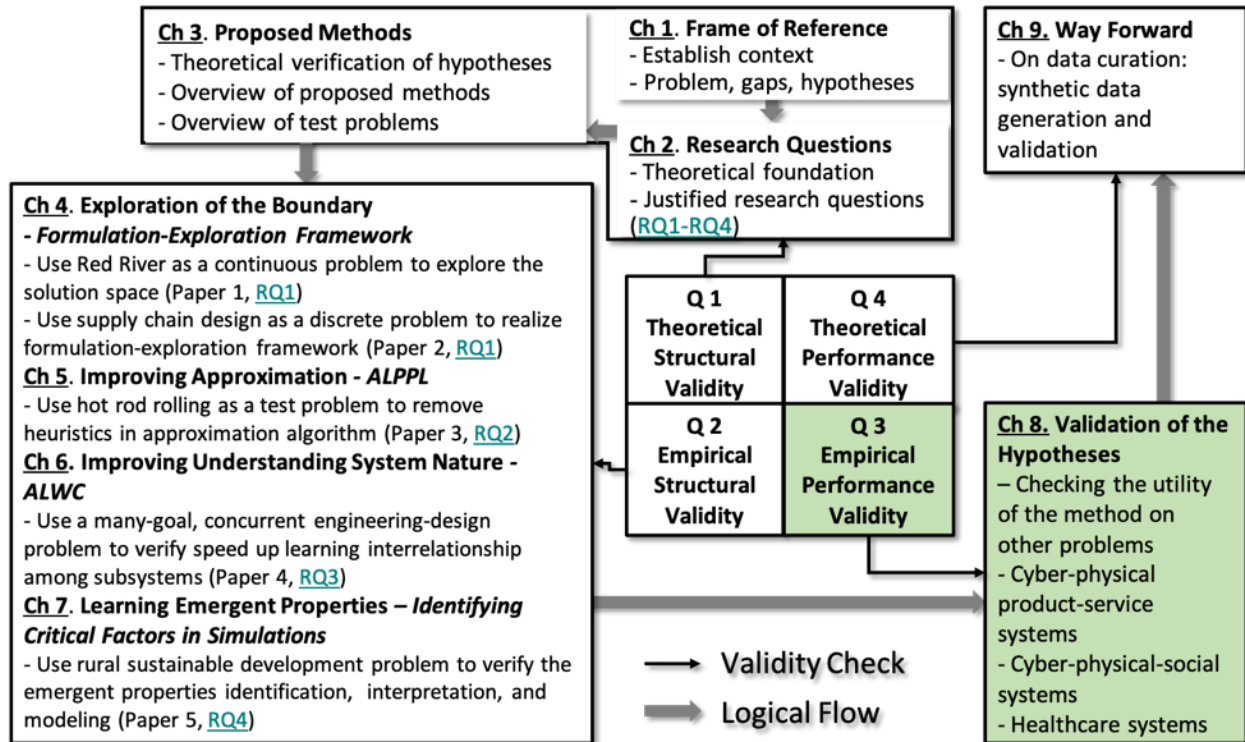


Figure 8. 2 Finishing Empirical Performance Validity in Chapter 8

## **CHAPTER 9 CLOSING REMARKS – ADVANCING MODEL EVOLUTION IN OTHER DISCIPLINES**

### ***– A VISION FOR FUTURE RESEARCH IN THE REALIZATION OF MODEL-BASED COMPLEX SYSTEMS IN THE FORM OF AN “I STATEMENT”***

The principal goal in this dissertation is to create decision-based approach that is suitable for the smart design in the realization of complex systems, managing emerging property and providing insight for future use. This provides an opportunity to design engineers to explore solution space without prior domain knowledge and analyze functionality of their design decisions. Principles for robust design in realization of complex systems are identified and articulated.

#### **9.1 Summary of This Dissertation**

##### ***9.1.1 Motivation of Model Evolution using Satisficing Strategy***

All models are approximations of the real world. Some solutions are sensitive to the incompleteness and inaccuracy of a decision model. The evolution of the model regarding the improvement in model formulation, approximation, exploration, and evaluation is important in obtaining the solution space that is relatively insensitive to the model inaccuracy. The concept of the “evolutionary model” comes from biological evolution, specifically indicating the models of DNA evolution. Inspired by this conception, in this dissertation, we expand “model evolution” to all model-based complex systems. In model evolution, two capabilities of a model are improved. 1) Model accuracy, as one of the foci of optimizing methods – the capability of a model to capture and incorporate more useful information of the physical world. 2) Model robustness, as one of the foci of *satisficing* methods – the capability of a model to deliver the solutions that are relatively insensitive to uncertainties.

### ***9.1.2 Contributions – Research Questions and Answers Leading to New Knowledge***

We pose four research questions (RQ1-4) and specified hypotheses (SH1-4) to create new knowledge and realize the model evolution.

**RQ1:** What is the method to evolve model boundary?

**SH1:** Explore the sensitivity of the segments of the model boundary and improve accordingly.

**New Knowledge – M1:** Formulation-Exploration Framework.

**RQ2:** What is the method to speed up learning the system nature?

**SH2:** Explore the sensitivity of the segments of the model boundary and improve accordingly.

**New Knowledge – M2:** the ALPPL (adaptive linear programming with parameter learning) algorithm.

**RQ3:** What is the method to speed up learning the system nature?

**SH3:** Learn system nature such as interrelationship among subsystems and reorganize them based on it.

**New Knowledge – M3:** the ALWC (adaptive leveling-weighting-clustering) algorithm.

**RQ4:** What is the method that allows passing the information through multiple scales of a system?

**SH4:** Capture and quantify emergent properties through scenario planning in simulation.

## New Knowledge – M4: Scenario planning workflow for Agent-Based Modeling

### *9.1.3 Verification and Validation*

**Method (theoretical structural validity):** we propose the four-stage design evolution loop – formulation-approximation-exploration-evaluation, to realize the model evolution. Through managing the complexity and uncertainty among different stages, the model of a complex system can be improved continuously through evolution.

**Test problems and experiments (empirical structural validity):** we use five test problems to illustrate the model evolution. A dam-network planning problem (T1.1) and a supply-chain-design problem (T1.2) for model evolution by identifying and removing bottlenecks through evolving the boundary of a continuous and a discrete model; a multi-stage manufacturing problem (T2 hot rod process chain) for model evolution by improving the approximation algorithm through parameter learning; a concurrent engineering-design problem (T3 Rankine cycle thermal system) for model evolution by reformulating the many-objective scalarization function; a social system design problem (T4 Learning emergent properties of a social system under interventions) for model evolution by capturing and incorporating emergent properties in the system algorithms.

**Results (empirical performance validity):** The proposed Formulation-Exploration framework facilitate designers to improve the robustness and achievement of system goals by evolving model boundary; the proposed ALPPL algorithm allows models to return solutions that are relatively robust to scenario changing; the proposed ALWC algorithm supports model evolution by improving the diversity of the solution space; the proposed multi-scale simulation framework helps decision-makers leverage critical factors to achieve system goals.

**Way Forward (theoretical performance validity):** The model evolution concept and the derived methods can be used to improve the model of other complex systems, especially when there are more than three system goals, the goals are conflicting with one another, the changing environment brings complexity and uncertainties to modeling, and the information sharing between components is not sufficient; for example, network planning for improving hospital visiting, leveraging social drivers in rural development, and knowledge management in designing cyber-physical product-service systems.

#### ***9.1.4 Relevant Publications***

##### ***Journal Publications***

Wang, R., Milisavljevic-Syed, J., **Guo, L.**, Huang, Y., Wang, G., 2021, “Knowledge-Based Design Guidance System for Cloud-Based Decision Support in the Design of Complex Engineered Systems,” *ASME Journal of Mechanical Design*, 143(7), 072001.

**Guo, L.**, Chen, S., Allen, J.K., Mistree, F., 2021, “A Framework for Designing the Customer Order Decoupling Point to Facilitate Mass Customization,” *ASME Journal of Mechanical Design*, 143(2): 022002.

**Guo, L.**, Mohebbi, S., Das, A., Allen, J.K., Mistree, F., 2020, “A Framework for the Exploration of Critical Factors on Promoting Two Season Cultivation in India,” *ASME Journal of Mechanical Design*, 142(12): 124503.

**Guo, L.**, Zamanisabzi, H., Neeson, T.M., Allen, J.K., Mistree, F., 2019, “Managing Conflicting Water Resource Goals and Uncertainties in a Dam-Network by Exploring the Solution Space,” *ASME Journal of Mechanical Design*, 141(3): 031702.

### ***Referred Conference Papers***

**Guo, L.**, Nellippallil, A.B., Smith, W.F., Allen, J.K., Mistree, F., 2020, “Adaptive Linear Programming Algorithm with Parameter Learning,” *ASME Design Automation Conference*, Online. Paper Number DETC2020-22602.

**Guo, L.**, Chen, S., Allen, J.K., Mistree, F., 2019, “Designing the Customer Order Decoupling Point to Facilitate Mass Customization,” *ASME Design Automation Conference*, Anaheim, CA, USA. Paper Number DETC2019-97379.

**Guo, L.**, Zamanisabzi, H., Neeson, T.M., Allen, J.K., Mistree, F., 2018, “Managing Conflicting Water Resource Goals and Uncertainties in a Dam-Network by Exploring the Solution Space,” *ASME Design Automation Conference*, Quebec City, Quebec, Canada. Paper Number DETC2018-86018.

### ***Manuscript Under Review***

**Guo, L.**, Nellippallil, A.B., Smith, W.F., Allen, J.K., Mistree, F., 2021, “A Smart Linear Programming Algorithm,” *ASME Journal of Mechanical Design*. Paper Number MD-21-1436, *under review*.

**Guo, L.**, Milisavljevic-Syed, J., Wang, R., Huang Y., Allen, J.K., Mistree, F., 2021 “Managing Many-Goal, Concurrent Design Problems using Adaptive Leveling-Weighting-Clustering Algorithm,” *Structural and Multidisciplinary Optimization*, *under review*.

### ***9.1.5 Closing Remarks of the Summary***

In this dissertation, we propose the concept of model evolution and derive corresponding methods and algorithms, regarding improving model accuracy and robustness. Through practicing the

proposed methods using five test problems, we demonstrate the internal consistency and general utility of the proposed methods. With the explosive development of Artificial Intelligence in the 21<sup>st</sup> century, the topic of model evolution will continue, and more beautiful theories and applications will be created.

## 9.2 Way Forward – “I Statement”

In this section, my research thrusts and applications in my early career in academia and their foundation (summarized in Table 9.1) are explained in detail.

**Table 9. 1 Research Thrusts and Application in My Early Career**

Research Thrust	Applications	Foundation
What is the mathematics that supports the directed evolution of the data curation methods and processes?	Designing lean process chains to support fail-safe healthcare networks and cyber-physical product-service systems	Chapter 4
How can we improve algorithms by replacing heuristics with insight and automate the process?	Realizing the customization of decision workflows for cyber-physical product-service systems (CPPSS).	Chapter 4 and 7
	Design fail-safe supply networks for healthcare systems	Chapter 4 and 6
What are the mechanisms and modeling strategies to support information sharing between multi-scale simulations?	Managing emergent properties of self-organizing systems	Chapter 7

### 9.2.1 Overarching Research Theme and Goals

In my academic career in academia, we plan to focus on research associated with the directed evolution of service systems, improving the robustness and resilience of multidisciplinary systems, knowledge management in data curation, and fail-safe network planning. We believe that the future of the realization of complex systems is on model evolution, including algorithm evolution, design automation, predictor-corrector mechanisms design, and new technologies incorporation.



### ***9.2.2 Research Thrusts and Applications***

Building on what has been done in my dissertation, in my early career (2021-2025), I plan to seek answers to key challenges anchored in four research thrusts (RT1-RT4). Each research thrust is an extension of my doctoral research.

**RT1** What is the mathematics that supports the directed evolution of the data curation methods and processes? Applications:

Designing lean process chains to support fail-safe healthcare networks and cyber-physical product-service systems. The first focus is on developing computational frameworks for ***synthetic data generation*** and data analytics. The second focus is developing algorithms to support the ***directed evolution of the metaheuristics*** used in modeling and approximations. The third focus is on improving the integration of decision models ***leveraging new technologies***, such as the Internet of Things, cloud computing, and deep learning, ***to support managing organizational complexities management***. This project is a way forward to Chapter 4.

**RT2** How can we improve algorithms by replacing heuristics with insight and automate the process? Applications:

Realizing the customization of decision workflows for cyber-physical product-service systems (CPPSS). The literature on CPPSS can be classified into three dimensions, managing complexity, uncertainty, and knowledge, using a ***design guidance framework, the Concept-Decision-Knowledge (CDK) framework***. I plan to go on with standardizing and customizing processes and workflows for a CPPSS to facilitate digitalization and automation of its process chains; see Figure 9.1. The rationale is to establish an iterative design loop in which knowledge is discovered through post-solution analysis and interpreted into new rules to update metaheuristics in the algorithms so

as to make the system adapt to the changing environment. This project will be a way forward to my Chapter 4 and Chapter 7.

Design fail-safe supply networks for healthcare systems. To make a healthcare network robust to 1) rare but high magnitude stochastic events – disruptions, and 2) frequent but low magnitude stochastic events – variations, we need to *manage the topology failure and flow variation simultaneously*. This project is based on my Chapter 4 and Chapter 6, and a book from Systems Realization Laboratory, *Architecting Fail-Safe Supply Networks* (<https://doi.org/10.1201/b22406>).

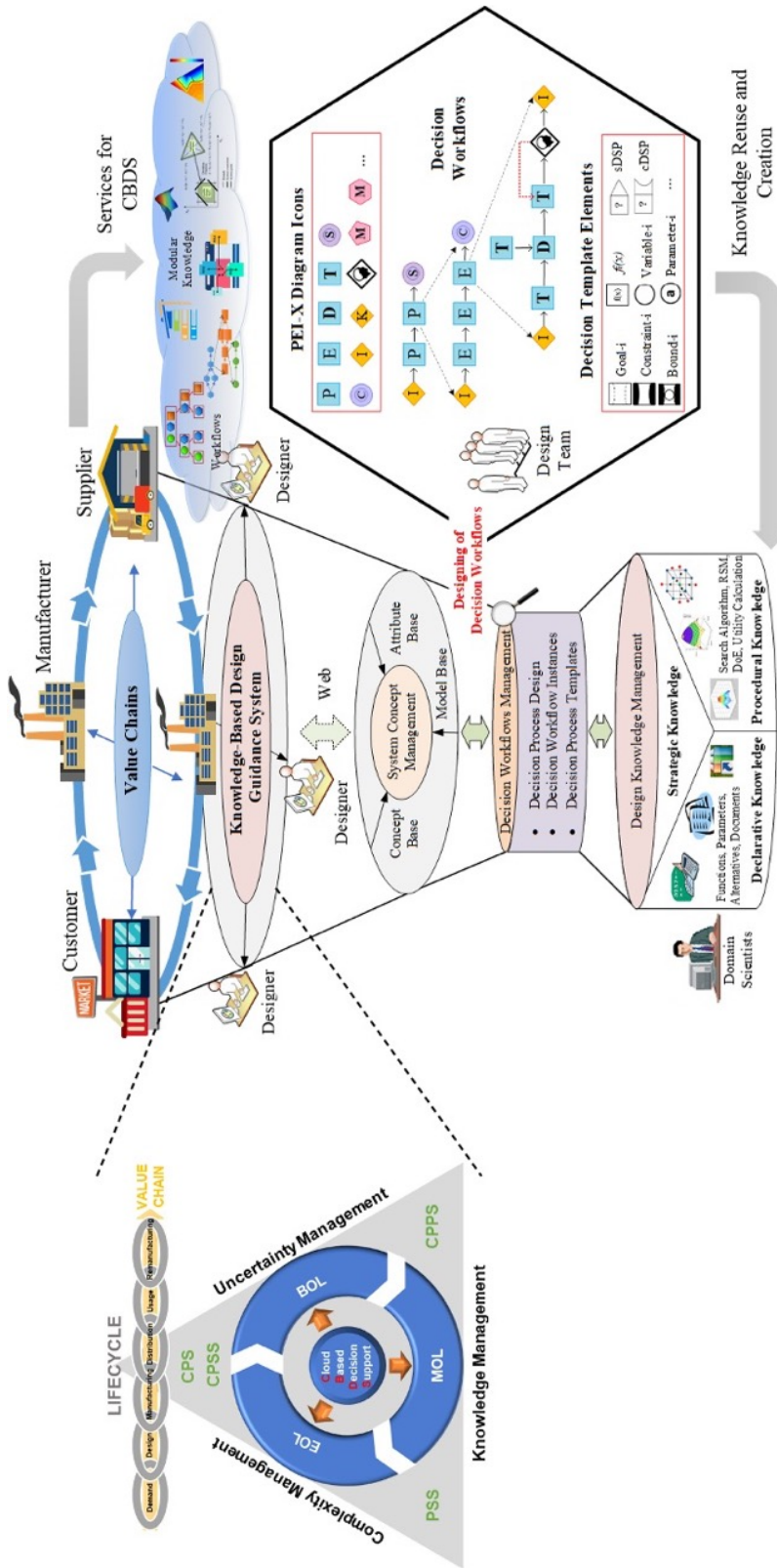


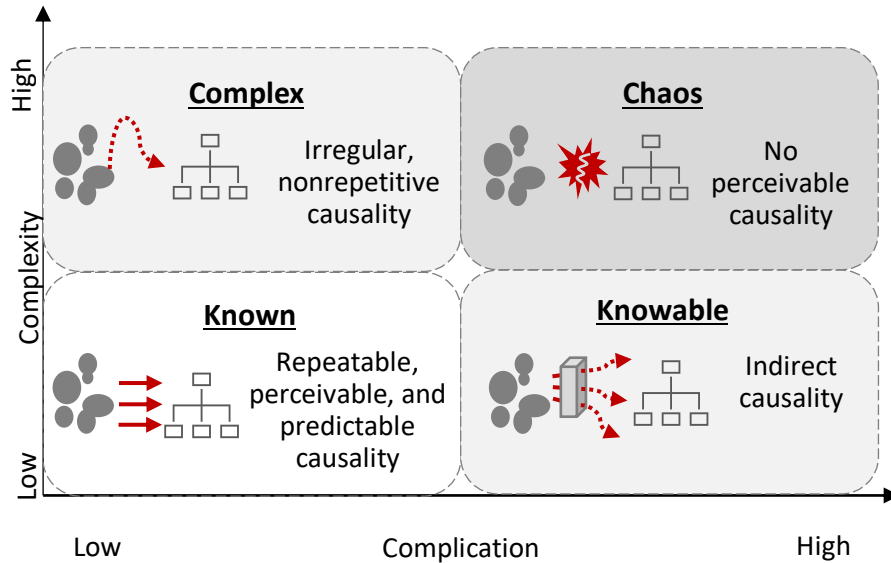
Figure 9. 1 A Knowledge-Based Design Guidance for CPPSS

**RT3** What are the mechanisms and modeling strategies to support information sharing between multi-scale simulations? This project will be an extension of my Chapter 7. Application:

Managing emergent properties of self-organizing systems. *A generic modeling strategy* and *multi-scale simulation workflow* will be developed to make and update rules and mechanisms to inspire individuals to self-organize into a globally efficient state. My objective is to manage chaos, the problem with high complexity and high complication; see Figure 9.2. For a complex system with chaos, there is no perceivable causality among elements. We assume that there are no mathematical relations among different scales can be captured and used for decision making. However, there can be factors more critical than others that can be leveraged to force the system to evolve towards a direction that we desire, which makes the system evolve to a complex system with high complexity but low complication, or to a knowable system with high complication but low complexity. I hypothesize that through the use of digital thread and digital twins, we can learn the critical factors and intervene the system through reinforcing learning. Therefore, the information can be shared between multiple scales of the system, and it can be self-organized and serve the system goals.

**RT4** When dealing with multicriteria decision support problems, what are the mechanisms to compare and/or integrate various solution algorithms and design strategies to provide decision support? There can be applications in several different fields. One of them is:

Design and managing lean process chains using digital twins. In process chains, we manage stakeholders' goals in multiple stages, at multiple levels, based on multiple criteria. We plan to use the *digital thread* and *digital twin to facilitate design automation, data analytics, and knowledge discovery*. This project will be an extension of my Chapter 6.



**Figure 9. 2 Managing Complex Systems with Different Types of Causality**

### ***9.2.3 Potential Cross-Disciplinary Research Opportunities***

Potential Cross-Disciplinary Research Opportunities include but are not limited to

- i) ***Knowledge awareness in data curation for managing organizational complexity of healthcare networks and cyber-physical product-service systems (CPPSS) (RT1)***. As researchers face the challenge in the accessibility of authentic data in multidisciplinary systems, synthetic data generation is a way to fill the gap. There are two types of data curation and validation, the data-driven methods and the process-driven methods. In this project, there can be potential contributions to both types of methods – dealing with sparse data in data-driven methods and the ontology-based multi-scale simulations and cross-validation framework in process-driven methods.
- ii) ***Lean process design leveraging new technologies (RT4)***. To help a rural community in sustainable development, a social entrepreneur needs to collaborate with other stakeholders in the community to make and achieve environmental, social, and economic goals. Optimizing the process chains for the community is a way to

accomplish their collective goals and individual goals at different times and spaces. In this project, I plan to fill three research gaps: 1) managing the evolving priority of the multiple goals using adaptive objective-scalarization algorithm using unsupervised learning, 2) realizing the directed evolution to “stay Lean,” and 3) using the digital thread and digital twin to facilitate design automation and knowledge management.

- iii) ***Realizing satisficing strategy based on adaptive approximation and Dual Simplex (RT2)***. Solution algorithms fall into two categories, formulating a problem exactly and then solving it approximately, and approximating a problem and then solving it exactly. The algorithms in the latter category are easy to apply the *satisficing* strategy and return good enough solutions for continuous improvement. When designing multidisciplinary systems, due to their usual features, such as multi-stage, hierarchical structure, nonlinearity, non-convexity, and discreteness, to identify feasible space and learn the features of each subspace, designers often need to approximate and partition the problem, exploring the dual, and exploring the solution space to obtain more information. The research gap I will fill is to realize the *satisficing* strategy for designing complex systems taking into account multiple uncertainties.
- iv) ***Mechanism design and policymaking for multidisciplinary systems based on critical and sensitive factors (RT3)***. In fast-growing economies with large populations, such as China and India, soil erosion and water pollution caused by overexploitation of natural resources have become hidden dangers that affect their long-term economic vitality. The rapid changing and overcorrecting policies make things worse, which affects the sustainability of their social-ecological system. In this project, I plan to fill the gap in providing knowledge-based decision support on policymaking based on

mapping the social costs that will occur in the future to the current social-ecological system stakeholders based on game theory.

- v) ***Managing complexity and uncertainty in service systems using ontology-based design guidance platforms (RT2).*** We divided complexities in multicriteria systems into four types – system complexity, design complexity, process complexity, and organizational complexity, and we divided uncertainties into four types, uncertainty in parameters, variables, model structure, and process chains. This project’s potential contributions include developing an ontology-based design guidance platform to categorize and quantify complexities and uncertainties so as to enhance the system performance and reflect human roles at the decision level.
- vi) ***Assessing the impact of public events or crises on engineering education based on social media data.*** For example, students’ evaluation of courses in engineering before and after the pandemic based on data from social media and website, such as “ratemyprofessors.com.” Three contributions can be made: 1) developing a computational framework to assess the impact of major crises on engineering education, especially online education, 2) evaluating the role and significance of today’s social media and rating websites in assessing lecturing quality and improving programs, and 3) leveraging technologies to improve the efficiency and students’ experience of engineering lecturing, tutorials, experiments, and the whole process of learning.

#### ***9.2.4 Closing Remarks of the Way Forward***

I envision carrying the model evolution loop on, enriching it with methods incorporating new technologies in the age of Industry 4.0, applying it to more disciplines, and developing a

sustainable research program with research interests in operations management, lean process design, and design automation of service systems and other multidisciplinary systems.

### **9.3 Role of Chapter 9 in this Dissertation**

In this chapter, we summarize the motivation, contributions, and verification and validation in Section 9.1; the way forward is given in Section 9.2, including the overarching research theme and goals in my early career in academia, my research thrusts and applications, and potential cross-disciplinary research opportunities.

In summary, in Chapter 9, we provide the theoretical performance validity; see Figure 9.3. The theoretical performance validity is done by anticipating what future theoretical breakthroughs can be made based on the model evolution loop using satisficing strategy and what applications can be done to make the world a better place.



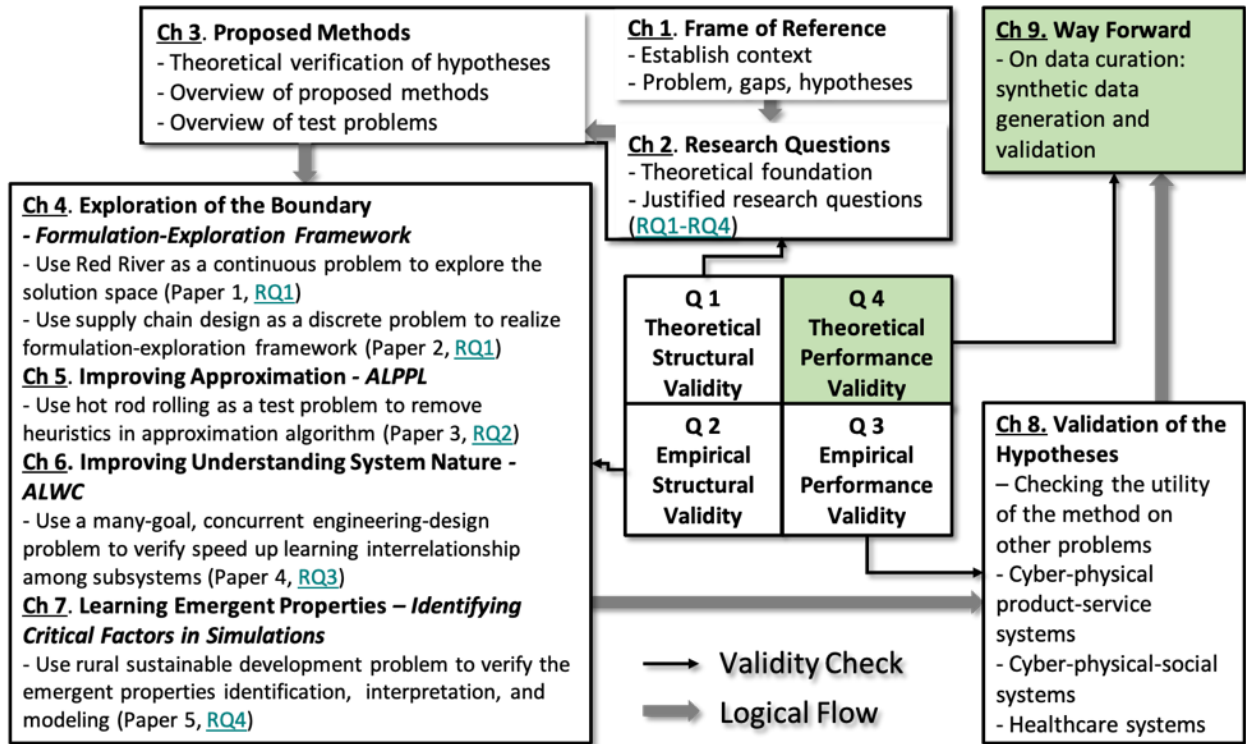


Figure 9. 3 Finishing Theoretical Performance Validity in Chapter 9

## REFERENCES

- Abreu, C. G. and C. G. Ralha (2018). "An Empirical Workflow to Integrate Uncertainty and Sensitivity Analysis to Evaluate Agent-Based Simulation Outputs." Environmental Modelling & Software **107**: 281-297.
- Afshari, H. and Q. Peng (2015). "Modeling and Quantifying Uncertainty in the Product Design Phase for Effects of User Preference Changes." Industrial Management & Data Systems **115**(9): 1637-1665.
- Ahmadi, M. and E. Teimouri (2008). "Determining the Order Penetration Point in Auto Export Supply Chain by the Use of Dynamic Programming." Journal of Applied Sciences **8**(18): 3214-3220.
- Ahmed, S., C.-H. Goh, J. K. Allen, F. Mistree, P. Zagade and B. Gautham (2014). Hot Forging of Automobile Steel Gear Blanks: An Exploration of the Solution Space. ASME Design Automation Conference, Buffalo, New York, USA, American Society of Mechanical Engineers.
- Aitken, J., P. Childerhouse and D. Towill (2003). "The impact of product life cycle on supply chain strategy." International Journal of Production Economics **85**(2): 127-140.
- Al Irsyad, M. I., A. Halog and R. Nepal (2019). "Estimating the Impacts of Financing Support Policies towards Photovoltaic Market in Indonesia: A Social-Energy-Economy-Environment Model Simulation." Journal of environmental management **230**: 464-473.
- Allen, J. K., C. Seepersad, H. Choi and F. Mistree (2006). "Robust design for multiscale and multidisciplinary applications."
- Bader, J. and E. Zitzler (2011). "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization." Evolutionary computation **19**(1): 45-76.
- Bandaru, S., A. H. Ng and K. Deb (2017). "Data Mining Methods for Knowledge Discovery in Multi-Objective Optimization: Part A-Survey." Expert Systems with Applications **70**: 139-159.
- Barros, M. T., F. T. Tsai, S.-I. Yang, J. E. Lopes and W. W. Yeh (2003). "Optimization of large-scale hydropower system operations." Journal of Water Resources Planning and Management **129**(3): 178-188.
- Barthelemy, J.-F. and R. T. Haftka (1993). "Approximation Concepts for Optimum Structural Design—A Review." Structural optimization **5**(3): 129-144.
- Bellman, R. (1957). "Dynamic Programming (DP)."
- Bellman, R. and S. Dreyfus (1962). "Computational aspects of dynamic programming." Princeton, New Jersey.
- Box, G. E. (1976). "Science and Statistics." Journal of the American Statistical Association **71**(356): 791-799.
- Box, G. E. (1979). "All models are wrong, but some are useful." Robustness in Statistics **202**: 549.
- Box, G. E. and N. R. Draper (1987). Empirical model-building and response surfaces, Wiley New York.
- Byron, M. (1998). "Satisficing and Optimality." Ethics **109**(1): 67-93.

- Changchit, C. and M. Terrell (1993). "A multiobjective reservoir operation model with stochastic inflows." Computers & industrial engineering **24**(2): 303-313.
- Chen, W., J. K. Allen and F. Mistree (1997). "A Robust Concept Exploration Method for Enhancing Productivity in Concurrent Systems Design." Concurrent Engineering **5**(3): 203-217.
- Chen, W., J. K. Allen, K.-L. Tsui and F. Mistree (1996). "A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors." Journal of mechanical design **118**(4): 478-485.
- Chen, W., K.-L. Tsui, J. K. Allen and F. Mistree (1994). "Integration of the response surface methodology with the compromise decision support problem in developing a general robust design procedure."
- Chitungo, S. K. and S. Munongo (2013). "Extending the Technology Acceptance Model to Mobile Banking Adoption in Rural Zimbabwe." Journal of Business Administration and Education **3**(1): 51-79.
- Choi, H., D. L. McDowell, J. K. Allen, D. Rosen and F. Mistree (2008). "An Inductive Design Exploration Method for Robust Multiscale Materials Design." Journal of mechanical design **130**(3): 031402.
- Choi, H.-J., R. Austin, J. K. Allen, D. L. McDowell, F. Mistree and D. J. Benson (2005). "An approach for robust design of reactive power metal mixtures based on non-deterministic micro-scale shock simulation." Journal of Computer-Aided Materials Design **12**(1): 57-85.
- Choi, H.-J., R. Austin, J. Shepherd, J. K. Allen, D. McDowell, F. Mistree and D. Benson (2004). An Approach for Robust Micro-Scale Materials Design under Unparameterizable Variability. 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.
- Choi, H.-J., D. L. Mcdowell, J. K. Allen and F. Mistree (2008). "An Inductive Design Exploration Method for Hierarchical Systems Design under Uncertainty." Engineering Optimization **40**(4): 287-307.
- Clayton, E. R., W. E. Weber and B. W. Taylor III (1982). "A goal programming approach to the optimization of multi response simulation models." Iie Transactions **14**(4): 282-287.
- Crawley, P. and G. Dandy (1993). A Headworks Optimisation Model Incorporating a Graphical User Interface. Watercomp 93: 2nd Australasian Conference on Computing for the Water Industry Today and Tomorrow; Preprints of Papers, Institution of Engineers, Australia.
- Daaboul, J., C. Da Cunha, J. Le Duigou, B. Novak and A. Bernard (2015). "Differentiation and Customer Decoupling Points: An Integrated Design Approach for Mass Customization." Concurrent Engineering **23**(4): 284-295.
- Daaboul, J., F. Laroche and A. Bernard (2010). Determining the CODP Position by Value Network Modeling and Simulation. Technology Management Conference (ICE), 2010 IEEE International, IEEE.
- Dahe, P. and D. Srivastava (2002). "Multireservoir multiyield model with allowable deficit in annual yield." Journal of Water Resources Planning and Management **128**(6): 406-414.
- Deb, K., A. Pratap, S. Agarwal and T. Meyarivan (2002). "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." IEEE transactions on evolutionary computation **6**(2): 182-197.
- Eschenbach, E. A., T. Magee, E. Zagona, M. Goranflo and R. Shane (2001). "Goal programming decision support system for multiobjective operation of reservoir systems." Journal of Water Resources Planning and Management **127**(2): 108-120.

- Fok, K. L. and A. K. Chopra (1986). "Earthquake analysis of arch dams including dam–water interaction, reservoir boundary absorption and foundation flexibility." Earthquake engineering & structural dynamics **14**(2): 155-184.
- Ford, L. and D. R. Fulkerson (1962). "Flows in networks. 1962." Princeton U. Press, Princeton, NJ.
- Gao, Y., F.-Y. Wang, Z.-Q. Zhao and Z.-Q. Xiao (2012). Flexible Manipulators: Modeling, Analysis and Optimum Design, Academic Press.
- Ghalehkhondabi, I., E. Ardjmand and G. Weckman (2017). "Integrated Decision Making Model for Pricing and Locating the Customer Order Decoupling Point of A Newsvendor Supply Chain." Opsearch **54**(2): 417-439.
- Ghalehkhondabi, I., D. Sormaz and G. Weckman (2016). "Multiple Customer Order Decoupling Points within a Hybrid MTS/MTO Manufacturing Supply Chain with Uncertain Demands in Two Consecutive Echelons." Opsearch **53**(4): 976-997.
- Guéret, C., C. Prins and M. Sevaux (1999). "Applications of Optimization with Xpress-MP." contract: 00034.
- Guo, L., S. Chen, J. K. Allen and F. Mistree (2019). Designing the Customer Order Decoupling Point to Facilitate Mass Customization. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.
- Guo, L., H. Zamanisabzi, T. M. Neeson, J. K. Allen and F. Mistree (2019). "Managing Conflicting Water Resource Goals and Uncertainties in a Dam Network by Exploring the Solution Space." Journal of Mechanical Design **141**(3): 031702.
- Hajfathaliha, A., E. Teimoury, I. G. Khondabi and M. Fathi (2011). "Using queuing approach for locating the order penetration point in a two-echelon supply chain with customer loss." International Journal of Business and Management **6**(1): 258.
- Hao, F. and J.-P. Merlet (2005). "Multi-criteria optimal design of parallel manipulators based on interval analysis." Mechanism and Machine Theory **40**(2): 157-171.
- Hsu, N.-S. and K.-W. Cheng (2002). "Network flow optimization model for basin-scale water supply planning." Journal of Water Resources Planning and Management **128**(2): 102-112.
- Ignizio, J. P. (1976). "An Approach to the Capital Budgeting Problem with Multiple Objectives." The Engineering Economist **21**(4): 259-272.
- Ignizio, J. P. (1982). Linear Programming in Single and Multi-Objective Systems. EnglewoodCliffs, New Jersey, Prentice Hall.
- Iravani, S. M., T. Liu and D. Simchi-Levi (2012). "Optimal Production and Admission Policies in Make-To-Stock/Make-To-Order Manufacturing Systems." Production and Operations Management **21**(2): 224-235.
- Isukapalli, S., A. Roy and P. Georgopoulos (1998). "Stochastic Response Surface Methods (SRSMs) for Uncertainty Propagation: Application to Environmental and Biological Systems." Risk analysis **18**(3): 351-363.
- Jacobs, F. R., R. B. Chase and N. Aquilano (2004). "Operations Management for Competitive Advantage." Boston: Mc-Graw Hill **64**: 70.

- Jaimes, A. L., C. A. C. Coello and J. E. U. Barrientos (2009). Online Objective Reduction to Deal with Many-Objective Problems. International Conference on Evolutionary Multi-Criterion Optimization, Springer.
- Jeong, I.-J. (2011). "A Dynamic Model for the Optimization of Decoupling Point and Production Planning in a Supply Chain." International Journal of production economics **131**(2): 561-567.
- Jiang, R., J. Wang and Y. Guan (2012). "Robust Unit Commitment with Wind Power and Pumped Storage Hydro." IEEE Transactions on power systems **27**(2): 800.
- Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. University of Chicago, Master's thesis.
- Kortanek, K. and W. Maxwell (1969). "On a New Class of Combinatoric Optimizers for Multi-Product Single-Machine Scheduling." Management Science **15**(5): 239-248.
- Kuczera, G. and G. Diment (1988). "General water supply system simulation model: WASP." Journal of Water Resources Planning and Management **114**(4): 365-382.
- Kuhn, H. W. and A. Tucker (1951). W., 1951, "Nonlinear Programming,". Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability (University of California Press, Berkeley, CA).
- Ladyman, J., J. Lambert and K. Wiesner (2013). "What is a complex system?" European Journal for Philosophy of Science **3**(1): 33-67.
- Lemmon, E. W. and M. L. Huber (2013). Implementation of Pure Fluid and Natural Gas Standards: Reference Fluid Thermodynamic and Transport Properties Database (REFPROP), National Institute of Standards and Technology, NIST.
- LindoOnlineHelp "Dual Price."
- LindoOnlineHelp. "Slack or Surplus." from [https://www.lindo.com/doc/online\\_help/lingo17\\_0/slack\\_or\\_surplus.htm](https://www.lindo.com/doc/online_help/lingo17_0/slack_or_surplus.htm).
- Liu, W., Y. Mo, Y. Yang and Z. Ye (2015). "Decision Model of Customer Order Decoupling Point on Multiple Customer Demands in Logistics Service Supply Chain." Production Planning and Control **26**(3): 178-202.
- Liu, W., R. Wu, Z. Liang and D. Zhu (2018). "Decision Model for the Customer Order Decoupling Point Considering Order Insertion Scheduling with Capacity and Time Constraints in Logistics Service Supply Chain." Applied mathematical modelling **54**: 112-135.
- Loganathan, G. and D. Bhattacharya (1990). "Goal-programming techniques for optimal reservoir operations." Journal of Water Resources Planning and Management **116**(6): 820-838.
- Loucks, D. P. (2000). "Sustainable water resources management." Water international **25**(1): 3-10.
- Loucks, D. P., J. R. Stedinger and D. A. Haith (1981). Water resource systems planning and analysis, Prentice-Hall.
- Lumbroso, D. and M. Davison (2018). "Use of an Agent-Based Model and Monte Carlo Analysis to Estimate the Effectiveness of Emergency Management Interventions to Reduce Loss of Life during Extreme Floods." Journal of Flood Risk Management **11**: S419-S433.

- Macquorn Rankine, W. J. (1853). "XVIII. On the General Law of the Transformation of Energy." The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **5**(30): 106-117.
- McPherson, R., et al. (2016). "Impacts of climate change on flows in the Red River Basin." Final report to the South Central Climate Science Center.
- Messac, A. and C. A. Mattson (2002). "Generating Well-Distributed Sets of Pareto Points for Engineering Design Using Physical Programming." Optimization and Engineering **3**(4): 431-450.
- Ming, Z., A. B. Nellippallil, Y. Yan, G. Wang, C. H. Goh, J. K. Allen and F. Mistree (2018). "PDSIDES—A Knowledge-Based Platform for Decision Support in the Design of Engineering Systems." Journal of Computing and Information Science in Engineering **18**(4): 041001.
- Mistree, F. (1993). "Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm." Structural Optimization: Status and Promise: 40.
- Mistree, F., O. Hughes and H. Phuoc (1981). "An optimization method for the design of large, highly constrained complex systems." Engineering Optimization **5**(3): 179-197.
- Mistree, F., O. F. Hughes and B. Bras (1993). "Compromise decision support problem and the adaptive linear programming algorithm." Progress in Astronautics and Aeronautics **150**: 251-251.
- Mistree, F., O. F. Hughes and B. Bras (1993). "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm." Structural Optimization: Status and Promise, MP Kamat, ed., AIAA, Washington, DC.
- Mistree, F., O. F. Hughes and H. B. Phuoc (1981). "An Optimization Method for the Design of Large, Highly Constrained Complex Systems." Engineering Optimization **5**(3): 179-197.
- Mistree, F. and S. Kamal (1985). DSIDES: Decision Support in the Design of Engineering Systems, University of Houston.
- Mistree, F., D. Muster, S. Srinivasan and S. Mudali (1990). "Design of Linkages: A Conceptual Exercise in Designing for Concept." Mechanism and Machine Theory **25**(3): 273-286.
- Mistree, F., B. Patel and S. Vadde (1994). "On Modeling Multiple Objectives and Multi-Level Decisions in Concurrent Design." Advances in Design Automation **69**(2): 151-161.
- Mousavi, S. J., K. S. Moghaddam and A. Seifi (2004). "Application of an interior-point algorithm for optimization of a large-scale reservoir system." Water resources management **18**(6): 519-540.
- Nandalal, K. and J. J. Bogardi (2007). Dynamic programming based operation of reservoirs: applicability and limits, Cambridge university press.
- Naylor, J. B., M. M. Naim and D. Berry (1999). "Leagility: Integrating the Lean and Agile Manufacturing Paradigms in the Total Supply Chain." International Journal of production economics **62**(1-2): 107-118.
- Needham, J. T., D. W. Watkins Jr, J. R. Lund and S. Nanda (2000). "Linear programming for flood control in the Iowa and Des Moines rivers." Journal of Water Resources Planning and Management **126**(3): 118-127.
- Neely, W. P., J. Sellers and R. M. North (1980). "Goal programming priority sensitivity analysis: an application in natural resource decision making processes." Interfaces **10**(5): 83-89.

Nellippallil, A. B., Mohan, P., Allen, J.K., Mistree, F. (2018). Robust Concept Exploration of Materials, Products and Associated Manufacturing Processes. ASME Design Automation Conference, . Quebec City, Canada: V02BT03A010-V002BT003A010.

Nellippallil, A. B., V. Rangaraj, B. Gautham, A. K. Singh, J. K. Allen and F. Mistree (2018). "An Inverse, Decision-Based Design Method for Integrated Design Exploration of Materials, Products, and Manufacturing Processes." Journal of mechanical design **140**(11): 111403.

Nellippallil, A. B., K. N. Song, C.-H. Goh, P. Zagade, B. Gautham, J. K. Allen and F. Mistree (2017). "A Goal-Oriented, Sequential, Inverse Design Method for the Horizontal Integration of a Multistage Hot Rod Rolling System." Journal of mechanical design **139**(3): 031403.

Norman, G. (1990). Production and Operation Management: A Problem–Solving and Decision–Making Approach, USA.: Dryden Press.

Opiyo, N. (2019). "Impacts of Neighbourhood Influence on Social Acceptance of Small Solar Home Systems in Rural Western Kenya." Energy Research and Social Science **52**(C): 91-98.

Parker, C. J. (2016). "Human Acceptance of 3D Printing in Fashion Paradox: Is Mass Customization A Bridge Too Far?" WIT Transactions on Engineering Sciences **113**: 373-380.

Parker, D. C., S. M. Manson, M. A. Janssen, M. J. Hoffmann and P. Deadman (2003). "Multi-Agent Systems for the Simulation of Land-Use and Land-Cover Change: A review." Annals of the association of American Geographers **93**(2): 314-337.

Parra, M. A., A. B. Terol and M. R. Uria (2001). "A Fuzzy Goal Programming Approach to Portfolio Selection." European Journal of Operational Research **133**(2): 287-297.

Poff, N. L., C. M. Brown, T. E. Grantham, J. H. Matthews, M. A. Palmer, C. M. Spence, R. L. Wilby, M. Haasnoot, G. F. Mendoza and K. C. Dominique (2016). "Sustainable water management under future uncertainty with eco-engineering decision scaling." Nature Climate Change **6**(1): 25.

Ponnambalam, K., A. Vannelli and T. Unny (1989). "An application of Karmarkar's interior-point linear programming algorithm for multi-reservoir operations optimization." Stochastic Hydrology and Hydraulics **3**(1): 17-29.

Qiu, R., W. Xu, J. Zhang and K. Staenz (2018). "Modelling and Simulating Urban Residential Land Development in Jiading New City, Shanghai." Applied Spatial Analysis and Policy **11**(4): 753-777.

Rae, A. N. (1972). A Note on the Solution of Goal Programming Problems with Pre-emptive Priority, Market Research Centre, Massey University.

Rani, D. and M. M. Moreira (2010). "Simulation–optimization modeling: a survey and potential application in reservoir systems operation." Water resources management **24**(6): 1107-1138.

Reddy, M. J. and D. N. Kumar (2007). "Multiobjective differential evolution with application to reservoir system optimization." Journal of Computing in Civil Engineering **21**(2): 136-146.

Reddy, R., Smith, W.F., Mistree, F., Bras, B.A., Chen, W., Malhotra, A., Badhrinath, K., Lautenschlager, U., Pakala, R., Vadde, S., and Patel, P. (1992). DSIDES User Manual. U. o. H. Department of Mechanical Engineering. Houston, Texas, Systems Design Laboratory.

- Reynoso-Meza, G., X. Blasco, J. Sanchis and J. M. Herrero (2013). "Comparison of Design Concepts in Multi-Criteria Decision-Making Using Level Diagrams." Information Sciences **221**: 124-141.
- Rios, L. M. and N. V. Sahinidis (2013). "Derivative-Free Optimization: A Review of Algorithms and Comparison of Software Implementations." Journal of Global Optimization **56**(3): 1247-1293.
- Rossoshanskaya, E. A. (2019). "Integrated Agent-Based Model of Labor Potential Reproduction of a Municipal Formation." Economic and Social Changes: Facts, Trends, Forecast **12**(1): 124-137.
- Rudberg, M. and J. Wikner (2004). "Mass Customization in terms of the Customer Order Decoupling Point." Production Planning and Control **15**(4): 445-458.
- Sabeghi, M., R. Shukla, J. K. Allen and F. Mistree (2016). Solution Space Exploration of the Process Design for Continuous Casting of Steel. ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.
- Sabeghi, M., W. Smith, J. K. Allen and F. Mistree (2015). Solution Space Exploration in Model-Based Realization of Engineered Systems. ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.
- Salvendy, G. (2001). Handbook of Industrial Engineering: Technology and Operations Management, John Wiley and Sons.
- Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 1985, Carnegie-Mellon University, Pittsburgh, PA, Lawrence Erlbaum Associates. Inc., Publishers.
- Seada, H. and K. Deb (2014). "U-NSGA-III: A Unified Evolutionary Algorithm for Single, Multiple, and Many-Objective Optimization." COIN report **2014022**.
- Seifi, A. and K. W. Hipel (2001). "Interior-point method for reservoir operation with stochastic inflows." Journal of Water Resources Planning and Management **127**(1): 48-57.
- Shidpour, H., C. Da Cunha and A. Bernard (2014). "Analyzing Single and Multiple Customer Order Decoupling Point Positioning based on Customer Value: A Multi-Objective Approach." Procedia Cirp **17**: 669-674.
- Simon, H. A. (1956). "Rational choice and the structure of the environment." Psychological review **63**(2): 129.
- Simon, H. A. (1996). The Sciences of the Artificial, MIT Press.
- Simon, H. A. and J. B. Kadane (1975). "Optimal Problem-Solving Search: All-or-None Solutions." Artificial Intelligence **6**(3): 235-247.
- Smith, J. B. (2003). "Complex systems." arXiv preprint cs/0303020.
- Smith, W. F., S. Kamal and F. Mistree (1987). "The Influence of Hierarchical Decisions on Ship Design." Marine technology **24**(2): 131-142.
- Smith, W. F., J. Milisavljevic, M. Sabeghi, J. K. Allen and F. Mistree (2014). Accounting for Uncertainty and Complexity in the Realization of Engineered Systems. CSDM (Posters), Citeseer.



- Smith, W. F., J. Milisavljevic, M. Sabeghi, J. K. Allen and F. Mistree (2015). The realization of engineered systems with considerations of complexity. ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.
- Soltani, A. R., H. Tawfik, J. Y. Goulermas and T. Fernando (2002). "Path Planning in Construction Sites: Performance Evaluation of the Dijkstra, A\*, and GA search Algorithms." Advanced engineering informatics **16**(4): 291-303.
- Sreenivasan, K. and S. Vedula (1996). "Reservoir operation for hydropower optimization: a chance-constrained approach." Sadhana **21**(4): 503-510.
- Stewart, F. (2016). Technology and Underdevelopment, Springer.
- Taguchi, G. and D. Clausing (1990). "Robust Quality." Harvard business review **68**(1): 65-75.
- Taguchi, G., Plains W. (1993). "Taguchi on Robust Technology Development: Bringing Quality Engineering Upstream (ASME Press Series on International Advances in Design Productivity)." **191**: 1-191.
- Talvitie, A. (1981). Inaccurate or Incomplete Data as a Source of Uncertainty in Econometric or Attitudinal Models of Travel Behavior. In: New Horizons in Travel-Behavior Research. The Fourth International Conference on Behavioral Travel Modeling, Grainau, Bavaria, Germany.
- Tang, D., R. Zhu, J. Tang, R. Xu and R. He (2010). "Product Design Knowledge Management based on Design Structure Matrix." Advanced engineering informatics **24**(2): 159-166.
- Teegavarapu, R. S. and S. P. Simonovic (2000). "Short-term operation model for coupled hydropower reservoirs." Journal of Water Resources Planning and Management **126**(2): 98-106.
- Teimoury, E. and M. Fathi (2013). "An integrated operations-marketing perspective for making decisions about order penetration point in multi-product supply chain: a queuing approach." International Journal of Production Research **51**(18): 5576-5596.
- Teimoury, E., M. Modarres, I. Khondabi and M. Fathi (2012). "A Queuing Approach for Making Decisions about Order Penetration Point in Multiechelon Supply Chains." The International Journal of Advanced Manufacturing Technology **63**(1-4): 359-371.
- Triantaphyllou, E. and A. Sánchez (1997). "A sensitivity analysis approach for some deterministic multi-criteria decision-making methods." Decision Sciences **28**(1): 151-194.
- van Ackooij, W., R. Henrion, A. Möller and R. Zorgati (2014). "Joint chance constrained programming for hydro reservoir management." Optimization and Engineering **15**(2): 509-531.
- Vedula, S., P. Mujumdar and G. C. Sekhar (2005). "Conjunctive use modeling for multicrop irrigation." Agricultural Water Management **73**(3): 193-221.
- Velasquez, J., M. Khakifirooz and M. Fathi (2019). Large Scale Optimization Applied to Supply Chain & Smart Manufacturing: Theory & Applications, Springer Optimization and Its Applications.
- Vevea, J. L. and C. M. Woods (2005). "Publication Bias in Research Synthesis: Sensitivity Analysis Using a Priori Weight Functions." Psychological methods **10**(4): 428.

- Viswanathan, J. and I. E. Grossmann (1990). "A Combined Penalty Function and Outer-Approximation Method for MINLP Optimization." Computers and Chemical Engineering **14**(7): 769-782.
- Wang, F.-Y. (1994). "On the Extremal Fundamental Frequencies of One-Link Flexible Manipulators." The International journal of robotics research **13**(2): 162-170.
- Wang, R., A. B. Nellippallil, G. Wang, Y. Yan, J. K. Allen and F. Mistree (2018). "Systematic Design Space Exploration Using a Template-Based Ontological Method." Advanced engineering informatics **36**: 163-177.
- Wikipedia (2019). "Rankine Cycle." from [https://en.wikipedia.org/wiki/Rankine\\_cycle](https://en.wikipedia.org/wiki/Rankine_cycle).
- Wikner, J., M. M. Naim, V. L. Spiegler and J. Lin (2017). "IOBPCS based Models and Decoupling Thinking." International Journal of production economics **194**: 153-166.
- Wilhelm, A. G. (2000). Democracy in the Digital Age: Challenges to Political Life in Cyberspace, Psychology Press.
- Williams, R. J. and D. Zipser (1995). "Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity." Backpropagation: Theory, architectures, and applications **1**: 433-486.
- Wurbs, R. A. (1991). Optimization of multiple-purpose reservoir systems operations: A review of modeling and analysis approaches, HYDROLOGIC ENGINEERING CENTER DAVIS CA.
- Xu, X. and Z. Liang (2011). CODP Positioning Based on Extension Superiority Evaluation model. 2011 International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), IEEE.
- Xue, X., K. Zhang, Y. Hong, J. J. Gourley, W. Kellogg, R. A. McPherson, Z. Wan and B. N. Austin (2015). "New multisite cascading calibration approach for hydrological models: Case study in the red river basin using the VIC model." Journal of Hydrologic Engineering **21**(2): 05015019.
- Yakowitz, S. (1982). "Dynamic programming applications in water resources." Water Resources Research **18**(4): 673-696.
- Zarfl, C., A. E. Lumsdon, J. Berlekamp, L. Tydecks and K. Tockner (2015). "A global boom in hydropower dam construction." Aquatic Sciences **77**(1): 161-170.
- Zhang, Q. and H. Li (2007). "MOEA/D: A Multiobjective Evolutionary Algorithm based on Decomposition." IEEE transactions on evolutionary computation **11**(6): 712-731.
- Zhou, H.-c., G.-h. Zhang and G.-l. Wang (2007). "Multi-objective decision making approach based on entropy weights for reservoir flood control operation." Journal of hydraulic engineering **38**(1): 100-106.
- Zhou, W., W. Huang and R. Zhang (2014). "A Two-Stage Queueing Network on Form Postponement Supply Chain with Correlated Demands." Applied mathematical modelling **38**(11-12): 2734-2743.
- Zitzler, E., M. Laumanns and L. Thiele (2001). "SPEA2: Improving the Strength Pareto Evolutionary Algorithm." TIK-report **103**.

## APPENDIX A THE 34 WEIGHT SCENARIOS (WSS) AND THE CORRESPONDING ACHIEVEMENT OF THE GOALS

We use in W1, W2, and W3 as the three weights of the goals, and obtain the achievement values of the three goals Goal 1, Goal 2 and Goal 3. The “Total” is the value of “z”, the weighted sum of the three goals. These are the results of the original model. The results facilitate us to obtain the ternary plots (Figure 4.11-4.13) and *satisficing* area (Figure 4.14) in Section 4.2.5.

WS	Weights			Goals				WS	Weights			Goals			
	W1	W2	W3	1	2	3	Total		W1	W2	W3	1	2	3	Total
1	0.33	0.33	0.33	0.23	0.53	0.31	0.44	18	0.4	0.2	0.4	0.23	0.71	0.18	0.41
2	1	0	0	0.00	2.00	2.00	0.05	19	0.4	0.4	0.2	0.22	0.37	0.56	0.45
3	0	1	0	1.00	0.00	2.00	0.00	20	0	0.5	0.5	0.26	0.56	0.25	0.41
4	0	0	1	1.00	2.00	0.00	0.00	21	0.5	0	0.5	0.07	2.00	0.00	0.10
5	0.8	0.1	0.1	0.16	0.73	0.62	0.43	22	0.5	0.5	0	0.10	0.02	2.00	0.13
6	0.1	0.8	0.1	0.24	0.10	1.60	0.29	23	0.5	0.33	0.17	0.21	0.38	0.58	0.45
7	0.1	0.1	0.8	0.31	1.06	0.02	0.19	24	0.17	0.5	0.33	0.24	0.43	0.42	0.44
8	0.6	0.2	0.2	0.20	0.56	0.37	0.45	25	0.33	0.17	0.5	0.24	0.82	0.11	0.37
9	0.2	0.6	0.2	0.23	0.27	0.77	0.42	26	0.67	0.33	0	0.09	0.05	2.00	0.17
10	0.2	0.2	0.6	0.26	0.83	0.09	0.33	27	0	0.67	0.33	0.24	0.37	0.53	0.42
11	0.5	0.25	0.25	0.21	0.54	0.34	0.45	28	0.33	0	0.67	0.08	2.00	0.00	0.06
12	0.25	0.5	0.25	0.23	0.37	0.54	0.44	29	0.56	0.33	0.11	0.20	0.30	0.86	0.44
13	0.25	0.25	0.5	0.25	0.72	0.15	0.39	30	0.11	0.56	0.33	0.24	0.41	0.45	0.44
14	0.4	0.3	0.3	0.22	0.53	0.31	0.45	31	0.33	0.11	0.56	0.24	0.98	0.07	0.32
15	0.3	0.4	0.3	0.23	0.46	0.38	0.45	32	0.22	0.33	0.44	0.24	0.61	0.22	0.41
16	0.3	0.3	0.4	0.23	0.60	0.24	0.43	33	0.44	0.22	0.33	0.22	0.64	0.24	0.43
17	0.2	0.4	0.4	0.24	0.53	0.29	0.43	34	0.33	0.44	0.22	0.22	0.37	0.55	0.44

**APPENDIX B RESULTS OF THE 22 WEIGHT SCENARIOS (WSS) FROM THE IMPROVED MODEL (FIRST ITERATION).**

In Appendix B, we list the 22 weight scenarios (WSs) and the corresponding achievement of the goals. These are the results of the improved model in the first iteration. With the results, we get the *satisficing* area (Figure 4.17) in Section 4.2.5.

WS	Weights			Goals				WS	Weights			Goals			
	W1	W2	W3	1	2	3	Total		W1	W2	W3	1	2	3	Total
1	0.33	0.33	0.33	0.39	0.43	0.30	0.37	12	0.25	0.5	0.25	0.38	0.26	0.54	0.36
2	1	0	0	0.01	2.00	2.00	0.01	13	0.25	0.25	0.5	0.44	0.62	0.13	0.33
3	0	1	0	2.00	0.01	2.00	0.01	14	0.4	0.3	0.3	0.38	0.42	0.31	0.37
4	0	0	1	5.91	2.00	0.00	0.00	15	0.3	0.4	0.3	0.39	0.35	0.38	0.37
5	0.8	0.1	0.1	0.28	0.53	0.63	0.34	16	0.3	0.3	0.4	0.41	0.50	0.21	0.36
6	0.1	0.8	0.1	0.37	0.06	1.33	0.22	17	0.2	0.4	0.4	0.43	0.44	0.26	0.36
7	0.1	0.1	0.8	0.54	0.91	0.01	0.16	18	0.4	0.2	0.4	0.39	0.62	0.16	0.34
8	0.6	0.2	0.2	0.34	0.43	0.38	0.37	19	0.4	0.4	0.2	0.36	0.26	0.58	0.36
9	0.2	0.6	0.2	0.37	0.18	0.75	0.33	20	0	0.5	0.5	0.54	0.48	0.20	0.34
10	0.2	0.2	0.6	0.47	0.72	0.07	0.28	21	0.5	0	0.5	0.14	2.00	0.00	0.07
11	0.5	0.25	0.25	0.36	0.42	0.34	0.37	22	0.5	0.5	0	0.12	0.02	2.00	0.07

## APPENDIX C MATHEMATICAL FORMULATION OF THE CODP AS A CDSP

Appendix C is referenced in Section 4.3.4. It is the mathematical form of cDSP of the case study of CODP design. We explore the design preferences and design capacity in iterations using the algorithm in Table 4.12. After three iterations, the design is improved to overcome the original bottleneck and the solutions are relatively insensitive to the migration of Product Life Cycle phases and the changes in the market demand.

---

### Given

- ts** production time in supplier
- tsm** transportation time from supplier to manufacturer
- tm** production time in manufacturer
- tmr** transportation time from manufacturer to retailer
- tr** operation time in retailer
- trc** transportation time from retailer to customer
- t** delivery time of this supply chain,  
 $t = f_t(\text{CODP}, ts, tsm, tm, tmr, tr, trc)$
- VS** production volume of supplier  
 $VS = f_{sm}(rls, rls, ts, tsm, SRM, SFG)$
- VM** production volume of manufacturer  
 $VM = f_{mr}(rlm, rlmr, tm, tmr, MRM, MFG)$
- VR** preparation volume of retailer  
 $VR = f_{rc}(rlr, rlrc, tr, trc, RRM, RFG)$
- CS** capacity limit of supplier
- CM** capacity limit of manufacturer
- CR** capacity limit of retailer
- tave** average delivery time of the industry
- rlt** reliability of delivery time,  $rlt = f_{rlt}(t, tave)$
- sl** service level,  $sl = rls \cdot rls, rlm \cdot rlmr \cdot rlr \cdot rlrc \cdot rlt$
- slmin** minimum service level of the industry
- pmin** minimum unit price of the industry
- pmax** maximum unit price of the industry
- pave** average unit price of the industry
- p** unit price of retailer's finished goods,  
 $p = f_p(sl, slave, slmin, pave)$
- D** market demand of the retailer's finished goods,

$D = f_D(\text{sl, slave, slmin, pave})$

**hcsrm/scsrm/hcsfg/scsfg/hcmrm/scmr/hcmfg/scmfg/hcrrm/scrrm/hcrfg/scrfg**

unit holding / shortage cost in

supplier's

manufacturer's

retailer's

raw material / finished goods

**pcs/pcm/per**

unit production / operation cost of

supplier

manufacturer

retailer

**tcsm/tcmr/terc**

unit transportation cost from

supplier to manufacturer

manufacturer to retailer

retailer to customers

**FAS/FAM/FAR**

forecast accuracy of

supplier

manufacturer

retailer

**FSFG/FMFG/FRFG**

forecast of the demand for finished goods at

supplier

manufacturer

retailer (if it is MTS)

$FSFG = f_{SFG}(\text{CODP, FAS, rls, rlsm, p})$

$FMFG = f_{MFG}(\text{CODP, FAM, rlm, rlmr, p})$

$FRFG = f_{RFG}(\text{CODP, FAR, rlr, rlr, p})$

---

**Find**

System Variables

**rls** reliability of production of supplier

**rlsm** reliability of transportation from supplier to manufacturer

**rlm** reliability of production of manufacturer

**rlmr** reliability of transportation from manufacturer to retailer

**rlr** reliability of operation of retailer  
**rlrc** reliability of transportation from retailer to customers  
**SRM** CODP at raw material of supplier  
**SFG** CODP at finished goods of supplier  
**MRM** CODP at raw material of manufacturer  
**MFG** CODP at finished goods of manufacturer  
**RRM** CODP at raw material of retailer  
**RFG** CODP at finished goods of retailer  
**SRM** CODP at raw material of supplier  
**SFG** CODP at finished goods of supplier  
**MRM** CODP at raw material of manufacturer  
**MFG** CODP at finished goods of manufacturer  
**RRM** CODP at raw material of retailer  
**RFG** CODP at finished goods of retailer  
 (SRM, SFG, MRM, MFG, RRM, RFG are binary variables)

Deviation Variables

**$d_i^-$**  under-achievement of Goal i  
 **$d_i^+$**  over-achievement of Goal i

---

Satisfy

System Constraints

**SRM+SFG+MRM+MFG+RRM+RFG=1** Constraint 1

//There is one CODP in this supply chain

**$sl \geq slmin$**  Constraint 2

//Service level is greater than or equal to the industry minimum service level

**PTOT  $\geq$  PTmin** Constraint 3

//Total profit is greater than or equal to the industry minimum profit

**$t \leq tave$**  Constraint 4

//Delivery time of this supply chain is no more than the average of the industry

**$p \leq pave$**  Constraint 5

//Unit price of retailer's finished goods is no more than the average of the industry

System Goals

**$\frac{PTOT}{T1} + d_1^- - d_1^+ = 1$**  Goal 1

//To reach profit target

**$\frac{SL}{T2} + d_2^- - d_2^+ = 1$**  Goal 2

//To reach service level target

$$\mathbf{V} + \mathbf{d}_3^- - \mathbf{d}_3^+ = \mathbf{0}$$

Goal 3

//To reach the target of the variance of the reliability of the entities (three entities: supplier, manufacturer, and retailer)

Bounds

$$\mathbf{SRM}, \mathbf{SFG}, \mathbf{MRM}, \mathbf{MFG}, \mathbf{RRM}, \mathbf{RFG} \in \{0, 1\}$$

//The CODP variables are Boolean variables.

$$0.997 \leq \mathbf{rls}, \mathbf{rlsm}, \mathbf{rlm}, \mathbf{rlmr}, \mathbf{rlr}, \mathbf{rlrc} \leq 0.999997$$

//All reliabilities are between  $3\sigma$  and  $6\sigma$

$$0 \leq \mathbf{d}_i^-, \mathbf{d}_i^+ \leq 1, i = 1, 2, 3$$

$$\mathbf{d}_i^- \cdot \mathbf{d}_i^+ = 0, i = 1, 2, 3$$

$$\sum_{i=1}^3 \mathbf{w}_i \cdot (\mathbf{d}_i^- + \mathbf{d}_i^+) = 1$$

---

Minimize

$$\mathbf{Z} = \sum_{i=1}^3 \mathbf{w}_i \cdot (\mathbf{d}_i^- + \mathbf{d}_i^+)$$

//Minimize the weighted sum of the deviation variables



## APPENDIX D THE RMC TUNING ALGORITHM CUSTOMIZED FOR THE HOT ROLLING PROCESS CHAIN PROBLEM (CHAPTER 5)

Appendix D is the RMC tuning algorithm (Table 5.5) customized for the Cooling Procedure of the Hot Rolling Problem. Appendix D is referenced in Section 5.5.4. This algorithm is an extension of the algorithm in Table 5.5. More auxiliary parameters are defined to assist RMC tuning –  $T$ ,  $\alpha, \beta, \gamma, \theta, \iota, \kappa, M$ . For the parameters that are relatively more important (the results are more sensitive to their values), e.g.  $\theta$ , we tune their values. For the parameters that are relatively less important, e.g.  $\iota, \kappa, M$ , we set values to them with heuristics.

---

```

1   $t \leftarrow 0$  // Initiate the number of synthesis cycles
2   $b \leftarrow 0$  // Initiate the time of updating the best RMC
3   $RMC_t \leftarrow$  a random value // Initiate RMC with a random value (here we set  $RMC_0 \leftarrow 0.5$ )
4   $best_b \leftarrow$  a random value // Initiate the “best RMC” with a random value (here we set  $best_0 \leftarrow 0.5$ )
5  While  $t \leq T$  Do // Search for best RMC for T synthesis cycles (the first stop criterion, here we set  $T \leftarrow 20$ )
6      if  $EI[RMC_{t-1}] > EI[RMC_{t-2}]$  // If  $RMC_{t-1}$  performs better than  $RMC_{t-2}$ 
7           $RMC_t \leftarrow RMC_{t-1} + \alpha \cdot (RMC_{t-1} - RMC_{t-2})$  // Update  $RMC_t$  in the improving direction ( $\alpha \in [0,1]$ , and
            here we set  $\alpha$  as random values that uniformly distributed in  $[0, 1]$ )
8      else if  $best_b \neq RMC_{t-2}$ 
9           $RMC_t \leftarrow \beta \cdot best_b + (1 - \beta) \cdot PEI^{-1}[\text{better}\{PEI[RMC_{t-2}], PEI[best_{b-1}]\}]$  // Update  $RMC_t$  as the
            linear
            combination of best RMC and the last best RMC ( $best_{b-1}$ ) iff  $best_{b-1}$  performs better than  $RMC_{t-2}$ ;
            otherwise
            update  $RMC_t$  as the linear combination of best RMC and  $RMC_{t-2}$  ( $\beta \in [0,1]$ , and here we set  $\beta$  as random
            values that uniformly distributed in  $[0.5, 1]$ )
10     else if  $best_{b-1}$ 
11          $RMC_t \leftarrow \beta \cdot best_b + (1 - \beta) \cdot best_{b-1}$  // Update  $RMC_t$  as the linear combination of the current best
            RMC and the last best RMC
12     else
13          $RMC_t \leftarrow \beta \cdot RMC_{t-1} + (1 - \beta) \cdot RMC_{t-2}$ 
14     if  $RMC_t > 1$  // If  $RMC_t$  is larger than its upper bound
15          $RMC_t \leftarrow 1$  // Pull  $RMC_t$  back to range
16     if  $RMC_t < 0$  // if  $RMC_t$  is lower than its lower bound

```

```

17 |  $RMC_t = 0$  // Pull  $RMC_t$  back to range
18 |  $I <- 0$  // Initiate the number of EIs of  $RMC_t$  in  $DEI$ 
19 |  $J <- 0$  // Initiate the number of EIs of  $RMC_t$  that are better than the EIs of best RMC
20 |  $K <- 0$  // Initiate the number of EIs of  $RMC_t$  that violate  $DEI$  within  $\theta$ . We tune  $\theta$  maximizing the  $L_2 - norm$ 
    | distance between EIs of two adjacent iterations and get 10%
21 |  $L <- 0$  // Initiate the number of EIs of  $RMC_t$  that are better than the EIs of  $RMC_{t-1}$ 
22 | for  $i$  in 1 to  $n$  // For all the EIs ( $n$  is the number of EIs)
23 | | if  $PEI[RMC_t]_i \in DEI_i$  // If the  $i^{th}$  EI of  $RMC_t$  is in the desired range of the  $i^{th}$  EI
24 | | |  $I <- I + 1$  // Update the number of EIs of  $RMC_t$  in  $DEI$ 
25 | | | if  $EI[RMC_t]_i \geq EI[best]_i$ 36 // If for the  $i^{th}$  EI  $RMC_t$  performs better than or equal to best RMC
26 | | | |  $J <- J + 1$  // Update the number of EIs of  $RMC_t$  that are better than best RMC
27 | | | if  $EI[RMC_t]_i \notin DEI_i$  and  $EI[RMC_t]_i \in [DEI_i]_{-\gamma_i}^{+\gamma_i}$  // If the  $i^{th}$  EI of  $RMC_t$  violates the desired range of the
    | | |  $i^{th}$  EI within  $\gamma_i$  (in this problem,  $\gamma_i=30\%$ )
28 | | | |  $K <- K + 1$  // Update the number of EIs of  $RMC_t$  that violate  $DEI$  within  $\gamma_i$ 
29 | | | if  $EI[RMC_t]_i \geq EI[RMC_{t-1}]_i$  // If for the  $i^{th}$  EI,  $RMC_t$  performs better than or equal to  $RMC_{t-1}$ 
30 | | | |  $L <- L + 1$  // Update the number of EIs that  $RMC_t$  improves versus  $RMC_{t-1}$ 
31 | if  $I \geq \iota \cdot n$  and  $I + K = n$  // If for at least  $\iota$  (we set it as 2/3) EIs are in  $DEI$ , and the violation rate are all
    | within  $\gamma_i$ 
32 | | if  $L \geq \kappa \cdot n$  // If at least  $\kappa$  (we set it as 2/3) EIs are better than previous synthesis cycle
33 | | |  $EI[RMC_t] > EI[RMC_{t-1}]$  // We define that  $RMC_t$  overall performs better than  $RMC_{t-1}$ 
34 | | | if  $J \geq \kappa \cdot n$  // If at least  $\kappa$  EIs is better than best RMC
35 | | | |  $best <- RMC_t$  // Update best RMC
36 | | | |  $nupdt <- -1$  // Reset no updating pointer “nupdt” as “-1”
37 | | | if  $K \geq 1$  // If at least one violation EI
38 | | | |  $DEI_i <- EI[RMC_t]_i$  which  $EI[RMC_t]_i \notin DEI_i$  and  $EI[RMC_t]_i \in [DEI_i]_{-\theta}^{+\theta}$  // Update  $DEI_i$ 
    | | | |  $nupdt <- nupdt + 1$  // Increase no updating pointer “nupdt” by 1
39 | | else
40 | | |  $nupdt <- nupdt + 1$  // Increase no updating pointer “nupdt” by 1
41 | | if  $nupdt \geq M$  // If no updating in  $M$  synthesis cycles in a row (the second stop criterion, and here we set  $M <-$ 
    | 5)
42 | | | Break
43 | |  $t <- t+1$  // Move on to the next synthesis cycle
44 | Return  $best$  // Return the final best RMC as the appropriate RMC

```

---

<sup>36</sup>  $EI^i[RMC_t] > EI^i[best]$  means for the  $i^{th}$  PEI, comparing with the EI of current best, the EI of  $RMC_t$  is closer to the ideal value of the PEI.

Steps in the RMC-tuning algorithm:

Determine the evaluation indices (EIs) based on multiple criteria to classify good results from the bad ones;

Initialize the desired range of each EI (DEI) of the test problem;


Identify auxiliary parameters to assist RMC tuning;

Bring the EIs into DEI by tuning the auxiliary parameters;

Update DEI to ensure a proportion of good results out of all results;

Tradeoff between elitism and randomness to ensure a diversity while getting fast convergence.

# APPENDIX E LIN GUO'S DEFENSE SLIDES AND SPEECH



Ph.D. Dissertation Defense

## Model Evolution for the Realization of Complex Systems

Presented by  
**Lin Guo**  
Ph.D. Candidate  
School of Industrial and Systems Engineering  
University of Oklahoma

Co-chairs

**Dr. Janet K. Allen**  
School of Industrial and Systems Engineering

**Dr. Farrokh Mistree**  
School of Aerospace and Mechanical Engineering

Committee


**Dr. Theodore B. Trafalis**  
School of Industrial and Systems Engineering

**Dr. Charles D. Nicholson**  
School of Industrial and Systems Engineering


**Dr. Thomas M. Neeson**  
Department of Geography and Environmental Sustainability  
University of Oklahoma

July 12, 2021

*Welcome to my defense. I am Lin Guo. My dissertation is model evolution for the realization of complex systems. My committee includes Dr. Allen, Dr. Mistree, who are my advisors. And Dr. Trafalis and Dr. Nicholson from ISE. And Dr. Neeson from Geography and Environmental Sustainability.*



Model Evolution for the Realization of Complex Systems  
Lin Guo



---

### Overview of Today's Presentation

<h4>Addressing the Committee's Questions</h4> <ul style="list-style-type: none"> <li>▪ <b>Dr. Trafalis:</b> how is your method different from optimization? (<i>Section 1.2, 1.4, 2.1, 2.2, 2.3</i>)</li> <li>▪ <b>Dr. Nicholson:</b> (for Chapter 6), how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors? (<i>Section 6.14, 6.3</i>)</li> <li>▪ <b>Dr. Neeson:</b> how does the Red River project help or relevant to your future work? (<i>Section 9.2.2, 9.2.3</i>)</li> </ul>	<h4>Frame of Reference</h4> <ul style="list-style-type: none"> <li>▪ Characteristics and challenges in designing complex systems (<i>Section 1.1, 1.3</i>)</li> <li>▪ Modeling strategies – optimizing and <i>satisficing</i> – and their problems and differences (<i>Section 1.2, 1.4, 2.2, 2.3</i>)</li> <li>▪ Research gaps (<i>Section 1.5</i>)</li> </ul>
<h4>A Test Problem</h4> <ul style="list-style-type: none"> <li>▪ Dissertation layout (<i>Section 1.7</i>)</li> <li>▪ Research Questions and Hypotheses                             <ul style="list-style-type: none"> <li>▪ Research Question 3: What is the method to speed up learning the system nature, such as the interrelationship among the subsystems? (<i>Section 2.4.2</i>)</li> <li>▪ Specific Hypothesis 3: Learning interrelationship among subsystems using unsupervised learning and reorganize them based on it. (<i>Section 2.5</i>)</li> <li>▪ Method 3: Adaptive Leveling-Weighting-Clustering algorithm (<i>Section 3.3.3</i>)</li> <li>▪ Test Problem 3: Rankine cycle thermal system design (<i>Chapter 6</i>)</li> </ul> </li> </ul>	<h4>Contribution and Way forward</h4> <ul style="list-style-type: none"> <li>▪ Answering the research questions (<i>Section 8.1.3</i>)</li> <li>▪ Managing four types of uncertainties (<i>Section 8.1.4</i>)</li> <li>▪ Verification and Validation (<i>Section 1.6, 9.1.3</i>)</li> <li>▪ Relevant publications (<i>Section 9.1.4</i>)</li> <li>▪ Research thrusts and application in my early career (<i>Section 9.2</i>)</li> </ul>

2 / 28

In today's presentation, I will address my committee's questions from last time. First, Dr. Trafalis asked "how is the method used in this dissertation different from optimization?" I will introduce the frame of reference. Along the way, I will answer this question.

Overview of Today's Presentation	
<p><b>Addressing the Committee's Questions</b></p> <ul style="list-style-type: none"> <li>▪ <b>Dr. Trafalis:</b> how is your method different from optimization? (Section 1.2, 1.4, 2.1, 2.2, 2.3)</li> <li>▪ <b>Dr. Nicholson:</b> (for Chapter 6), how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors? (Section 6.14, 6.3)</li> <li>▪ <b>Dr. Neeson:</b> how does the Red River project help or relevant to your future work? (Section 9.2.2, 9.2.3)</li> </ul>	<p><b>Frame of Reference</b></p> <ul style="list-style-type: none"> <li>▪ Characteristics and challenges in designing complex systems (Section 1.1, 1.3)</li> <li>▪ Modeling strategies – optimizing and <i>satisficing</i> – and their problems and differences (Section 1.2, 1.4, 2.2, 2.3)</li> <li>▪ Research gaps (Section 1.5)</li> </ul>
<p><b>A Test Problem</b></p> <ul style="list-style-type: none"> <li>▪ Dissertation layout (Section 1.7)</li> <li>▪ Research Questions and Hypotheses <ul style="list-style-type: none"> <li>▪ Research Question 3: What is the method to speed up learning the system nature, such as the interrelationship among the subsystems? (Section 2.4.2)</li> <li>▪ Specific Hypothesis 3: Learning interrelationship among subsystems using unsupervised learning and reorganize them based on it. (Section 2.5)</li> <li>▪ Method 3: Adaptive Leveling-Weighting-Clustering algorithm (Section 3.3.3)</li> <li>▪ Test Problem 3: Rankine cycle thermal system design (Chapter 6)</li> </ul> </li> </ul>	<p><b>Contribution and Way forward</b></p> <ul style="list-style-type: none"> <li>▪ Answering the research questions (Section 8.1.3)</li> <li>▪ Managing four types of uncertainties (Section 8.1.4)</li> <li>▪ Verification and Validation (Section 1.6, 9.1.3)</li> <li>▪ Relevant publications (Section 9.1.4)</li> <li>▪ Research thrusts and application in my early career (Section 9.2)</li> </ul>

2 / 28

Dr. Nicholson asked, "for Chapter 6, how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors?" I will give the whole picture of my dissertation, including its layout and all the research questions and hypotheses, among which, I will go deeper in Research Question 3 and Chapter 6, to address Dr. Nicholson's question.



## Overview of Today's Presentation

### Addressing the Committee's Questions

- **Dr. Trafalis:** how is your method different from optimization? (Section 1.2, 1.4, 2.1, 2.2, 2.3)
- **Dr. Nicholson:** (for Chapter 6), how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors? (Section 6.14, 6.3)
- **Dr. Neeson:** how does the Red River project help or relevant to your future work? (Section 9.2.2, 9.2.3)

### Frame of Reference

- Characteristics and challenges in designing complex systems (Section 1.1, 1.3)
- Modeling strategies – optimizing and *satisficing* – and their problems and differences (Section 1.2, 1.4, 2.2, 2.3)
- Research gaps (Section 1.5)

### A Test Problem

- Dissertation layout (Section 1.7)
- Research Questions and Hypotheses
  - Research Question 3: What is the method to speed up learning the system nature, such as the interrelationship among the subsystems? (Section 2.4.2)
  - Specific Hypothesis 3: Learning interrelationship among subsystems using unsupervised learning and reorganize them based on it. (Section 2.5)
- Method 3: Adaptive Leveling-Weighting-Clustering algorithm (Section 3.3.3)
- Test Problem 3: Rankine cycle thermal system design (Chapter 6)

### Contribution and Way forward

- Answering the research questions (Section 8.1.3)
- Managing four types of uncertainties (Section 8.1.4)
- Verification and Validation (Section 1.6, 9.1.3)
- Relevant publications (Section 9.1.4)
- Research thrusts and application in my early career (Section 9.2)

2 / 28

*Dr. Neeson asked, “how does the Red River project help or relevant to your future work?” I will summarize the contribution and indicate the way forward. Along the way, I will introduce the research thrusts and applications in my early career, especially the ones based on the Red River project.*



## Overview of Today's Presentation

### Addressing the Committee's Questions

- **Dr. Trafalis:** how is your method different from optimization? (Section 1.2, 1.4, 2.1, 2.2, 2.3)
- **Dr. Nicholson:** (for Chapter 6), how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors? (Section 6.14, 6.3)
- **Dr. Neeson:** how does the Red River project help or relevant to your future work? (Section 9.2.2, 9.2.3)

### Frame of Reference

- Characteristics and challenges in designing complex systems (Section 1.1, 1.3)
- Modeling strategies – optimizing and *satisficing* – and their problems and differences (Section 1.2, 1.4, 2.2, 2.3)
- Research gaps (Section 1.5)

### A Test Problem

- Dissertation layout (Section 1.7)
- Research Questions and Hypotheses
  - Research Question 3: What is the method to speed up learning the system nature, such as the interrelationship among the subsystems? (Section 2.4.2)
  - Specific Hypothesis 3: Learning interrelationship among subsystems using unsupervised learning and reorganize them based on it. (Section 2.5)
- Method 3: Adaptive Leveling-Weighting-Clustering algorithm (Section 3.3.3)
- Test Problem 3: Rankine cycle thermal system design (Chapter 6)

### Contribution and Way forward

- Answering the research questions (Section 8.1.3)
- Managing four types of uncertainties (Section 8.1.4)
- Verification and Validation (Section 1.6, 9.1.3)
- Relevant publications (Section 9.1.4)
- Research thrusts and application in my early career (Section 9.2)

3 / 28

*First, let me frame the reference of this dissertation and answer Dr. Trafalis's question. How is the method used in this dissertation different from optimization?*

*I will frame the reference by analyzing the demand side, which is the characteristics and challenges in designing complex systems, then the supply side, which is modeling strategies – optimizing and satisficing – and their problems and differences, based on which, I come up with the research gaps.*

Systems Realization Laboratory @ OU      Model Evolution for the Realization of Complex Systems      Lin Guo      GATEWAY COLLEGE OF ENGINEERING      SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING      UNIVERSITY OF OKLAHOMA

**Observation – Characteristics and challenges in designing complex systems** (Section 1.1, 1.3)

**Model World**      **Physical World**



www.usapangecon.com

Frame of Reference      A Test Problem      Contribution and Way Forward      4 / 28

*If we categorize everything in the model world and physical world, we may observe that there is an intellectual disconnection between them. In the model world, we desired a more organized world. We expect to obtain all information. If not, we hope that we can predict all kinds of uncertainties. But in the physical world, things are quite the opposite. We may have chaos and we cannot capture all the information. We may not have time to get aware of or respond to the emergent properties. That is where and how the intellectual disconnection comes from.*



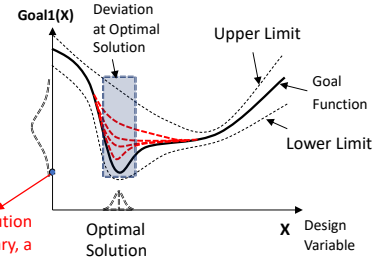
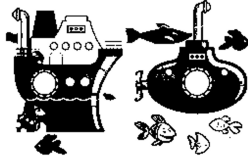
### Examples



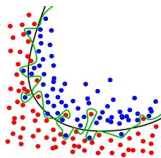
Essentially, all models are **wrong**, but some are **useful**.

- George E.P. Box, British Statistician

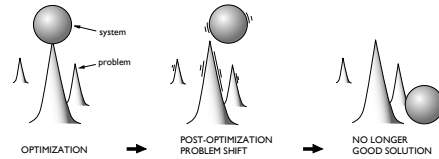
Design a ship and get a submarine



Overfitting noisy data



No Data



Frame of Reference

A Test Problem

Contribution and Way Forward

5 / 28

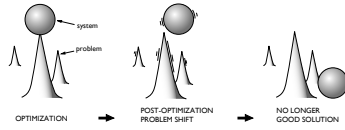
As George Box said, “all models are wrong, but some are useful,” we may encounter some difficulties when designing a complex system. For example, we may want to design a ship but end up with a submarine because we fail to capture all the requirements. We may overfit the noisy data, as a result, we make wrong decisions. For a new project, we may not have any data at all. We may lose an optimal solution due to variations. When visualize it in a 2D plane, where the horizontal axis is a decision variable and the vertical axis is the value of the goal function, and the goal function is the bold black line, we may get an optimal solution, which is on the boundary of the model bounded by constraints or bounds and it is a single point. However, the physical system may be one of the red dotted lines. Designers sometimes fail to capture the gap between the black line and the red dotted lines. As a result, the single optimal solution may not work in the physical world. There can be a deviation at the optimal solution. That is where some complex-system designers’ hassle comes from. That is the analysis of the “demand side.”





## Modeling strategies – optimizing and *satisficing* (Section 1.2)

### Optimize

**Given**

$$f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n$$

$$\Omega$$

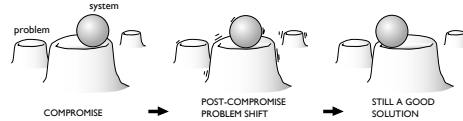
$$= \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k\}$$

**Find**

$$x^*: f(x^*) \geq f(x), \forall x \in \Omega$$

- The model is incomplete and inaccurate
- The boundary evolves
- Design preferences evolve
- Multiple types of uncertainty
- Sensitive to parameter setting
- Interactions among subsystems
- Emergent properties not captured

### Satisfice

**Given**

$$f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n$$

$$\Omega$$

$$= \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k, f(x) + d^- - d^+ = Target\}$$

**Find**

$$x^s: \mathcal{P}_{x \in \Omega} (f(x) = Target)$$

- Manage incompleteness and inaccuracies by adding buffers
- Use fuzzy solutions to replace exact solutions
- High computational complexity in post-solution analysis but no reusable knowledge

Go to backup slide 34 for details in critically reviewing the literature

Frame of Reference

A Test Problem

Contribution and Way Forward

6 / 28

Now, it is the analysis of the “supply side.”

We categorize all the modeling methods into two categories – optimizing strategy and satisficing strategy. How did we come up with this conclusion?



## How is *satisficing* different from optimization? (Section 1.2)

Table 1. 2 Advantages and Disadvantages of The Two Categories of Solution Algorithms

	Category	Example Methods	Advantages	Disadvantages
Optimizing Strategy	Formulate a problem exactly and solve it approximately	Gradient-based methods, pattern search methods, penalty function methods, etc.	- Maintaining a relatively accurate model along the solution search (given the information that the designer has on hand).	- The solution is still an approximate, inaccurate one; - Cannot get the information of the dual and use it to facilitate problem solving or post-solution analysis; - Heuristics are used in solution algorithms, which may result in premature convergence or unnecessarily high computational complexity.
Satisficing Strategy	Approximate a problem and solve it exactly	ALP, SLP, SQL, etc.	- Solutions are on the vertices of the approximated problem so the dual of the approximated problem can be explored; - Solutions may be away from the boundary of the original problem so they are relatively insensitive to variations; - The approximation of the problem can be improved by accumulating the linearized constraints during iterating and an approximated problem with acceptable level of accuracy can be obtained.	- Introducing information loss while doing approximation, making the solution inaccurate; - Heuristics are used in approximation algorithms, which may result in premature convergence or unnecessarily high computational complexity.

Backup Slides

34

We critically review the literature on modeling methods. Among 99 publications, essentially, there are two modeling strategies. (Let's go to the backup slides for more information.)



# How is *satisficing* different from optimization? (Section 1.2)

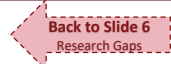


Table 1. 3 Several Representative Methods and Their Features (based on 99 publications)

Construct	Method	Managing complexity				Converge condition / solution feature		Solution algorithms				Decision support		
		Managing nonlinearity	Managing nonconvexity	Managing uncertainty	Managing multiple objectives (goals)	Managing hard requirements (constraints)	Necessary KKT conditions	Sufficient KKT conditions	Interior-point searching	(Linearizing and) obtaining vertex solutions	Comprehensively using interior-point searching and searching for local solutions	Can apply visualization tools to assist decision making	Managing knowledge discovery and reuse	Supporting exploring design preferences
Mathematical Programming seeking optimal solutions	Mathematical programming													
	Mathematical programming using AI/ML													
Goal programming	Goal programming													
	Goal programming using AI/ML													
CDSP	CDSP using AI/ML													
	CDSP using AI/ML													

Backup Slides

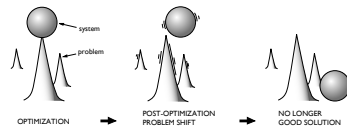
35

In Table 1.3, we illustrate the major issues that designers focus on when they design complex systems. The scholars may have various foci, but their methods all fall into two categories – optimizing and satisficing. (Let's go back to the main slides.)



## Modeling strategies – optimizing and satisficing (Section 1.2)

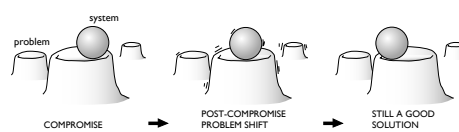
### Optimize



**Given**  
 $f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n$   
 $\Omega = \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k\}$   
**Find**  
 $x^*: f(x^*) \geq f(x), \forall x \in \Omega$

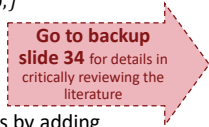
- The model is incomplete and inaccurate
- The boundary evolves
- Design preferences evolve
- Multiple types of uncertainty
- Sensitive to parameter setting
- Interactions among subsystems
- Emergent properties not captured

### Satisfice



**Given**  
 $f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n$   
 $\Omega = \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k, f(x) + d^- - d^+ = Target\}$   
**Find**  
 $x^s: \mathcal{P}_{x \in \Omega} (f(x) = Target)$

- Manage incompleteness and inaccuracies by adding buffers
- Use fuzzy solutions to replace exact solutions
- High computational complexity in post-solution analysis but no reusable knowledge



Frame of Reference

A Test Problem

Contribution and Way Forward

6 / 28

In optimizing strategy, designers seek the optimal solution by maximizing (or minimizing) an objective function through determining the value of decision variables. Whereas in satisficing strategy, we have a target value for each objective, so the objective becomes a goal. A goal is an equation with the objective on the left-hand side and its target value on the right-hand side. So,

*the designers seek the nearest projection of the goal onto the feasible space bounded by the constraints and bounds. This is a satisficing solution. This is the difference.*

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

## Response to Dr. Trafalis: How is *satisficing* different from optimization?

- a) Why can designers obtain good enough but relatively robust solutions using *satisficing strategy*, such as the compromise Decision Support Problem (cDSP) with the Adaptive Linear Programming algorithm (ALP) implemented in the decision support in the design of engineering systems (DSIDES)?

### Optimizing vs. Satisficing

- *Difference in the assumptions regarding the KKT conditions* (Section 2.1)

Satisficing

Optimizing

**Assumption/Requirement 1** – mathematical models are 100% complete and accurate abstractions of physical problems.

**Assumption/Requirement 2** – all equations of the problem are differentiable.

**Assumption/Requirement 3** – the convexity degree of at least one non-zero linear combination of all constraints is higher than the convexity degree of the objective function.

Go to backup slide 36 for details

Frame of Reference

A Test Problem

Contribution and Way Forward

7 / 28

*What does this difference mean to us? Let me answer this, meanwhile, respond to Dr. Trafalis's question "how is satisficing difference from optimization?" I will answer this question by answering three sub-questions. The first is "why can designers obtain good enough but relatively robust solutions using satisficing strategy, such as using the compromise decision support problem (cDSP), which is the formulation construct, with the Adaptive Linear Programming Algorithm (ALP), which is the solution algorithm, and using the decision support in the design of engineering systems (DSIDES), which is the platform that implements the cDSP and ALP? (In this dissertation, when we mention satisficing strategy, we always refer to the cDSP, ALP, and DSIDES as particular tools to realize satisficing strategy. Other people may have other ways of realizing satisficing strategy, but we realize it using the cDSP, ALP, and DSIDES.) If we explain the difference using the KKT conditions, or Kuhn-Tucker conditions, we observe that when using optimizing strategy, designers have to accept three assumptions: First, the mathematical model is a perfect abstraction of the physical system, so the optimal solution to the mathematical model is also the optimal solution to the physical problem. Second, all equations of the problem are differentiable. Third, the convexity degree of at least one non-zero linear combination of all constraints is higher than the convexity degree of the objective function. The non-zero vectors that combine the constraints are Lagrange multipliers. Using satisficing strategy, designers only accept one assumption, the second one. What is the difference between having three assumptions and having one? (Let me use the backup slides to explain.)*

## How is *satisficing* different from optimization? (Section 2.1)

- a) Why can designers obtain good enough but relatively robust solutions using *satisficing* methods, such as the compromise Decision Support Problem (cDSP) with the Adaptive Linear Programming algorithm (ALP) implemented in the decision support in the design of engineering systems (DSIDES)?

### First-order (necessary) KKT conditions

Stationary:

$$\nabla f(x^*) + \sum_{i=1}^m \mu_i \nabla g_i(x^*) - \sum_{j=1}^{\ell} \lambda_j \nabla h_j(x^*) = 0$$

Primal feasibility:

$$g_i(x^*) \geq 0, \forall i = 1, \dots, m$$

$$h_j(x^*) = 0, \forall j = 1, \dots, \ell$$

Dual feasibility:

$$\mu_i \geq 0, \forall i = 1, \dots, m$$

Complementary slackness:

$$\mu_i g_i(x^*) = 0, \forall i = 1, \dots, m$$

### Physical meaning

At the solution point  $x^*$ , where both the primal and the dual are feasible, the gradient vector of the objective  $\nabla f(x^*)$  can be represented as the non-zero linear combination of the gradient matrix of all equality constraints  $h_j(x^*)$  and the active inequality constraints  $\nabla g_i(x^*)$  for  $i$  iff  $g_i(x^*) = 0$ .

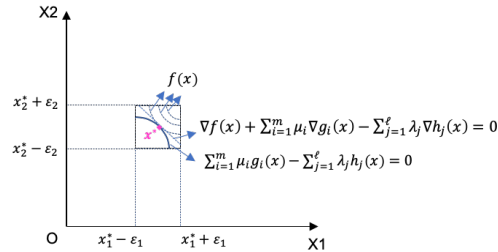


Figure 2. 2 The first-order necessary KKT conditions are satisfied at  $x^*$

Backup Slides

36

First, let's review the KKT conditions. The first-order or the necessary KKT conditions have four parts. The stationary, primal feasibility, dual feasibility, and complementary slackness. What is their physical meaning? It means, at a solution point, the primal and the dual are feasible, which explains the primal feasibility and dual feasibility, and the objective function is tangent to the linear combination of all quality constraints and the active inequality constraints, which explains the stationary and complementary slackness. The vectors linearly combine the constraints are Lagrange multipliers. This is the meaning of necessary KKT conditions. Both optimal solutions and satisficing solutions meet the necessary KKT conditions.

## How is *satisficing* different from optimization? (Section 2.1)

### Optimizing vs. *Satisficing*

- *Difference in the assumptions regarding the KKT conditions*

#### Second-order sufficient conditions

For the Lagrangian:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x)$$

$$\Rightarrow s^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) s \geq 0, \text{ where } s \neq 0$$

And

$$[\nabla_x g_i(x^*), \nabla_x h_j(x^*)]^T s = 0$$

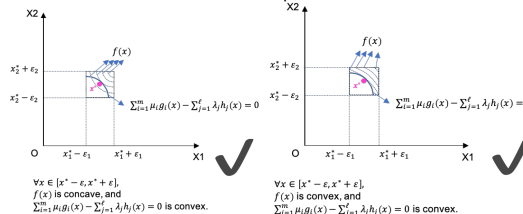


Figure 2.3 The convexity requirements for satisfying the second-order sufficient KKT conditions

Backup Slides

#### Physical meaning:

At the solution point  $x^*$ , there exists a nonzero vector  $s$  that is orthogonal to the gradient matrix of all active inequality and equality constraints, such that the second-order matrix of the Lagrange's equation with respect to decision variables  $x^*$  and Lagrange multipliers  $\lambda^*$  and  $\mu^*$  is conditionally positive semidefinite:  $s^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) s \geq 0, \forall s \in \mathcal{S}$ .

In a small range around  $x^*$ , the convexity degree of the objective should not exceed the convexity degree of the constraints combined by Lagrange multipliers.

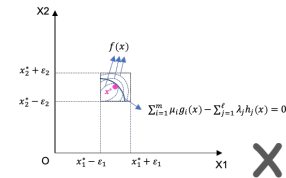


Figure 2.4 Lagrange multipliers fail to identify an optimal for a highly convex objective

37

The difference lies in the second-order or sufficient KKT conditions. The physical meaning of the sufficient conditions is that at the solution point  $x^*$ , there exists a nonzero vector  $s$  that is orthogonal to the gradient matrix of all active inequality and equality constraints, such that the second-order matrix of the Lagrange's equation with respect to decision variables  $x^*$  and Lagrange multipliers  $\lambda^*$  and  $\mu^*$  is conditionally positive semidefinite:  $s^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) s \geq 0, \forall s \in \mathcal{S}$ .

In other words, it means that in a small range around  $x^*$ , the convexity degree of the objective should not exceed the convexity degree of the constraints combined by Lagrange multipliers. We define the convexity degree of an equation as the average value of the diagonal terms of the equation's Hessian matrix at  $x^*$ .

If we visualize this, it means for the first two cases, there exists optimal solutions – either the linear combination of the constraints (combined by the Lagrange multipliers) is convex whereas the objective is concave, or the linear combination of the constraints is convex, but the objective is less convex. However, for the third case, we may have some problem of seeking an optimal solution because the convexity of the objective is larger than the convexity of the linear combination of the constraints.

Using optimizing strategy, designers try to identify an optimal solution meeting the sufficient KKT condition, so for the third case, they may fail to find a solution. But using *satisficing* strategy, since we do not need to meet the sufficient conditions, we can still find a *satisficing* solution for the third case. In engineering design, we may face a lot of situations as the third case, when the objective has a high convexity degree, so we desire using *satisficing* strategy. It does not mean that meeting the sufficient KKT conditions are not useful. Nevertheless, considering the challenges in our demand side – the characteristics of complex system design problems, we prefer to choose *satisficing* strategy to manage our problems, by only meeting the necessary KKT conditions.

That is the answer to the first sub-question. (Let me go back to main slides.)



## Response to Dr. Trafalis: How is *satisficing* different from optimization?

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

**Table 2. 1** The Advantages of Realizing *Satisficing* Strategy using cDSP and ALP implemented on DSIDES (Section 2.2, 2.3)

Stage	Feature	Advantage	Introduction and Discussion
Formulation	Using Goals and Minimizing Deviation Variables Instead of Objectives	At a solution point, only the necessary KKT conditions are met, whereas the sufficient KKT conditions do not have to be met. Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and unparameterizable uncertainties.	Section 2.2.7
	Using second-order sequential linearization	Designers can have a balance between linearization accuracy and computational complexity.	Section 2.2.5 and 5.2.1
Approximation	Using accumulated linearization	Designers can manage nonconvex problems in a way, and deal with highly convex, nonlinear problems relatively more accurately.	
	Combining interior-point searching and vertex searching	Designers can avoid being stuck into local optimum to some extent and identify <i>satisficing</i> solutions relatively insensitive to starting points changing.	Section 2.2.3
Exploration	Allowing some violations of soft requirements, such as the bounds of deviation variables	Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility. As a result, goals and constraints with different scale can be managed	Section 2.2.9

Go to backup slide 38 for details

Frame of Reference

A Test Problem

Contribution and Way Forward

8 / 28

The second sub-question is “how can designers obtain *satisficing* solutions using cDSP, ALP, and DSIDES,” given that *satisficing* solutions are more desired in complex systems design? How do we ensure these methods can return us *satisficing* solutions? Here are the reason. They are in my Table 2.1.

We define that there are four stages in designing a complex system – formulation, approximation, exploration, and evaluation. In each of these four stages, there are some differences when using *satisficing* strategy. For example, in formulation, we use goals instead of objectives, and we minimize the deviation variables between the achieved value and the target of a goal instead of minimizing (or maximizing) the objective function consisting of decision variables. This part is similar to the Goal Programming. I will use several toy problems to illustrate the differences in each stage. (Go to backup slides.)



## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 2 The Features of the Toy Problems (TP)

Toy Problem (TP)	I	II	III	IV	V
Feature					
Two objectives	*	*	*	*	*
Nonlinear	*	*	*	*	*
Non-convex		*	*	*	*
Objectives with various units (scale)			*	*	*
Target of goals with various levels of achievability				*	*
More than two objectives					*

Table 2. 3 Methods for Comparison the Two Strategies

Strategy	Optimizing	Satisficing
<b>Item</b>		
<b>Model formulation construct</b>	Mathematical programming or Goal programming	Compromise Decision Support Problem
<b>Solution algorithm</b>	Constrained Optimization by Linear Approximation (COBYLA) algorithm	Adaptive Linear Programming (ALP) algorithm
	Trust-region constrained (trust-constr) algorithm	
	Sequential Least Squares Programming (SLSQP) algorithm	
	Nondominated Sorting Generation Algorithm II/III (NSGA II/III)	
<b>Solver</b>	Python Scipy.optimize	DSIDES

Backup Slides

38

In Chapter 2, I use five toy problems. I call them “toy problems” because they are very simple problems that allow us to tell the differences between optimizing and satisficing strategy. And I want to separate them with the “test problems” in later chapters (which are used to demonstrate the proposed methods in this dissertation.) For the toy problems in Chapter 2, each of them encounters one more complexity.

For model formulation construct, for optimizing strategy, I use either Mathematical Programming or Goal Programming as the formulation construct, and for satisficing strategy I use cDSP.

For solution algorithm, for optimizing strategy, I use Constrained Optimization by Linear Approximation (COBYLA) algorithm, Trust-region constrained (trust-constr) algorithm, Sequential Least Squares Programming (SLSQP) algorithm, and Nondominated Sorting Generation Algorithm II/III (NSGA II/III), for satisficing strategy, I use ALP.

As to solver, I use Python Scipy.optimize for optimizing and DSIDES for satisficing.

Due to the time limitation, I will only go through Toy Problem II and III to demonstrate the differences.

## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 6 The Optimization Model and Compromise DSP of the TP-II

Strategy TP	Optimizing	Satisficing
II	<b>Objective Functions</b> $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = \frac{1}{2} \cdot (x1 - 2)^3 + (x2 - 2)^3 + x1 \cdot x2^2$	<b>Given</b> $x1, x2, d1^\pm, d2^\pm$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = \frac{1}{2} \cdot (x1 - 2)^3 + (x2 - 2)^3 + x1 \cdot x2^2$
	<b>Constraints and Bounds</b> s. t. $\begin{cases} x1 \cdot x2 \leq 1 \\ 0 \leq x1 \leq 2 \\ 0 \leq x2 \leq 2 \end{cases}$	<b>Find</b> $x1, x2, d1^\mp, d2^\mp$
	<b>Combination of Objective Functions</b> $Max \sum_{i=1}^2 w_i \cdot f_i(x)$	<b>Satisfy</b> <b>Goals:</b> $\frac{f_1(x)}{12} + d1^- = 1$ $\frac{f_2(x)}{8} + d2^- = 1$
		<b>Constraints:</b> $x1 \cdot x2 \leq 1$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$
		<b>Bounds:</b> $0 \leq x1, x2 \leq 2$ $0 \leq d1^\pm, d2^\pm \leq 1$
	<b>Minimize</b> $Z = \sum_{i=1}^2 w_i \cdot di^-$	

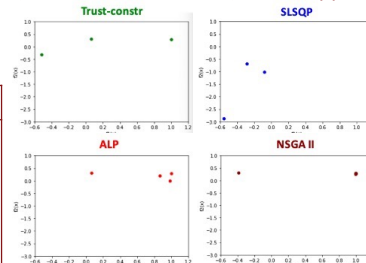


Figure 2. 10 The Solution Points to TP-II on the Objective Space Using Four Algorithms – Solutions returned by Trust-constr and SLSQP are not “good enough,” solutions returned by ALP are “good enough” and diverse, and solutions returned by NSGA II contain nondominated solutions but are not diverse

**Observation:** for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, some optimizing algorithms (e.g. COBYLA) cannot manage the non-convexity, whereas some other optimizing algorithms are easy to converge local optima. NSGA II/III is sensitive to parameter setting but can return high-quality solutions (if the population size is large enough) which are nondominated but not quite diverse and require relatively high computational power. The ALP can return “good enough” and relatively diverse solutions.

Backup Slides

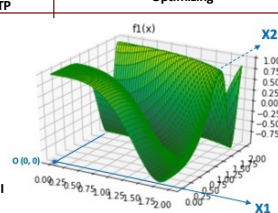
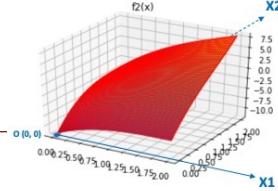
39

This is Toy Problem II. We have two objectives in the optimization formulation, so we have two goals in cDSP. The difference is that we set a target value for each goal and minimize the deviation variables which measure the distance between the achieved goal value and the target. Here we use the Archimedean strategy, which is using weight vectors to combine the deviation variables. We can also use Pre-emptive strategy, which is also known as Lexicographic, that is to prioritize the goals and minimize their deviations one by one.

## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 6 The Optimization Model and Compromise DSP of the TP-II

Strategy TP	Optimizing	Satisficing
II		<b>Given</b> $x1, x2, d1^\pm, d2^\pm$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = \frac{1}{2} \cdot (x1 - 2)^3 + (x2 - 2)^3 + x1 \cdot x2^2$
		<b>Find</b> $x1, x2, d1^\mp, d2^\mp$
		<b>Satisfy</b> <b>Goals:</b> $\frac{f_1(x)}{12} + d1^- = 1$ $\frac{f_2(x)}{8} + d2^- = 1$
		<b>Constraints:</b> $x1 \cdot x2 \leq 1$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$
		<b>Bounds:</b> $0 \leq x1, x2 \leq 2$ $0 \leq d1^\pm, d2^\pm \leq 1$
	<b>Minimize</b> $Z = \sum_{i=1}^2 w_i \cdot di^-$	

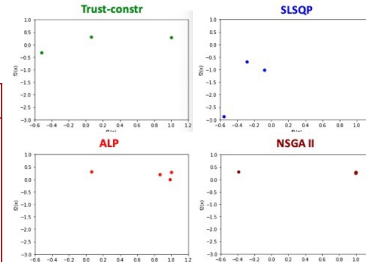


Figure 2. 10 The Solution Points to TP-II on the Objective Space Using Four Algorithms – Solutions returned by Trust-constr and SLSQP are not “good enough,” solutions returned by ALP are “good enough” and diverse, and solutions returned by NSGA II contain nondominated solutions but are not diverse

**Observation:** for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, some optimizing algorithms (e.g. COBYLA) cannot manage the non-convexity, whereas some other optimizing algorithms are easy to converge local optima. NSGA II/III is sensitive to parameter setting but can return high-quality solutions (if the population size is large enough) which are nondominated but not quite diverse and require relatively high computational power. The ALP can return “good enough” and relatively diverse solutions.

Backup Slides

39



If we visualize the two objectives, we can observe that the first objective is nonlinear and nonconvex. When we use the aforementioned methods in both strategies to formulate and solve the problem, using optimization, only the Trust-constr and SLSQP can return feasible solutions but COBYLA cannot return any feasible solution because one of the objectives is non-convex. We visualize the solutions on the objective space. Since we maximize both objectives, the solution points close to the up-right corner are preferred. Using optimization methods, Trust-constr or SLSQP, the solutions are relatively far from the up-right corner, but NSGA II allows us to get one solution close to the up-right corner. Using the ALP, there are several solutions close to the up-right corner.

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
 Lin Guo

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
 UNIVERSITY OF OKLAHOMA

## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

**Table 2.6 The Optimization Model and Compromise DSP of the TP-II**

Strategy TP	Optimizing	Satisficing
II	<b>Objective Functions</b> $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = \frac{1}{2} \cdot (x1 - 2)^3 + (x2 - 2)^3 + x1 \cdot x2^2$	<b>Given</b> $x1, x2, d1^\pm, d2^\pm$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = \frac{1}{2} \cdot (x1 - 2)^3 + (x2 - 2)^3 + x1 \cdot x2^2$
	<b>Constraints and Bounds</b> $s. t. \begin{cases} x1 \cdot x2 \leq 1 \\ 0 \leq x1 \leq 2 \\ 0 \leq x2 \leq 2 \end{cases}$	<b>Find</b> $x1, x2, d1^\mp, d2^\mp$
	<b>Combination of Objective Functions</b> $Max \sum_{i=1}^2 w_i \cdot f_i(x)$	<b>Satisfy</b> $\frac{f_1(x)}{12} + d1^- = 1$ $\frac{f_2(x)}{8} + d2^- = 1$
		<b>Goals:</b> $x1^- \cdot x2 \leq 1$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$
		<b>Bounds:</b> $0 \leq x1, x2 \leq 2$ $0 \leq d1^\pm, d2^\pm \leq 1$
		<b>Minimize</b> $Z = \sum_{i=1}^2 w_i \cdot di^-$

Trust-constr

SLSQP

ALP

NSGA II

**Figure 2.10** The Solution Points to TP-II on the Objective Space. Solutions returned by Trust-constr and SLSQP are not "good enough" and diverse, and solutions returned by ALP are "good enough" and diverse, and solutions returned by NSGA II are nondominated solutions but are far from the up-right corner.

**Observation:** for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, some optimizing algorithms (e.g. COBYLA) cannot manage the non-convexity, whereas some other optimizing algorithms are easy to converge local optima. NSGA II/III is sensitive to parameter setting but can return high-quality solutions (if the population size is large enough) which are nondominated but not quite diverse and require relatively high computational power. The ALP can return "good enough" and relatively diverse solutions.

Backup Slides

39

If we visualize the solution points obtained using the ALP and NSGA II in 3D space,  $x_1 x_2 f(x)$  – plane, the red dots are ALP solutions, the dark red ones are NSGA II solutions. They are very close. Our observation is that ALP allows us to obtain good enough and relatively diverse solutions. When I say “diverse,” I mean that in a good enough small area, there are more solutions based on difference scenarios (weight vectors). This can give us more alternatives in engineering design, so we prefer diverse solutions to single point solution.



## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

- Why does the ALP manage non-convex problems and return solutions close to the nondominated solutions (the solutions returned by NSGA II/III)?
- Two mechanisms of the ALP allow it to linearize the non-convex function relatively accurately and converge with good enough solutions.
  - **First**, using “second-order sequential linearization.”
  - **Second**, using “accumulated constraints.”

Backup Slides

40

So, why does ALP manage non-convex problems and return solutions close to the nondominated solutions (and even more diverse)? Because of the two mechanisms in the ALP – using “second-order sequential linearization” and using “accumulated constraints.” Let me introduce how the two mechanisms allows us to manage non-convex problems in detail.



## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

- **First**, using “second-order sequential linearization.”

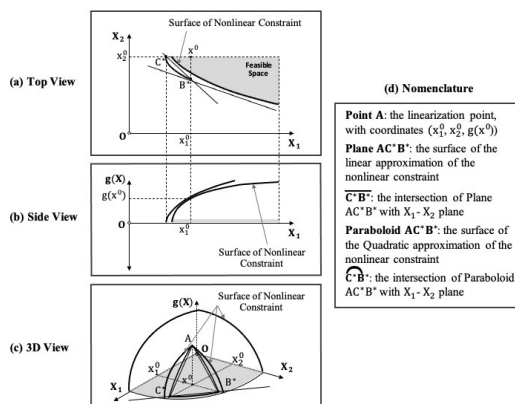


Figure 2.12 Illustration of the Sequential Linearization using the ALP with Different Views When the Quadratic Approximated Paraboloid Has Real Roots

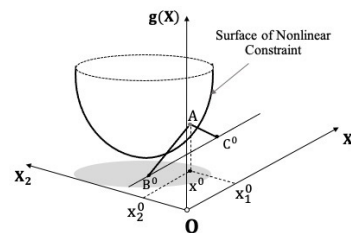


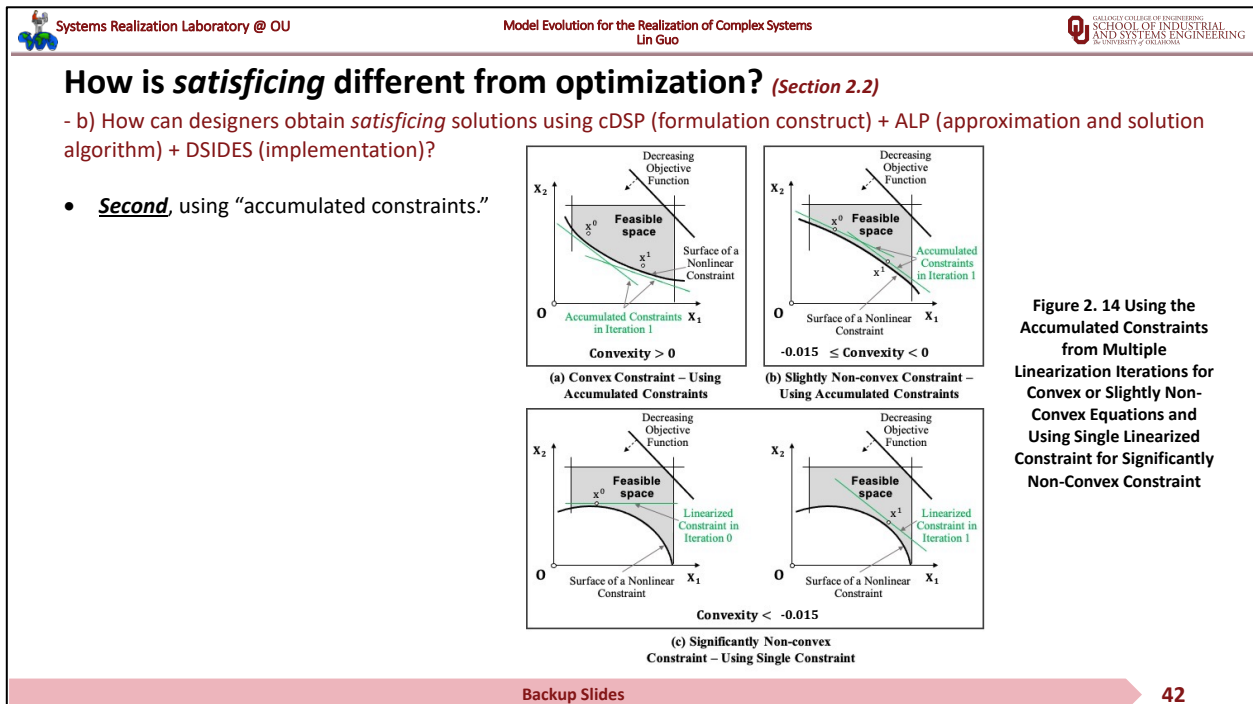
Figure 2.13 Linearization using the ALP When the

Backup Slides

41

The second-order sequential linearization was proposed by Dr. Farrokh Mistree in his 1881 paper. The surface of a non-linear equation is first approximated to a parabola at the starting A (used as the linearization point) and then linearized to a plane, AB\*C\*. The coefficients of the parabola

are the diagonal terms of the Hessian matrix at the starting point. We only use the diagonal terms of the Hessian matrix instead of all terms because the second-order partial derivatives degenerate quickly, especially for engineering-design problems. After we get the parabola, we obtain its two real roots,  $B^*$  and  $C^*$ , which are the end points of the intersection segment between the paraboloid and the  $x_1 - x_2$  plane. The plane passing through the starting point  $A$  and the two real roots  $B^*$  and  $C^*$  is the surface of the linearized constraint. Essentially, using the diagonal terms is projecting the equation onto each dimension and then approximate them. Why is this good? Because in this way, we can have a balance between computational complexity and approximation accuracy. If the parabola has no real roots, like the one in the right picture, we directly linearized the constraint into a plane using the first-order derivative at the starting point.



The second mechanism is using “accumulated constraints.” This allows us to manage non-convex problems. If a constraint is slightly convex or slightly concave, we use “accumulated constraints” to replace it. In the first iteration, we choose a starting point and linearize the equation at it. In the second iteration, we choose a new starting point and linearize the equation at it. If the convexity degree at the new starting point is greater than or equal to  $-0.015$ , we use the linearized constraint in the previous iteration together with the linearized constraint in the current iteration to replace the original nonlinear constraint. However, if a constraint is highly non-convex, like the one in the bottom picture, we do not accumulate the linearized constraints from multiple iteration, because if we do so, we will cut off a big chunk of the feasible area. The highly non-convex constraints often exist in engineering-design problems. We use difference ways to linearize the constraints with different convexity degree. That is how we manage non-convex problems.



## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 1 The Advantages of Realizing *Satisficing* Strategy using cDSP and ALP implemented on DSIDES

Stage	Feature	Advantage	Introduction and Discussion
Formulation	Using Goals and Minimizing Deviation Variables Instead of Objectives	At a solution point, only the necessary KKT conditions are met, whereas the sufficient KKT conditions do not have to be met. Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and unparameterizable uncertainties.	Section 2.2.7
TP-II Approximation	Using second-order sequential linearization Using accumulated linearization	Designers can have a balance between linearization accuracy and computational complexity. Designers can manage nonconvex problems in a way, and deal with highly convex, nonlinear problems relatively more accurately.	Section 2.2.5 and 5.2.1
Exploration	Combining interior-point searching and vertex searching	Designers can avoid being stuck into local optimum to some extent and identify <i>satisficing</i> solutions relatively insensitive to starting points changing.	Section 2.2.3
Evaluation	Allowing some violations of soft requirements, such as the bounds of deviation variables	Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility. As a result, goals and constraints with different scale can be managed	Section 2.2.9

Backup Slides

43

In summary, using Toy Problem II, I demonstrate how the two mechanisms in the approximation using the ALP allow designers to have a balance between linearization accuracy and computational complexity, and deal with the non-convex problems relatively accurately.



## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 8 The Optimization Model and Compromise DSP of the TP-III

Strategy TP	Optimizing	Satisficing
III	<p><b>Objective Functions</b></p> $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><b>Constraints and Bounds</b></p> $\begin{cases} x1 \cdot x2 \leq 1 \\ f_1(x) \geq 0 \\ f_2(x) \geq 0 \\ 0 \leq x1 \leq 2 \\ 0 \leq x2 \leq 2 \end{cases}$ <p><b>Combination of Objective Functions</b></p> $\text{Max} \sum_{i=1}^2 w_i \cdot f_i(x)$	<p><b>Given</b></p> $x1, x2, d1^\pm, d2^\pm$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><b>Find</b></p> $x1, x2, d1^\mp, d2^\mp$ <p><b>Satisfy Goals:</b></p> $\frac{f_1(x)}{12} + d1^- = 1$ $\frac{f_2(x)}{400} + d2^- = 1$ <p><b>Constraints:</b></p> $\begin{cases} x1 \cdot x2 \leq 1 \\ f_1(x) \geq 0 \\ f_2(x) \geq 0 \\ d_i^- \cdot d_i^+ = 0, i = 1, 2 \\ 0 \leq x1, x2 \leq 2 \\ 0 \leq d1^\pm, d2^\pm \leq 1 \end{cases}$ <p><b>Minimize</b></p> $Z = \sum_{i=1}^2 w_i \cdot (d_i^- + d_i^+)$ <p><b>Bounds:</b></p> $\begin{cases} 0 \leq x1, x2 \leq 2 \\ 0 \leq d1^\pm, d2^\pm \leq 1 \end{cases}$ <p><b>Minimize</b></p> $Z = \sum_{i=1}^2 w_i \cdot d_i^-$

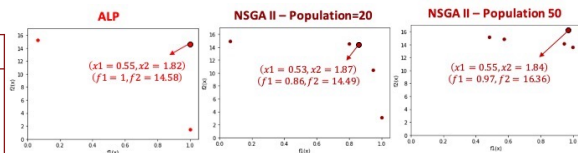


Figure 2. 16 The Solution Points to TP-III on the Objective Space Using Two Algorithms – Solutions returned by NSGA II are closer to the nondominated solution and more diverse but sensitive to parameter setting and require higher computational power.

**Observation:** for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, and the scale of the objectives varies largely, why some optimizing algorithms (e.g. COBYLA) cannot work it out? The answer is given as follows.

Backup Slides

44

Then, I use another toy problem to demonstrate how we deal with another challenge in engineering design, that is when the goals have units with different scales.

## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 8 The Optimization Model and Compromise DSP of the TP-III

Strategy	Optimizing	Satisficing
TP		<p><b>Given</b></p> $x_1, x_2, d_1^\pm, d_2^\pm$ $f_1(x) = \cos(x_1^2 + x_2^3)$ $f_2(x) = 25 \cdot (x_1 - 2)^3 + 50 \cdot (x_2 - 2)^3 + 50 \cdot x_1 \cdot x_2^2$ <p><b>Find</b></p> $x_1, x_2, d_1^\mp, d_2^\mp$ <p><b>Satisfy</b></p> <p><b>Goals:</b></p> $\frac{f_1(x)}{1.2} + d_1^- = 1$ $\frac{f_2(x)}{400} + d_2^- = 1$ <p><b>Constraints:</b></p> $x_1 \cdot x_2 \leq 1$ $f_1(x) \geq 0$ $f_2(x) \geq 0$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><b>Bounds:</b></p> $0 \leq x_1, x_2 \leq 2$ $0 \leq d_1^\pm, d_2^\pm \leq 1$ <p><b>Minimize</b></p> $Z = \sum_{i=1}^2 w_i \cdot (d_i^- + d_i^+)$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><b>Bounds:</b></p> $0 \leq x_1, x_2 \leq 2$ $0 \leq d_1^\pm, d_2^\pm \leq 1$ <p><b>Minimize</b></p> $Z = \sum_{i=1}^2 w_i \cdot d_i^-$

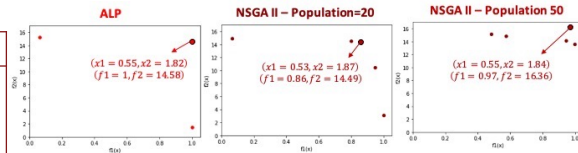


Figure 2. 16 The Solution Points to TP-III on the Objective Space Using Two Algorithms – Solutions returned by NSGA II are closer to the nondominated solution and more diverse but sensitive to parameter setting and require higher computational power.

**Observation:** for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, and the scale of the objectives varies largely, why some optimizing algorithms (e.g. COBYLA) cannot work it out? The answer is given as follows.

Backup Slides

44

One goal is from -0.75 to 1, whereas the other is from -600 to 400. So, the two goals have very different units. This often takes place in engineering problems. Again, we use the methods in the two strategies. This time, no methods in optimizing strategy can return any feasible solution but NSGA II. We choose NSGA II as a verification method to evaluate whether the solutions identified using the selected methods are good enough.

For this problem, when we set population as 20 for NSGA II, the returned solutions are not quite close to the up-right corner. They are not better than the ALP solutions. But when we set population as 50 for NSGA II, the NSGA II solutions are closer to the up-right corner comparing with the ALP solutions.



## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 8 The Optimization Model and Compromise DSP of the TP-III

Strategy TP	Optimizing	Satisficing
III	<p><b>Objective Functions</b></p> $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><b>Constraints and Bounds</b></p> $s.t. \begin{cases} x1 \cdot x2 \leq 1 \\ f_1(x) \geq 0 \\ f_2(x) \geq 0 \\ 0 \leq x1 \leq 2 \\ 0 \leq x2 \leq 2 \end{cases}$ <p><b>Combination of Objective Functions</b></p> $Max \sum_{i=1}^2 w_i \cdot f_i(x)$	<p><b>Given</b></p> $x1, x2, d1^\pm, d2^\pm$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><b>Find</b></p> $x1, x2, d1^\mp, d2^\mp$ <p><b>Satisfy Goals:</b></p> $\frac{f_1(x)}{1.2} + d1^- = 1$ $\frac{f_2(x)}{400} + d2^- = 1$ <p><b>Constraints:</b></p> $x1 \cdot x2 \leq 1$ $f_1(x) \geq 0$ $f_2(x) \geq 0$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><b>Bounds:</b></p> $0 \leq x1, x2 \leq 2$ $0 \leq d1^\pm, d2^\pm \leq 1$ <p><b>Minimize</b></p> $Z = \sum_{i=1}^2 w_i \cdot (d_i^- + d_i^+)$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$ <p><b>Bounds:</b></p> $0 \leq x1, x2 \leq 2$ $0 \leq d1^\pm, d2^\pm \leq 1$ <p><b>Minimize</b></p> $Z = \sum_{i=1}^2 w_i \cdot d_i^-$

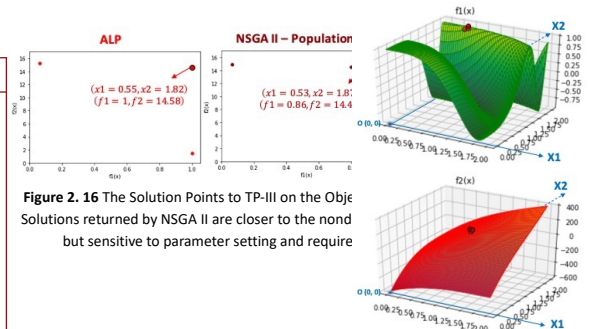


Figure 2.16 The Solution Points to TP-III on the Objective Functions returned by NSGA II are closer to the nondominated front but sensitive to parameter setting and require

**Observation:** for a multi-objective (multi-goal) problem with nonlinear, non-convex functions, and the scale of the objectives varies largely, why some optimizing algorithms (e.g. COBYLA) cannot work it out? The answer is given as follows.

Backup Slides

44

If we visualize them into the 3D space, we observe that all of the solutions close to the up-right corner, either from the ALP or from NSGA II, are concentrated into a small, good enough area. Our observation is "for a multi-objective problem with nonlinear, non-convex objectives, and the scale of the objectives varies largely, most optimization algorithms cannot work it out, whilst some others depend heavily on the hyperparameter setting. Why is that?"



## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

- Using cDSP, designers minimize the deviation variables that measure the distance between the real achieved value of a goal and the target of the goal.

Strategy TP	Optimizing	Satisficing
III	<p><b>Objective Functions</b></p> $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><b>Combination of Objective Functions</b></p> $Max \sum_{i=1}^2 w_i \cdot f_i(x)$	<p><b>Given</b></p> $x1, x2, d1^\pm, d2^\pm$ $f_1(x) = \cos(x1^2 + x2^3)$ $f_2(x) = 25 \cdot (x1 - 2)^3 + 50 \cdot (x2 - 2)^3 + 50 \cdot x1 \cdot x2^2$ <p><b>Goals:</b></p> $\frac{f_1(x)}{1.2} + d1^- = 1$ $\frac{f_2(x)}{400} + d2^- = 1$ <p><b>Minimize</b></p> $Z = \sum_{i=1}^2 w_i \cdot (d_i^- + d_i^+)$ $d_i^- \cdot d_i^+ = 0, i = 1, 2$

- So, the Lagrange equation does not depend on decision variables X:
 
$$\frac{\partial L(X, \mu, \lambda)}{\partial X} \equiv 0$$
- So, the variation of the decision variables does not affect the feasibility of a solution.
- The goal does not dominate one another in completion (or achievement, or fulfillment) due to the difference in the actual achieved value:

$$\frac{f_i(x)}{T_i} \approx \frac{f_j(x)}{T_j}, \text{ given that } T_i \gg T_j$$

Backup Slides

45

The answer is that using cDSP, designers minimize the deviation variables that measure the distance between the real achieved value of a goal and the target of the goal. In optimization

formulation, the objective is the linear combination of the multiple objectives, so the objective is consisted of decision variables. In satisficing formulation, the cDSP, the objective is consisted of deviation variables instead of decision variables. So, using satisficing, the Lagrange equation does not depend on decision variables, so the variation in decision variables does not affect the feasibility of the solution. In addition, the goals do not dominate one another even when their unit scales are different, because if the target value of Goal  $i$  and Goal  $j$  varies a lot, the completion of Goal  $i$  and Goal  $j$  is within the same scale.

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF UTAH

### How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Optimizing

Satisficing

The first-order Lagrange equations is a function of parameters  $\mathcal{P}$ , decision variables  $x$ , and Lagrange multipliers  $\mu, \lambda$

$\nabla_x L(x, \mu, \lambda) = \psi(\mathcal{P}, x, \mu, \lambda)$

The second-order Lagrange equation is a functions of parameters  $\mathcal{P}$  and decision variables  $x$

$\nabla_{xx}^2 L(x, \mu, \lambda) = \nabla_x \psi(\mathcal{P}, x, \mu, \lambda) = \psi'(\mathcal{P}, x)$

If any uncertainty with probability  $P$  takes place (to at least one item of  $\mathcal{P}$  or  $x$  or  $\mu$  or  $\lambda$ ), the probability  $\mathbb{P}$  that an optimal solution is still optimal is

$\mathbb{P}(x^*|P) \approx \prod_{q=1}^Q [1 - p(\bar{\mathcal{P}}_q|P)] \prod_{n=1}^N [1 - P(\bar{x}_n|P)] \prod_{i=1}^m [1 - P(\bar{\mu}_i|P)] \prod_{j=1}^l [1 - P(\bar{\lambda}_j|P)]$

The first-order Lagrange equation is a function of the coefficients  $p$  of the deviation variables  $d$  in the goal function

$\nabla_d L(x^s, d, \mu, \lambda) = \nabla_d z(d) + \sum_{i=1}^m \mu_i \nabla_d g_i(x^s) - \sum_{j=1}^l \lambda_j \nabla_d h_j(x^s) - \lambda_{p+1} \nabla_d G(x^s, d)$   
 $= \nabla_d z(d) + \mathbf{0} - \mathbf{0} \pm \mathbf{1} = \nabla_d z(d) \pm \mathbf{1} = \psi(p) \pm \mathbf{1}$

The second-order Lagrange equation degenerates to zero

$\nabla_{dd}^2 L(x^s, d, \mu, \lambda) = \nabla_{dd}^2 z(d) \equiv \mathbf{0}$

If any uncertainty with probability  $P$  takes place (to  $p$ ), the probability  $\mathbb{P}$  that a satisficing solution is still satisficing is

$\mathbb{P}(x^s|P) \approx \prod_{r=1}^R [1 - p(\bar{\mathcal{P}}_r|P)]$

$\Rightarrow \mathbb{P}(x^*|P) \ll \mathbb{P}(x^s|P)$

Backup Slides
46

Let me use the KKT conditions to explain it more thoroughly.

For optimization, the first-order derivative of Lagrange equation with respect to decision variable  $x$  is a function of the parameters of the model (the coefficients in objectives and constraints), decision variables (if any objective or constraint is nonlinear), and the Lagrange multipliers. For satisficing, the first-order derivative of Lagrange equation should be with respect to the deviation variables because only deviation variables show up in the objective function (no decision variables), and it is only consisted of the coefficients of the deviation variables in the objective function. If we use Archimedean (weight vectors) format to combine the goals, then such coefficients left in the first-order Lagrange equation would be the weights.

For optimization, for the second-order derivative of Lagrange equation with respect to decision variables, it may still have some parameters and decision variables left in the equation because of the nonlinearity. As to satisficing, the second-order derivative of Lagrange equation with respect to deviation variables degenerates to zero because the objective of a cDSP is a linear combination of deviation variables. That's why using satisficing strategy, we do not need to meet the second-order KKT conditions – the second order equation degenerates.

As we know that both optimizing and satisficing requires to meet the first-order or necessary KKT conditions. Here we can see that the chance of maintaining the first-order KKT conditions for optimizing and satisficing varies. If any uncertainty with probability  $P$  takes place to any item in the first-order equation, it may or may not bring variation to the item. If the uncertainty takes

place to parameters that breaks the balance of the first-order Lagrange equation, we denote it as  $p(\mathcal{P}_q|\mathbf{P})$ . And the same thing with decision variables and Lagrange multipliers, we denote the probability of the uncertainty happens to them not breaking the first-order Lagrange equation as  $P(\tilde{\mathbf{x}}_n|\mathbf{P})$ ,  $P(\tilde{\boldsymbol{\mu}}_i|\mathbf{P})$ , and  $P(\tilde{\boldsymbol{\lambda}}_j|\mathbf{P})$ . If all of the items under the uncertainty do not vary and break the first-order equation, then then optimal solution is still optimal under this uncertainty. The probability of it can be represented as the equation in the left bottom. We all know that any probability or “one minus this probability” is a value in the range of  $[0, 1]$ , so the more items on the right-hand side we multiply, the lower the value on the left-hand side is. For satisficing strategy, if any uncertainty with probability  $P$  takes place (to  $p$ ), the probability that a satisficing solution is still satisficing is only by multiplying the probability that the uncertainty does not bring variation to the coefficients of the deviation variables in the objective that break the balance of the first-order Lagrange equation. The multipliers in the bottom-right equations are way fewer than the multipliers in the bottom-left equations. Therefore, the chance of maintaining an optimal solution under uncertainties is often smaller than the chance of maintaining a satisficing solution under the same uncertainties.

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
GALILEI COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

### How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 1 The Advantages of Realizing *Satisficing* Strategy using cDSP and ALP implemented on DSIDES

Stage	Feature	Advantage	Introduction and Discussion
<b>TP-III</b>			
<b>Formulation</b>	Using Goals and Minimizing Deviation Variables Instead of Objectives	At a solution point, only the necessary KKT conditions are met, whereas the sufficient KKT conditions do not have to be met. Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and unparameterizable uncertainties.	Section 2.2.7
<b>Approximation</b>	Using second-order sequential linearization Using accumulated linearization	Designers can have a balance between linearization accuracy and computational complexity. Designers can manage nonconvex problems in a way, and deal with highly convex, nonlinear problems relatively more accurately.	Section 2.2.5 and 5.2.1
<b>Exploration</b>	Combining interior-point searching and vertex searching	Designers can avoid being stuck into local optimum to some extent and identify satisficing solutions relatively insensitive to starting points changing	Section 2.2.3
<b>Evaluation</b>	Allowing some violations of soft requirements, such as the bounds of deviation variables	Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility. As a result, goals and constraints with different scale can be managed	Section 2.2.3

**Back to Slide 8**  
 How can we obtain satisficing solutions using cDSP+ALP+DSIDES

Backup Slides
47

In summary, in satisficing strategy, the formulation is different from optimization. The advantage is that designers have a higher chance of finding a solution and a lower chance of losing a solution for parameterizable and unparameterizable uncertainties. (The unparameterizable uncertainties are not shown in the equations in the previous slide but they exist.). Again, that is one of the reasons why we choose satisficing strategy to manage engineering-design problems.





## Why do not use NSGA II to design complex systems?

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

First, NSGA II/III **cannot give designers insight** on the nature of the decision model or the possible ways to improve the model. NSGA II/III is an interior searching algorithm, which uses metaheuristics to search for solutions that generationally improve the optimality and diversity of the solutions, but **for information, such as the bottleneck of the model, or the sensitivity of each part of the model, or anything else that may indicate model improvement, cannot be provided** along with the searching.

Second, the performance of NSGA II/III (include convergence speed, optimality of solutions, and diversity of solutions) is **sensitive to hyperparameter setting**. Typical hyperparameters, the population size and generation number, should be predefined by designers. However, designers only have the idea that a larger population size or a larger generation number can return better solutions, but they may not have a clue how large is “good enough.” In Toy Problem II and III, I can show how different the solutions can be when setting the population as 20 and 50 for the same problem.

Third, NSGA II/III **requires much more computational power** than satisficing algorithms such as Linear Programming (ALP) algorithm.

**Back to Slide 8**  
How can we obtain  
satisficing solutions using  
cDSP+ALP+DSIDES

Backup Slides

48

*By the way, why don't we use NSGA II/III to solve engineering-design problems since NSGA II/III performs really well? Here's the answer. First, besides the solutions, designers want to learn more about the model and how we can improve it, but NSGA II/III cannot give such information. Second, NSGA II/III's performance heavily depends on the hyperparameter setting, such as the population size and generation number. Designers only know that a larger population or more generations end up with better solutions, but they do not know how large is “good enough.” Third, NSGA II/III requires much more computational power than the method we use in satisficing strategy. The essence of NSGA II/III is to generate a lot of decedents, choose the elites ones, allow some diversity and mutation, and mate them to produce a huge number of later generations, so on and so forth. It is like the “lower-evolved animals” who reproduce a large number of offspring for natural selection. The computational complexity is high.*

*However, NSGA II/III are beautiful algorithms that can return good enough solutions. So, we use it as a verification method to evaluate whether the solutions obtained by using satisficing strategy are good enough, although we do not fully depend on it to provide all information that we need for engineering designs. (Let's go back to the main slides.)*



## Response to Dr. Trafalis: How is *satisficing* different from optimization?

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Table 2. 1 The Advantages of Realizing *Satisficing* Strategy using cDSP and ALP implemented on DSIDES (Section 2.2, 2.3)

Stage	Feature	Advantage	Introduction and Discussion
Formulation	Using Goals and Minimizing Deviation Variables Instead of Objectives	At a solution point, only the necessary KKT conditions are met, whereas the sufficient KKT conditions do not have to be met. Therefore, designers have a higher chance of finding a solution and a lower chance of losing a solution due to parameterizable and unparameterizable uncertainties.	Section 2.2.7
Approximation	Using second-order sequential linearization	Designers can have a balance between linearization accuracy and computational complexity.	Section 2.2.5 and 5.2.1
	Using accumulated linearization	Designers can manage nonconvex problems in a way, and deal with highly convex, nonlinear problems relatively more accurately.	
Exploration	Combining interior-point searching and vertex searching	Designers can avoid being stuck into local optimum to some extent and identify <i>satisficing</i> solutions relatively insensitive to starting points changing.	Section 2.2.3
Evaluation	Allowing some violations of soft requirements, such as the bounds of deviation variables	Designers can manage rigid requirements and soft requirements in different ways to ensure feasibility. As a result, goals and constraints with different scale can be managed	Section 2.2.9 Go to backup slide 38 for details

Frame of Reference

A Test Problem

Contribution and Way Forward

8 / 28

There are three other toy problems and some discussions illustrating the differences in exploration and evaluation. I'm not going through all of them in this presentation due to the time limit. They are in my Chapter 2. If you have interest, welcome to read it.



## Response to Dr. Trafalis: How is *satisficing* different from optimization?

- c) Why do I choose *satisficing* methods, particularly, cDSP + ALP + DSIDES (implementation) to manage complex-system design?

### Model World



### Physical World



Intellectual Disconnection

We have proved we can manage them better using *satisficing* strategy

www.usaengecon.com

- Nonlinear, discrete, and non-convex
- Fewer factors to control but a lot of requirements
- Need to explore the ways to combine the multiple goals
- Manage multiple types of uncertainty
- Need to know more about the model robustness and ways to improve the model formulation

Frame of Reference

A Test Problem

Contribution and Way Forward

9 / 28

So far we have answered the first two sub-questions – how is *satisficing* different from optimizing and how can we obtain *satisficing* solutions using cDSP+ALP+DSIDES. The third sub-question is “why do I choose cDSP, ALP, and DSIDES to realize *satisficing* strategy?” The answer is a summary of the first two answers. As we observe that there is an intellectual disconnection between

the model world and the physical world, the physical world is usually nonlinear, discrete, non-convex, has fewer factors to control (decision variables) but a lot of requirements to meet (constraints), has uncertainties, and requires us to know more about the model robustness and ways to improve the model formulation, and we have demonstrated that we can manage them using cDSP+ALP+DSIDES by identifying and analyzing satisficing solutions and realize the model evolution.

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

## Research Gaps in both Modeling Strategies – Optimizing and Satisficing (Section 1.4)

### Optimize

So far as the theories of mathematics are about **reality**, they are **not certain**; so far as they are **certain**, they are **not about reality**.  
- Albert Einstein, one of the greatest physicists

Why?

- Satisfying KKT conditions
- No information passing
- Easy to stuck into local optimum
- Hard to compromise goals with different units
- Sensitive to parameter setting
- No mechanism to update metaheuristics

### Satisfice

The decision maker has a choice between an optimal decision from an **imaginary simplified world**, or decisions that are “good enough”, that **satisfice**, for a world **approximating** the complex real one more closely.  
- Herbert Simon, American Economist

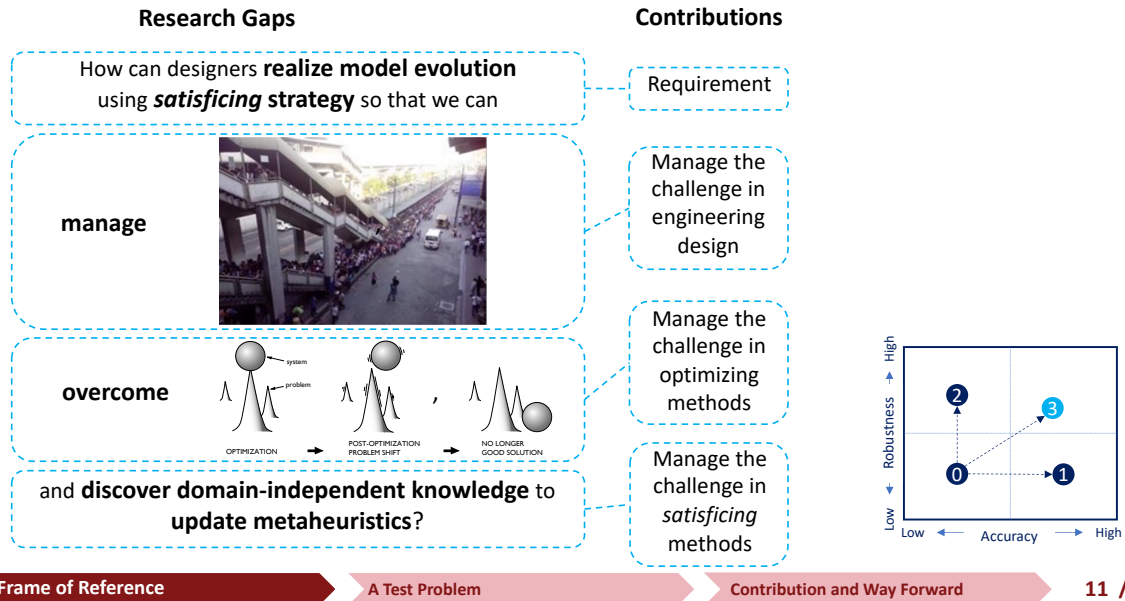
Why?

- No information passing
- Rely on domain expertise
- Rely on metaheuristics to make rules
- No mechanism to update metaheuristics

Frame of Reference
A Test Problem
Contribution and Way Forward
10 / 28

Based on what we have discussed so far, there are research gaps in both strategies. In optimizing strategy, as Albert Einstein said, “So far as the theories of mathematics are about reality, they are not certain; so far as they are certain, they are not about reality.” Using optimizing strategy is like to improve the accuracy of the solutions (to approach the optimal) without improving their robustness to uncertainties. On the contrary, in satisficing, as Herbert Simon said, “The decision maker has a choice between an optimal decision from an imaginary simplified world, or decisions that are “good enough”, that satisfice, for a world approximating the complex real one more closely. So, using satisficing strategy is like to improve the robustness of the solutions without improving their accuracy.

## Research Gaps filled in this Dissertation (Section 1.5)



Given the research gaps in both strategies, here is the summary. So far till now is some existing knowledge that I present or summarize in another way. From now on is something new in my dissertation. In summary, the research gap to be filled in this dissertation is “How can designers realize model evolution using satisficing strategy so that we can manage the chaos in the physical world, overcome the risk of losing an optimal solution, and discover domain-independent knowledge to update metaheuristics?” For each of the research gaps, we have a potential contribution. We will meet the requirements in complex-systems designing, manage the challenge in engineering design, manage the challenge in optimizing methods, and manage the challenge in satisficing methods. Our goal is to improve the accuracy and robustness of our design simultaneously through model evolution.



## Overview of Today's Presentation

### Addressing the Committee's Questions

- **Dr. Trafalis:** how is your method different from optimization? (Section 1.2, 1.4, 2.1, 2.2, 2.3)
- **Dr. Nicholson:** (for Chapter 6), how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors? (Section 6.14, 6.3)
- **Dr. Neeson:** how does the Red River project help or relevant to your future work? (Section 9.2.2, 9.2.3)

### Frame of Reference

- Characteristics and challenges in designing complex systems (Section 1.1, 1.3)
- Modeling strategies – optimizing and *satisficing* – and their problems and differences (Section 1.2, 1.4, 2.2, 2.3)
- Research gaps (Section 1.5)

### A Test Problem

- Dissertation layout (Section 1.7)
- Research Questions and Hypotheses
  - Research Question 3: What is the method to speed up learning the system nature, such as the interrelationship among the subsystems? (Section 2.4.2)
  - Specific Hypothesis 3: Learning interrelationship among subsystems using unsupervised learning and reorganize them based on it. (Section 2.5)
  - Method 3: Adaptive Leveling-Weighting-Clustering algorithm (Section 3.3.3)
  - Test Problem 3: Rankine cycle thermal system design (Chapter 6)

### Contribution and Way forward

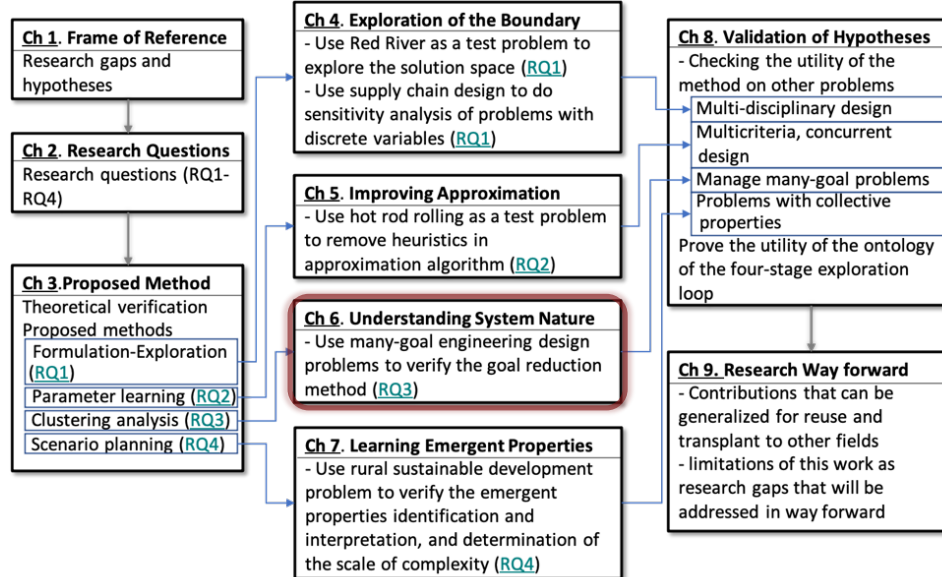
- Answering the research questions (Section 8.1.3)
- Managing four types of uncertainties (Section 8.1.4)
- Verification and Validation (Section 1.6, 9.1.3)
- Relevant publications (Section 9.1.4)
- Research thrusts and application in my early career (Section 9.2)

12 / 28

Next, I will introduce my dissertation layout, research questions, and hypotheses, along the way answer Dr. Nicholson's question.



## Dissertation Layout



Frame of Reference

A Test Problem

Contribution and Way Forward

13 / 28

There are nine chapters in my dissertation. In Chapter 1, 2, and 3, I frame the reference, pose research questions, and propose methods. From Chapter 4 to 7, I use test problems to demonstrate the internal consistency which is the correctness of the proposed methods. In Chapter 8, I validate the methods, which is to demonstrate the utility and application scope of the methods. In Chapter 9, I describe the way forward base on this dissertation, especially the "I statement," which is my

research plan in my early career in academia. Now, let me go deeper into Chapter 6 and answer Dr. Nicholson's question.

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

### Research Questions and Hypotheses

**RQ1** What is the method to evolve model boundary?

**SH1** Explore the sensitivity of the segments of the model boundary and improve accordingly.

**RQ2** What is the method to evolve model to update metaheuristics?

**SH2** Learn, evaluate, and update metaheuristics to improve model performance.

**RQ3** What is the method to speed up learning the system nature?

**SH3** Learn system nature such as interrelationship among subsystems and reorganize them based on it.

**RQ4** What is the method that allows model evolution by incorporating emergent properties?

**SH4** Capture and quantify emergent properties through scenario planning in simulations.

Frame of Reference
A Test Problem
Contribution and Way Forward
14 / 28

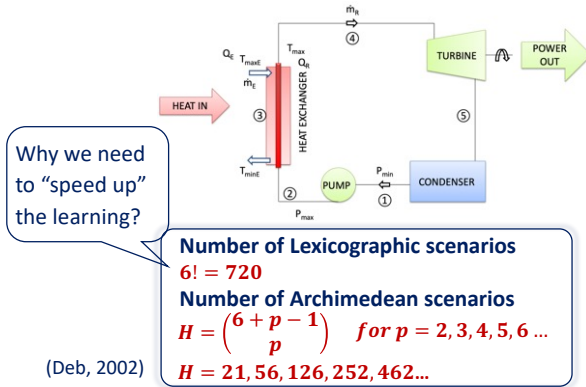
Here are the four research questions in my dissertation. I propose that model evolution is about connecting the different stages in the design evolution loop using different methods and algorithms. When we establish connections, we can pass information among them, evaluate the heuristics used in early stages or iterations, and update the heuristics using the information we learn from later steps. This process is the model evolution.

Each research question is about how I can connect different stages to exchange certain information to evolve the decision model of a complex system. For each Research Question, I have a specific hypothesis to answer it. In Chapter 6 is about Research Question 3, “what is the method to speed up learning the system nature?” The specific hypothesis is “learn system nature such as interrelationship among subsystems and reorganize them based on it.”

## Problem statement

### Managing many-goal problems

Given a *concurrent design* problem, how can we *learn the interrelationship among subsystems* to reorganize the subsystems to boost the system performance?



### Six goals:

- Min Turbine moisture
- Max Rankine cycle efficiency
- Max Temperature increase in exchanger
- Max System efficiency indicator 1
- Max System efficiency indicator 2
- Max Heat transfer effectiveness in exchanger

### Why?

- No information passing
- Rely on domain expertise
- Rely on metaheuristics to make rules
- No mechanism to update metaheuristics

Smith, W. F., Milisavljevic, J., Sabeghi, M., Allen, J. K., and Mistree, F., "The realization of engineered systems with considerations of complexity," Proc. ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V007T006A019-V007T006A019.

Frame of Reference

A Test Problem

Contribution and Way Forward

15 / 28

To answer the research question, I use a test problem, a concurrent engineering-design problem with six goals. We want to boost the system performance by reorganizing the subsystems. This problem is to design a Rankine cycle thermal system. There are a lot of possible applications for small-scale power systems. For example, to provide power to farming equipment. A common approach given a heat source is to build around a Rankine cycle like this. The primary components are a power-producing turbine, which transfers the heat source into motion, a pump to pressurize the fluid to the turbine, and two exchangers – one is a condenser, and the other is a heat exchanger. They make the liquid into vapor or the opposite way to convert energy between different forms. A common fluid used in a Rankine cycle is water. Sometimes, we can also have organic fluids in chemistry, which makes the Rankine cycle an organic Rankine cycle. Our task is to select the fluid and determine the geometry specification of each part of the Rankine cycle.

We have six goals: to minimize the moisture in the turbine, to maximize the Rankine cycle efficiency, to maximize the temperature increase in the heat exchanger, to maximize system efficiency indicator 1 and 2, and to maximize heat transfer effectiveness in exchanger.

Suppose we do not have any domain knowledge on how to combine the six goals, the two most popular methods to enumerate all different scenarios of combining the goals are Lexicographic (or Pre-emptive) and Archimedean (or weighted combination). However, for a six-goal problem, there are 720 Lexicographic scenarios and hundreds of Archimedean scenarios (depends on the value of parameter "p" of the Archimedean scenarios they set). And there will be a lot more if we mix the Lexicographic and Archimedean scenarios. Either way requires huge amount of computation. In the paper where this Rankine cycle problem first published, the authors selected 15 scenarios to explore the combinations of the goals, based on their domain knowledge, but are the 15 scenarios sufficient, and are they the right ones or most representative ones? We don't know and they did not verify. In summary, the authors rely on domain expertise to simplify the problem by picking 15 scenarios to combine the goals. They rely on metaheuristics to make rules, like why they pick up 15 scenarios, why not 20 or 50. And there is no mechanism to update their

metaheuristics – what if the 15 scenarios they use are not the most representative ones, and what if it is too much or too little?

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

### Features and Limitations of the Methods in Literature (Section 6.1)

Method	Gaps
VEGA	Vector valued feedback can be subjective; local optimum
SPEA2	The fitness assignment being used to evolve solutions is with heuristics and may be domain-dependent
MOEA/D	Decomposition of the problem may cause inaccuracies and computational complexity
NSGA-II/III	No guarantee of the even-distribution of the solution on the near-pareto front; no insight on formulation improvement
REDGA	Losing information when removing partially redundant objectives
HypE	Monte Carlo approximation is computational costly
Interval analysis	Aiming at geometry determination of the problem

**Summary of gaps:**

- Aiming at finding near Pareto front, which is consisted of optimal solutions that are sensitive to model errors and variations
- No decision support on how we can use the solutions in different area under various situations
- Relying on domain knowledge to scale multiple objectives or decompose the problem

Go to backup slide 49 for details in critically reviewing the literature

Frame of Reference

A Test Problem

Contribution and Way Forward

16 / 28

Given the problem, we critically review the literature in managing many-objective or many-goal problems. (Go the backup sides.)

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

### Critically Reviewing Literature on Many-Goal Problems (Section 6.1)

Back to Slide 16

Table 6. 2 The Features and Limitations of Some Classic Multi-Objective (Multi-Goal) Solution Algorithms and Methods

Emphases	Algorithm or method	Description	Searching for interior solutions	Having adaptable ways of producing more solutions (offspring)	Giving information of tradeoffs between multiple characteristics (such as solution accuracy vs convergence speed)	Use the knowledge of tradeoffs between goals (objectives) to improve model formulation	Explore multiple forms of goals (objectives) structure, such as decomposing the problem, reducing goals, etc.
Solution algorithms	VEGA (Schaffer 1985)	Using vector-valued feedback with adaptive procedures for searching high-order multi-objective problems		*			*
	SPEA2 (Zitzler, Laumanns et al. 2001)	Using fitness assignment, archiving and truncating (the near-Pareto front) to evolve solutions to approach the Pareto-optimal set	*	*	*		
	MOEA/D (Zhang and Li 2007)	Decomposing a problem into scalar optimization subproblems and optimizing them simultaneously	*	*	*		*
	NSGA-II/III (Deb, Pratap et al. 2002, Seada and Deb 2014)	Using the nondominated sorting evolutionary algorithm to adaptively update reference points to approach the Pareto front	*	*			*
	REDGA (Jaimes, Coello et al. 2009)	Reducing the number of objectives by removing redundant (to some degree) objectives			*	*	*
	HypE (Bader and Zitzler 2011)	Using Monte Carlo simulation to approximate the exact hypervolume values and seeking ranking of solutions	*	*	*		
Design improvement	Multi-Level Decisions (Mistree, Patel et al. 1994)	Using two design strategies and multi-level decisions to foster discussion on multi-objective problems			*	*	
	RCEM (Chen, Allen et al. 1997, Choi, Austin et al. 2005)	Improving the robustness of the design using indices based on the results of exploring the solution space				*	*
	Interval analysis (Hao and Merlet 2005)	Using parallel robots based on interval analysis to determine geometries to satisfy all compulsory requirements				*	*
	Level diagrams (Reynoso-Meza, Blasco et al. 2013)	Comparing multiple Pareto fronts based on different design concepts using level diagrams, to support decision making on design concept selection	*	*			*
	XPLORE in DSIDES (Smith, Milisavljevic et al. 2015, Sabeghi, Shukla et al. 2016)	Exploring the design space by exploring different goal structures using a compromise Decision Support Problem			*	*	*
	CORTHOOG (Warwick 2019)	Removing poor measurement degrees-of-freedom iteratively until pseudo-orthogonality check was optimized	*	*		*	*

Backup Slides

49



Through reviewing more than 60 publications, and the details are in Table 6.2, we summarize that there are three research gaps in the field of managing many-goal, engineering-design problems. (Go back to main slides.)

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

### Features and Limitations of the Methods in Literature *(Section 6.1)*

Method	Gaps
VEGA	Vector valued feedback can be subjective; local optimum
SPEA2	The fitness assignment being used to evolve solutions is with heuristics and may be domain-dependent
MOEA/D	Decomposition of the problem may cause inaccuracies and computational complexity
NSGA-II/III	No guarantee of the even-distribution of the solution on the near-pareto front; no insight on formulation improvement
REDGA	Losing information when removing partially redundant objectives
HypE	Monte Carlo approximation is computational costly
Interval analysis	Aiming at geometry determination of the problem

**Summary of gaps:**

- Aiming at finding near Pareto front, which is consisted of optimal solutions that are sensitive to model errors and variations
- No decision support on how we can use the solutions in different area under various situations
- Relying on domain knowledge to scale multiple objectives or decompose the problem

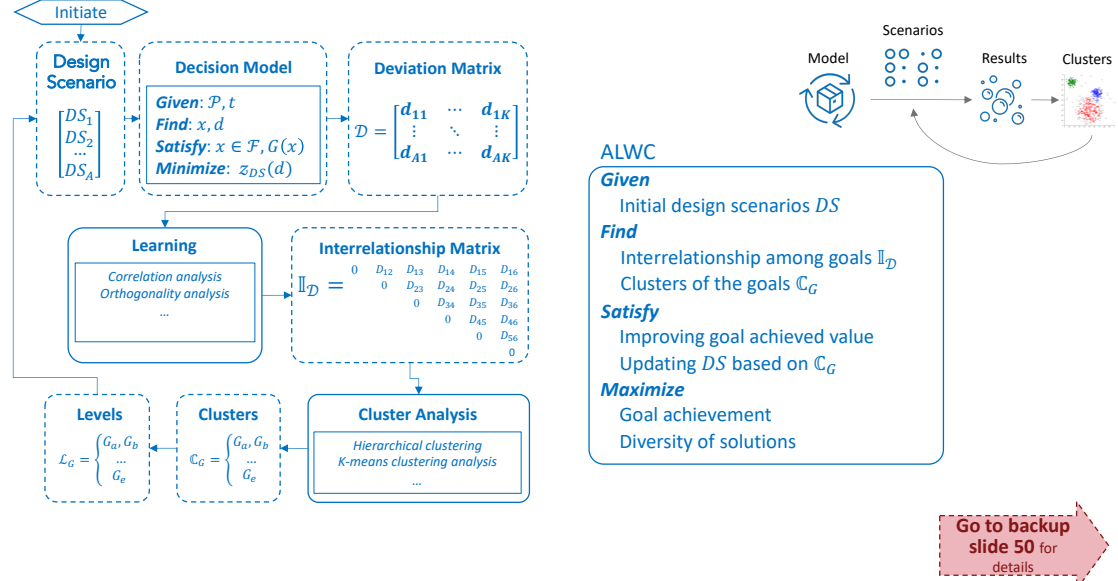
**Go to backup slide 49** for details in critically reviewing the literature

Frame of Reference
A Test Problem
Contribution and Way Forward
16 / 28

The first gap is that most authors aim at finding near Pareto front, which is consisted of optimal solutions that are sensitive to model errors and variations.

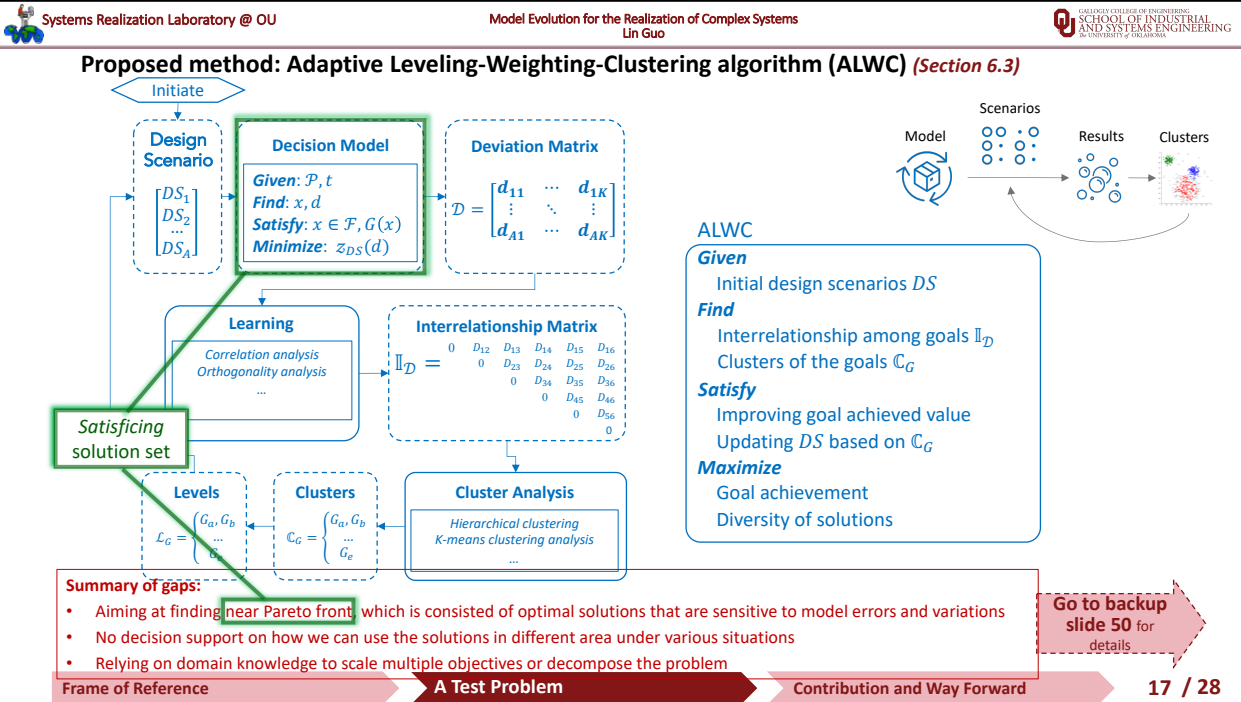
Second, no decision support on how we can use the solutions under different scenarios. They only give a lot of different solutions but without recommendations on when and how designers should use each of them.

Third, they rely on domain knowledge to scale multiple objectives or decompose the problem.

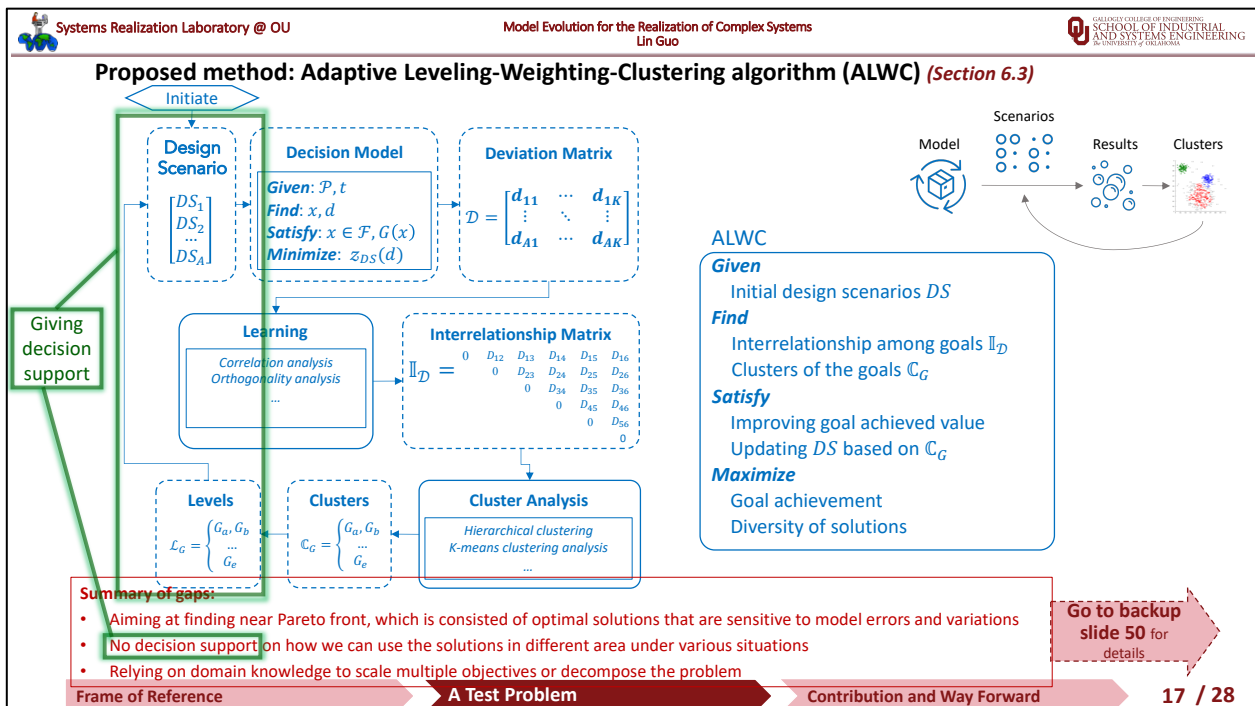
**Proposed method: Adaptive Leveling-Weighting-Clustering algorithm (ALWC) (Section 6.3)**


We propose the Adaptive Leveling-Weighting-Clustering algorithm (ALWC) to fill the research gaps. First, we choose design scenarios – can be weight vectors, or lexicographic scenarios, or mixture of the two. We implement the design scenarios into cDSP and get a deviation matrix. For a  $k$ -goal problem with  $A$  design scenarios, our deviation matrix is a  $A$ - $k$  dimension matrix. Then using the deviation matrix, we process unsupervised learning, which is to learn the correlation or orthogonality among the goals (the columns). Other than the correlation and orthogonality, we can have more types of interrelationship among the goals, but for this test problem, I only use two and they verify each other. (More types of interrelationship can be explored in the way forward.) Then we get the interrelationship matrix, which is an upper triangle matrix. Then we perform cluster analysis based on the interrelationship matrix. We use multiple algorithms and do cross validation to determine the clustering results, based on which, we determine the design scenarios of the next iteration – by setting the leveling (lexicographic) and weighting (Archimedean) of the goals: we set the goals belong to the same cluster in the one level, and weighted combine the goals in the same level. That is the whole procedure of one iteration. With the new design scenarios, we process the whole process again for another iteration, so on and so forth until the cluster results and the deviation matrix reach to a stable status, we converge.

But, how do I fill the aforementioned three research gaps by using the ALWC?

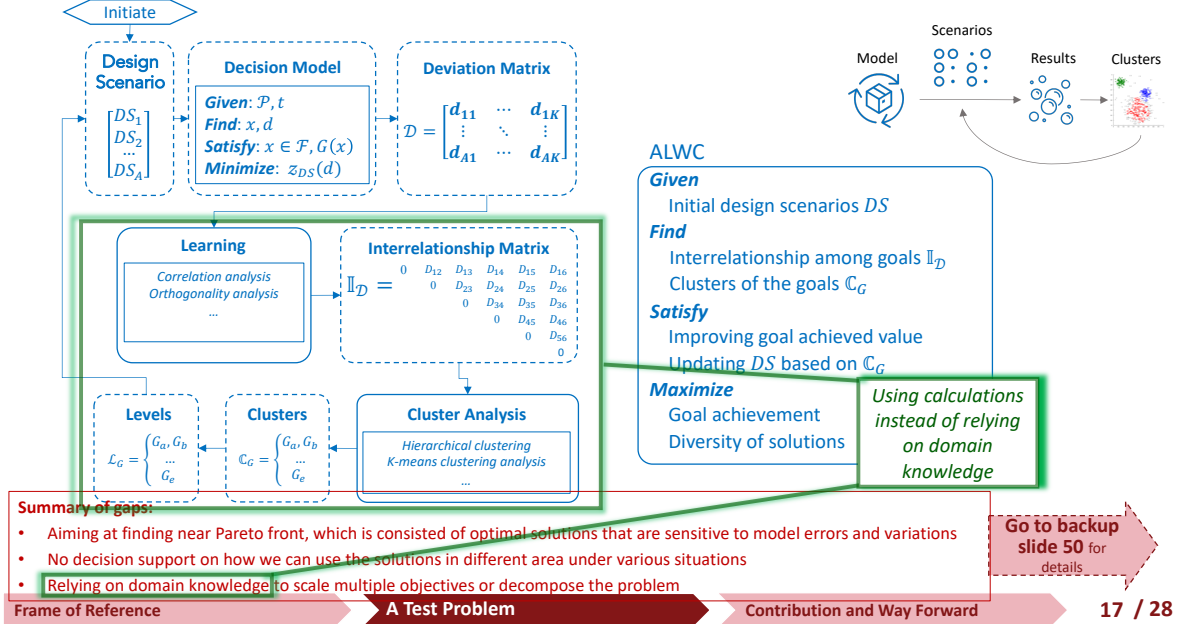


First, instead of finding new Pareto front, which is consisted of single optimal solutions that are sensitive to uncertainties, we search for satisficing solutions set that is relatively insensitive to uncertainties, using satisficing strategy. How? Using cDSP, ALP, and DSIDES.



Second, unlike most literature in this field that does not give decision support on how to choose the solutions, we provide decision support by using the clustering results into the design scenarios of the next iteration.

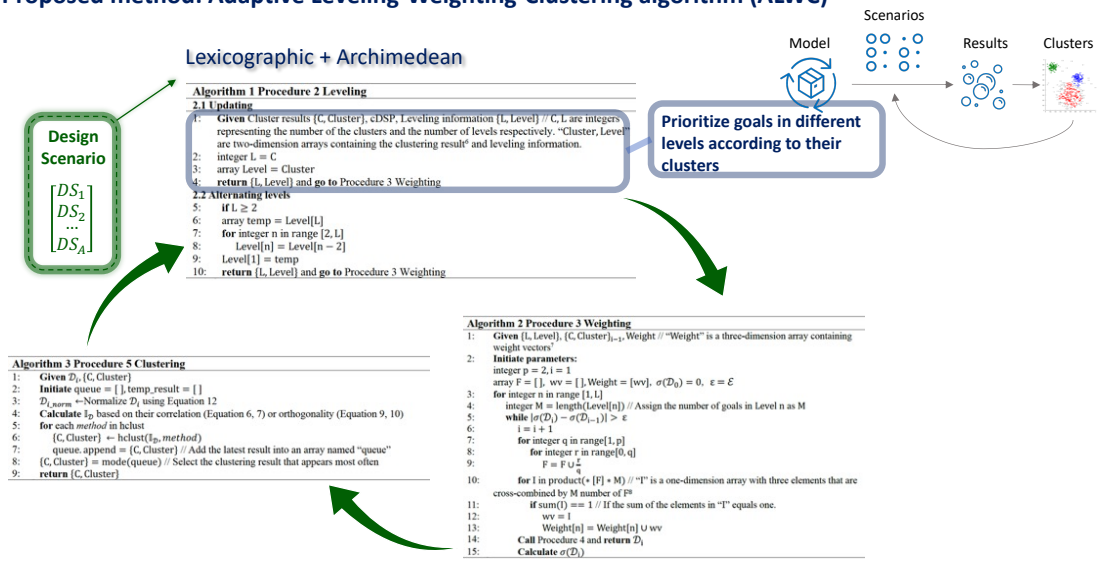
**Proposed method: Adaptive Leveling-Weighting-Clustering algorithm (ALWC) (Section 6.3)**



Third, to avoid relying too much on domain knowledge and in case we do not have sufficient domain knowledge for a new problem, we use calculations to obtain insight, for example, the interrelationships among the goals, to manage the multiple goals.

That's how we fill the three research gaps. Here are the details of how each part of this algorithm works. (Go to backup slides.)

**Proposed method: Adaptive Leveling-Weighting-Clustering algorithm (ALWC)**



In the very first iteration, since we do not have any clustering result yet, the design scenarios are randomly generated. But for any later iteration, given we have a clustering result from the previous

iteration, we prioritize the goals in different levels according to their clusters. The pseudocode in the rounded rectangle enables us to do that.

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

GALATIA COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
UNIVERSITY OF ORADEA

### Proposed method: Adaptive Leveling-Weighting-Clustering algorithm (ALWC)

Design Scenario

$$\begin{bmatrix} DS_1 \\ DS_2 \\ \dots \\ DS_A \end{bmatrix}$$

Lexicographic + Archimedean

**Algorithm 1 Procedure 2 Leveling**

2.1 Updating

- 1: Given Cluster results {C, Cluster}, cDSP, Leveling information {L, Level} // C, L are integers representing the number of the clusters and the number of levels respectively. "Cluster, Level" are two-dimension arrays containing the clustering result and leveling information.
- 2: integer L = C
- 3: array Level = Cluster
- 4: return {L, Level} and go to Procedure 3 Weighting

2.2 Alternating levels

- 5: if L ≥ 2
- 6: array temp = Level[L]
- 7: for integer n in range [2, L]
- 8: Level[n] = Level[n - 2]
- 9: Level[1] = temp
- 10: return {L, Level} and go to Procedure 3 Weighting

Alternate the levels

**Algorithm 2 Procedure 3 Weighting**

- 1: Given {L, Level}, {C, Cluster}, Weight // "Weight" is a three-dimension array containing weight vectors
- 2: Initiate parameters:  
integer p = 2, i = 1  
array F = [], wv = [], Weight = [wv], σ(D<sub>0</sub>) = 0, ε = ε
- 3: for integer n in range [1, L]
- 4: integer M = length(Level[n]) // Assign the number of goals in Level n as M
- 5: while |σ(D<sub>n</sub>) - σ(D<sub>n-1</sub>)| > ε
- 6: i = i + 1
- 7: for integer q in range[1, p]
- 8: for integer r in range[0, q]
- 9: F = F ∪  $\frac{1}{q}$
- 10: for l in product(\* [F] \* M) // "T" is a one-dimension array with three elements that are cross-combined by M number of F<sup>n</sup>
- 11: if sum(T) == 1 // If the sum of the elements in "T" equals one.
- 12: wv = 1
- 13: Weight[n] = Weight[n] ∪ wv
- 14: Call Procedure 4 and return D<sub>i</sub>
- 15: Calculate σ(D<sub>i</sub>)

**Algorithm 3 Procedure 5 Clustering**

- 1: Given D<sub>i</sub>, {C, Cluster}
- 2: Initiate queue = [], temp\_result = []
- 3: D<sub>i\_norm</sub> ← Normalize D<sub>i</sub> using Equation 12
- 4: Calculate I<sub>0</sub> based on their correlation (Equation 6, 7) or orthogonality (Equation 9, 10)
- 5: for each method in hclust
- 6: {C, Cluster} ← hclust(I<sub>0</sub>, method)
- 7: queue.append = {C, Cluster} // Add the latest result into an array named "queue"
- 8: {C, Cluster} = mode(queue) // Select the clustering result that appears most often
- 9: return {C, Cluster}

Scenarios

Model

Results

Clusters

Backup Slides
50

Then we alternate the levels using the pseudocode in this rounded box.

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

GALATIA COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
UNIVERSITY OF ORADEA

### Proposed method: Adaptive Leveling-Weighting-Clustering algorithm (ALWC)

Design Scenario

$$\begin{bmatrix} DS_1 \\ DS_2 \\ \dots \\ DS_A \end{bmatrix}$$

Lexicographic + Archimedean

**Algorithm 1 Procedure 2 Leveling**

2.1 Updating

- 1: Given Cluster results {C, Cluster}, cDSP, Leveling information {L, Level} // C, L are integers representing the number of the clusters and the number of levels respectively. "Cluster, Level" are two-dimension arrays containing the clustering result and leveling information.
- 2: integer L = C
- 3: array Level = Cluster
- 4: return {L, Level} and go to Procedure 3 Weighting

2.2 Alternating levels

- 5: if L ≥ 2
- 6: array temp = Level[L]
- 7: for integer n in range [2, L]
- 8: Level[n] = Level[n - 2]
- 9: Level[1] = temp
- 10: return {L, Level} and go to Procedure 3 Weighting

Assign weights to the goals in each level

**Algorithm 2 Procedure 3 Weighting**

- 1: Given {L, Level}, {C, Cluster}, Weight // "Weight" is a three-dimension array containing weight vectors
- 2: Initiate parameters:  
integer p = 2, i = 1  
array F = [], wv = [], Weight = [wv], σ(D<sub>0</sub>) = 0, ε = ε
- 3: for integer n in range [1, L]
- 4: integer M = length(Level[n]) // Assign the number of goals in Level n as M
- 5: while |σ(D<sub>n</sub>) - σ(D<sub>n-1</sub>)| > ε
- 6: i = i + 1
- 7: for integer q in range[1, p]
- 8: for integer r in range[0, q]
- 9: F = F ∪  $\frac{1}{q}$
- 10: for l in product(\* [F] \* M) // "T" is a one-dimension array with three elements that are cross-combined by M number of F<sup>n</sup>
- 11: if sum(T) == 1 // If the sum of the elements in "T" equals one.
- 12: wv = 1
- 13: Weight[n] = Weight[n] ∪ wv
- 14: Call Procedure 4 and return D<sub>i</sub>
- 15: Calculate σ(D<sub>i</sub>)

**Algorithm 3 Procedure 5 Clustering**

- 1: Given D<sub>i</sub>, {C, Cluster}
- 2: Initiate queue = [], temp\_result = []
- 3: D<sub>i\_norm</sub> ← Normalize D<sub>i</sub> using Equation 12
- 4: Calculate I<sub>0</sub> based on their correlation (Equation 6, 7) or orthogonality (Equation 9, 10)
- 5: for each method in hclust
- 6: {C, Cluster} ← hclust(I<sub>0</sub>, method)
- 7: queue.append = {C, Cluster} // Add the latest result into an array named "queue"
- 8: {C, Cluster} = mode(queue) // Select the clustering result that appears most often
- 9: return {C, Cluster}

Scenarios

Model

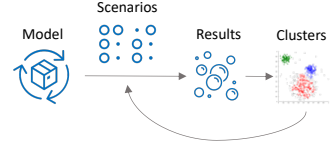
Results

Clusters

Backup Slides
50

Then we generate weights and assign them to the goals in each level using the pseudocode in this rounded box.

### Proposed method: Adaptive Leveling-Weighting-Clustering algorithm (ALWC)



**Lexicographic + Archimedean**

**Design Scenario**

$$\begin{bmatrix} DS_1 \\ DS_2 \\ \dots \\ DS_A \end{bmatrix}$$

```

Algorithm 1 Procedure 2 Leveling
2.1 Updating
1: Given Cluster results {C, Cluster}, cDSP, Leveling information {L, Level} // C, L are integers representing the number of the clusters and the number of levels respectively. "Cluster, Level" are two-dimension arrays containing the clustering result* and leveling information.
2: integer L = C;
3: array Level = Cluster;
4: return {L, Level} and go to Procedure 3 Weighting
2.2 Alternating levels
5: if L ≥ 2
6: array temp = Level[L];
7: for integer n in range [2, L]
8: Level[n] = Level[n - 2];
9: Level[1] = temp;
10: return {L, Level} and go to Procedure 3 Weighting
  
```

```

Algorithm 3 Procedure 5 Clustering
1: Given Di, {C, Cluster}
2: Initiate queue = [], temp_result = []
3: Di_norm ← Normalize Di using Equation 12
4: Calculate Iij based on their correlation (Equation 6, 7) or orthogonality (Equation 9, 10)
5: for each method in list
6: {C, Cluster} ← heclust(Iij, method)
7: queue.append = {C, Cluster} // Add the latest result into an array named "queue"
8: {C, Cluster} ← mode(queue) // Select the clustering result that appears most often
9: return {C, Cluster}
  
```

Cluster the goals based on their deviations in multiple design scenarios (levels X weights)

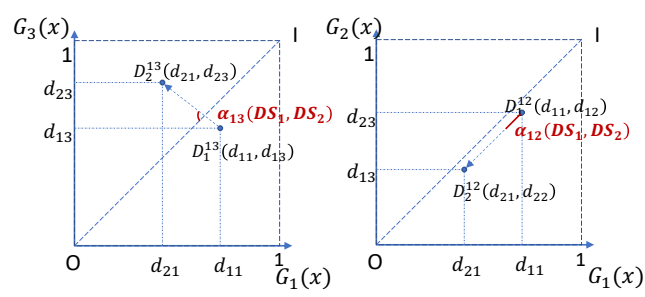
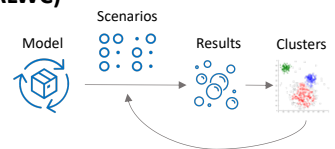
```

Algorithm 2 Procedure 3 Weighting
1: Given {L, Level}, {C, Cluster}, Weight // "Weight" is a three-dimension array containing weight vectors
2: Initiate parameters:
integer p = 2, i = 1
array F = [], wv = [], Weight = [wv], σ(D0) = 0, ε = ε
3: for integer n in range [1, L]
4: integer M = length(Level[n]) // Assign the number of goals in Level n as M
5: while |σ(Di) - σ(Di-1)| > ε
6: i = i + 1
7: for integer q in range [1, p]
8: for integer r in range [0, q]
9: F = F ∪ Diqr
10: for i in product({F} × M) // "I" is a one-dimension array with three elements that are cross-combined by M number of Pq
11: if sum(I) == 1 // If the sum of the elements in "I" equals one.
12: wv = I
13: Weight[n] = Weight[n] ∪ wv
14: Call Procedure 4 and return Di
15: Calculate σ(Di)
  
```

After that, we cluster the goals using their deviations returned by running the multiple scenarios implemented in the cDSP, using the pseudocode in this rounded box.

We run this in iterations until solutions and clustering results do not change much, we converge.


### Proposed method: Adaptive Leveling-Weighting-Clustering algorithm (ALWC)



Back to Slide 18

Here we illustrate two different cases of the interrelationship between two goals. If by changing the design scenario, the deviation of the two goals move parallel with Line  $OI$ , the diagonal, we define the two goals are highly correlated, as it is shown in the right-hand side picture. If by changing the design scenario, the deviation of the two goals move orthogonal to Line  $OI$ , we define

the two goals are more orthogonal, as it is shown in the left-hand side picture. In a single iteration, if by changing a lot of design scenario, two goals are more orthogonal than correlated, it is highly likely that they will be grouped into different clusters. (Go back to main slides.)

 Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
 Lin Guo

OHSU SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
 A UNIVERSITY OF OREGON

### Results and outcome

	Result 1	Result 2	Result 3	...	Result 7	...	Result 18
<b>Level 1</b>	Goal 2, 4	Goal 2, 4	Goal 2, 4	...	Goal 3, 6	...	Goal 1, 5
<b>(Weight)</b>	(1,0)	(0,1)	$(\frac{1}{2}, \frac{1}{2})$	...	(1,0)	...	$(\frac{1}{2}, \frac{1}{2})$
<b>Level 2</b>	Goal 3, 6	Goal 3, 6	Goal 3, 6	...	Goal 2, 4	...	Goal 3, 6
<b>(Weight)</b>	$(\frac{1}{2}, \frac{1}{2})$	$(\frac{1}{2}, \frac{1}{2})$	$(\frac{1}{2}, \frac{1}{2})$	...	$(\frac{1}{2}, \frac{1}{2})$	...	$(\frac{1}{2}, \frac{1}{2})$
<b>Level 3</b>	Goal 1, 5	Goal 1, 5	Goal 1, 5	...	Goal 1, 5	...	Goal 2, 4
<b>(Weight)</b>	$(\frac{1}{2}, \frac{1}{2})$	$(\frac{1}{2}, \frac{1}{2})$	$(\frac{1}{2}, \frac{1}{2})$	...	$(\frac{1}{2}, \frac{1}{2})$	...	$(\frac{1}{2}, \frac{1}{2})$

$d_{norm.1.j}$	0.24	0.24	0.24	1.00	0.00
$d_{norm.2.j}$	0.00	0.00	0.00	0.73	0.09
$d_{norm.3.j}$	1.00	1.00	1.00	0.02	0.99
$d_{norm.4.j}$	0.00	0.00	0.00	0.73	0.09
$d_{norm.5.j}$	1.00	1.00	1.00	0.00	1.00
$d_{norm.6.j}$	0.14	0.14	0.14	0.00	0.14

	Meaning	Goals
<b>Cluster 1</b>	Rankine cycle efficiency	Goal 2 Goal 4
<b>Cluster 2</b>	Moisture in turbine + Heat exchanger efficiency	Goal 1 Goal 6
<b>Cluster 3</b>	Temperature increase	Goal 3 Goal 5

Frame of Reference

A Test Problem

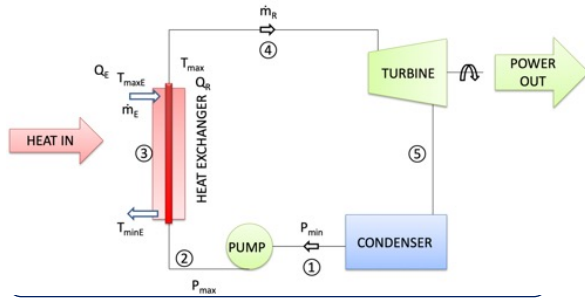
Contribution and Way Forward

18 / 28

Using the ALWC, we get the converged deviation matrix and we normalize it to ensure every row is ranged from 0 to 1, as the one in the rounded box. Its clustering result is shown in the table on the right-hand side. In the early iterations, we have different clustering results, but in the last a few iterations, they do not change any more.



### Results and outcome



Cluster	Meaning	Goals
Cluster 1	Rankine cycle efficiency	Goal 2 Goal 4
Cluster 2	Moisture in turbine + Heat exchanger efficiency	Goal 1 Goal 6
Cluster 3	Temperature increase	Goal 3 Goal 5

Frame of Reference

A Test Problem

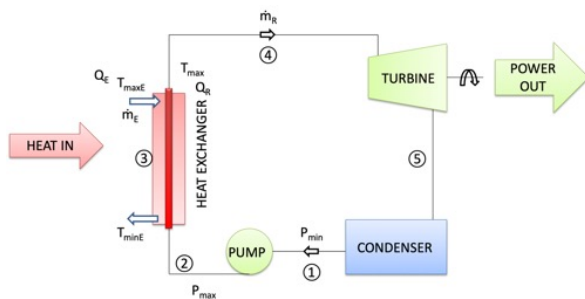
Contribution and Way Forward

18 / 28

*We observe that the result is not that each of the four components forms a cluster. So, unsupervised learning may return us something that is not quite in line with our domain knowledge that we thought should be right. This indicates that for a problem that we do have domain expertise, such expertise can be wrong or misleading. So, we cannot fully rely on domain knowledge even when we have it.*



### Results and outcome



Cluster	Meaning	Goals
Cluster 1	Rankine cycle efficiency	Goal 2 Goal 4
Cluster 2	Moisture in turbine + Heat exchanger efficiency	Goal 1 Goal 6
Cluster 3	Temperature increase	Goal 3 Goal 5

Number of Lexicographic-Archimedean scenarios

$$H' = C! \cdot \binom{x_1 + p - 1}{p}, \text{ for } C = 3, x_1 = 2, p = 2$$

$$H' = 36 \ll 720 \text{ or } 462$$

Frame of Reference

A Test Problem

Contribution and Way Forward

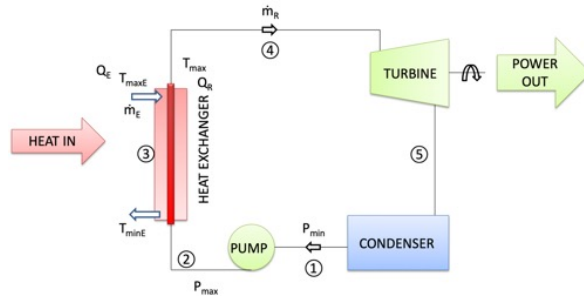
18 / 28

*One of the contributions is that we significantly save the computing power by reducing the number of design scenarios – 36 is way smaller than 720 for Lexicographic and 462 for Archimedean.*





## Results and outcome

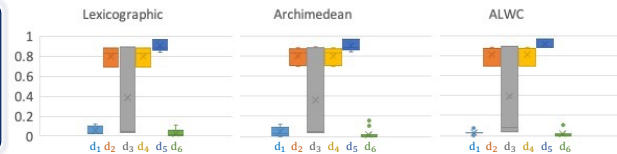


	Meaning	Goals
Cluster 1	Rankine cycle efficiency	Goal 2 Goal 4
Cluster 2	Moisture in turbine + Heat exchanger efficiency	Goal 1 Goal 6
Cluster 3	Temperature increase	Goal 3 Goal 5

Number of Lexicographic-Archimedean scenarios

$$H' = C! \cdot \binom{x_1 + p - 1}{p}, \text{ for } C = 3, x_1 = 2, p = 2$$

$$H' = 36 \ll 720 \text{ or } 462$$



Frame of Reference

A Test Problem

Contribution and Way Forward

18 / 28

Another contribution is that the 36 scenarios can get us even better results than enumerating the Lexicographic or Archimedean strategy of combining the goals. As we plot the deviations using Lexicographic, Archimedean, and ALWC respectively in the bottom right, we want to minimize the deviations and minimize the variance of the deviations under all design scenarios, so we desire the bars close to the horizontal axis and as short as possible. We observe that using the ALWC, the deviation of Goal 1 and Goal 5 is improved regarding the value and distribution, comparing with both Lexicographic and Archimedean strategy. The deviation of Goal 6 is similar to Archimedean but better than Lexicographic strategy.

In summary, using the ALWC, we not only reduce the computational complexity, but also achieve the goals better.



**Response to Dr. Nicholson: (for Chapter 6), how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors? (Section 6.14, 6.3)**

- a) Using a large number of design scenarios to manage the goals in a lot of different ways – “enrich the sample size” and avoid sucking into local optimum through exploitation.
- b) If we run a large number of iterations, and in each iteration, we use a different group of design scenarios – “remove correlation” and avoid fake convergence through exploitation.
- c) Inclined to use the design scenarios that are more representative – “increase the frequency and weight of the more representative design scenarios” and avoid local optimum through exploration.

*So now is a good time to answer Dr. Nicholson’s question, “how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors?” We did three things to ensure we obtain the right result. First, we use a large number of design scenarios to combine the goals – not as many as enumerating all the scenarios but large enough, which is to “enrich the sample size” and avoid sucking into local optimum through exploitation. Second, we run a large number of iterations, which is to “remove correlation” and avoid fake convergence through exploitation. Third, we use design scenarios that are more representative, that are design scenarios based on the clustering result, which is to “increase the frequency and weight of the more representative design scenarios” and avoid local optimum through exploration. So, those are the three ways we use to avoid using a wrong interrelationship.*



### Overview of Today's Presentation

#### Addressing the Committee's Questions

- **Dr. Trafalis:** how is your method different from optimization? (Section 1.2, 1.4, 2.1, 2.2, 2.3)
- **Dr. Nicholson:** (for Chapter 6), how do you know that using the proposed method, we learn the correlation among the goals instead of the weight vectors? (Section 6.14, 6.3)
- **Dr. Neeson:** how does the Red River project help or relevant to your future work? (Section 9.2.2, 9.2.3)

#### Frame of Reference

- Characteristics and challenges in designing complex systems (Section 1.1, 1.3)
- Modeling strategies – optimizing and *satisficing* – and their problems and differences (Section 1.2, 1.4, 2.2, 2.3)
- Research gaps (Section 1.5)

#### A Test Problem

- Dissertation layout (Section 1.7)
- Research Questions and Hypotheses
  - Research Question 3: What is the method to speed up learning the system nature, such as the interrelationship among the subsystems? (Section 2.4.2)
  - Specific Hypothesis 3: Learning interrelationship among subsystems using unsupervised learning and reorganize them based on it. (Section 2.5)
  - Method 3: Adaptive Leveling-Weighting-Clustering algorithm (Section 3.3.3)
  - Test Problem 3: Rankine cycle thermal system design (Chapter 6)

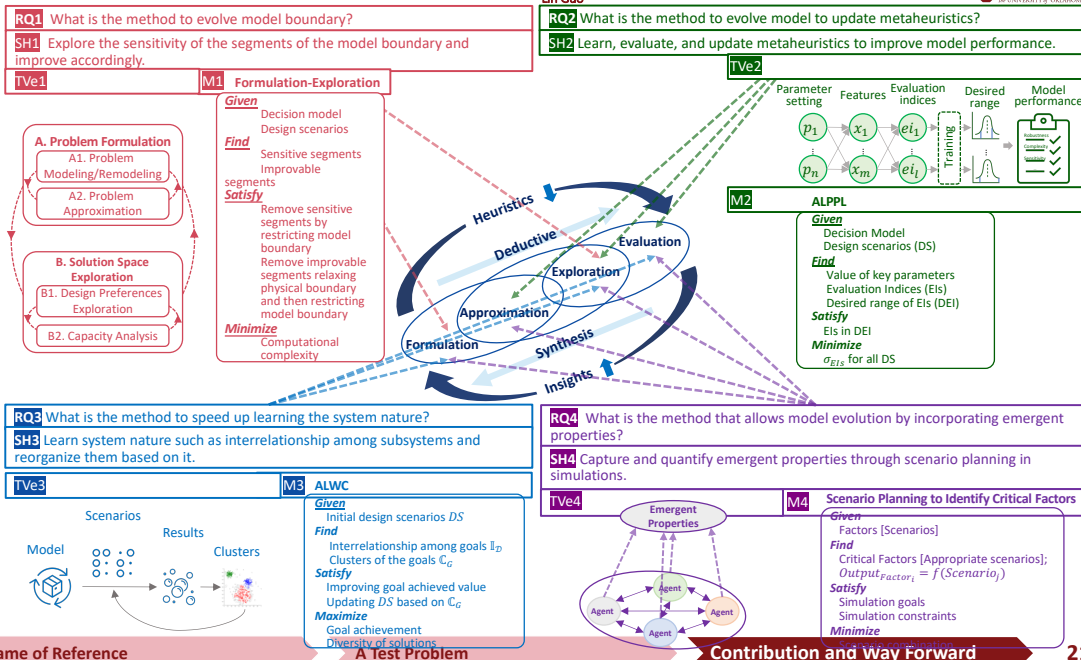
#### Contribution and Way forward

- Answering the research questions (Section 8.1.3)
- Managing four types of uncertainties (Section 8.1.4)
- Verification and Validation (Section 1.6, 9.1.3)
- Relevant publications (Section 9.1.4)
- Research thrusts and application in my early career (Section 9.2)

Next, let me move on to summarize the contribution and way forward and answer Dr. Neeson's question.

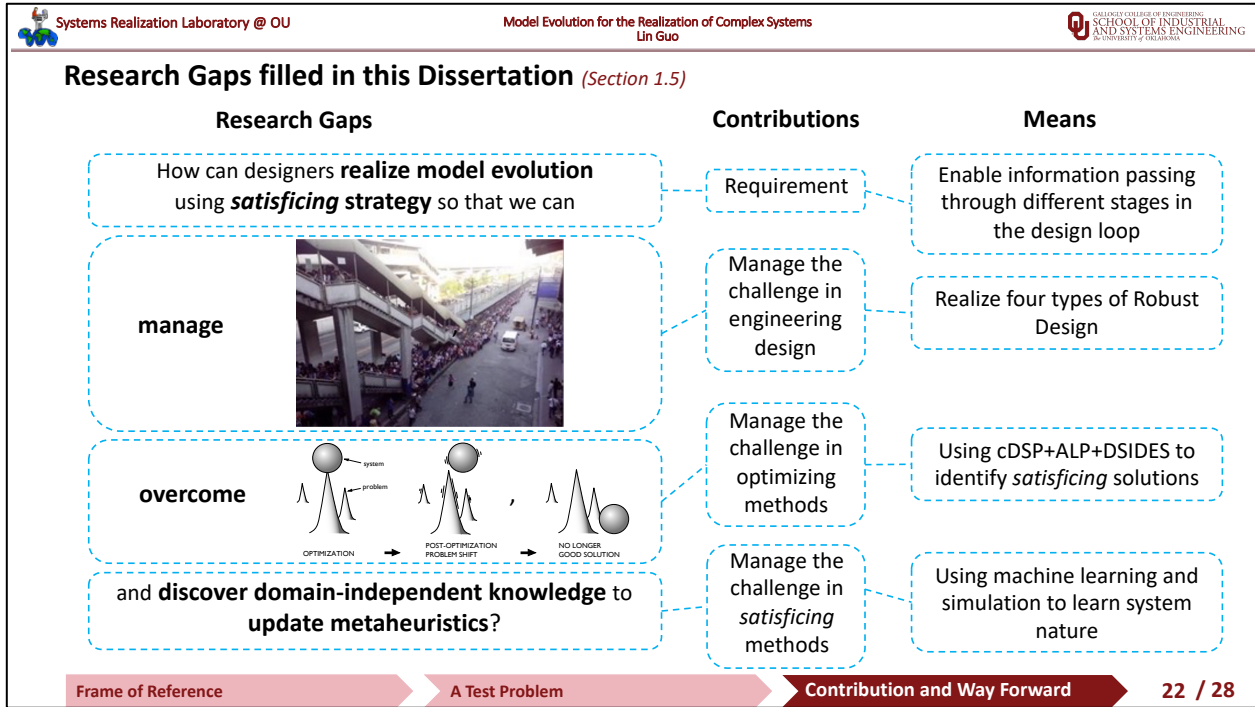


Answer the Research Questions



For each of the research questions, I propose a method and use one or two test problems to use the method. First, what is the method to evolve model boundary? I propose the formulation-exploration framework to identify the sensitive segment and the bottleneck of the model, and treat it by adding buffer or exploit the potential of the mathematical model in the next iteration. Second, what is the method to evolve model to update metaheuristics? I propose to use parameter learning

to identify the features and build evaluation indices that reflect the association between the parameter setting and the approximation performance, and train the evaluation indices to fall a desired range. By doing this, we ensure the parameter setting returns satisficing results. The third research question is what I presented a minute ago. Fourth, what is the method that allows model evolution by incorporating emergent properties? to speed up learning the system nature? I propose the scenario planning in agent-based model, using which, we capture the emergent properties and quantify them for modeling. (The details for Method 1, 2, and 4 are in my backup slides. Due to the time limit, I am not presenting them today. If you have interest, welcome to read my backup slides or dissertation.)



Here are the means to fill the research gaps.

## Manage Four Types of Uncertainties (Section 2.4.1, 3.4.2)

**Type I** – noise factors. *Noise factors* are not under a designer’s control

(Taguchi, 1980)

**Type II** – design variables

(Chen, Allen, and Mistree 1996)

**Type III** – variations in the *mathematical models*

(Choi, Allen, and Mistree 2005)

**Type IV** – variability introduced by a *hierarchical, multiscale or multidisciplinary formulation of the product*.

(Seepersad, Allen, and Mistree 2005)

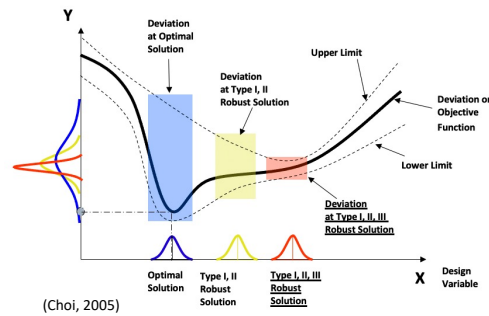


Figure 2.26 Four Types of Robust Solution

Frame of Reference

A Test Problem

Contribution and Way Forward

23 / 28

Another contribution is managing four types of uncertainty. This is also the answer to Dr. Trafalis’s question, what do you mean by “robust solutions?” In this dissertation, we define robust solutions are solutions that are relatively insensitive to one or more of the four types of uncertainty.

## Manage Four Types of Uncertainties (Section 2.4.1, 3.4.2)

Method	M1: Formulation-Exploration Framework		M2: Adaptive Linear Programming with Parameter Learning (ALPPL)	M3: Adaptive Leveling-Weighting-Clustering Algorithm (ALWC)	M4: Scenario Planning in Agent-Based Modeling
Chapter	Ch 4		Ch 5	Ch 6	Ch 7
Test Problem	T1.1: Dam network	T1.2: Supply chain	T2: Hot rolling process chain	T3: Thermal system	T4: Promoting second-season farming
Uncertainty					
Type I	Uncertainty in timing and amount of inflow (precipitation) and outflow demand (user demand)	Uncertainty in demand side (order fluctuation)	Uncertainty in hyper parameter setting (Parameters in approximation algorithm)	Uncertainty in parameter setting in solution algorithm (Starting point of searching)	Uncertainty in price (Price of agriculture products)
Type II	Uncertainty in outflow (water release)	Uncertainty in supply side (productivity fluctuation)	Uncertainty in user preferences		Promotion effort and timing
Type III			Uncertainty in model approximation due to heuristics in approximation	Uncertainty in model approximation (ways of combining multiple goals)	
Type IV				Uncertainty in using domain knowledge to simplify the model (fixing decision variables and selecting design scenarios)	Interventions that change the mathematical relation among promotion and result (developing local market)

Frame of Reference

A Test Problem

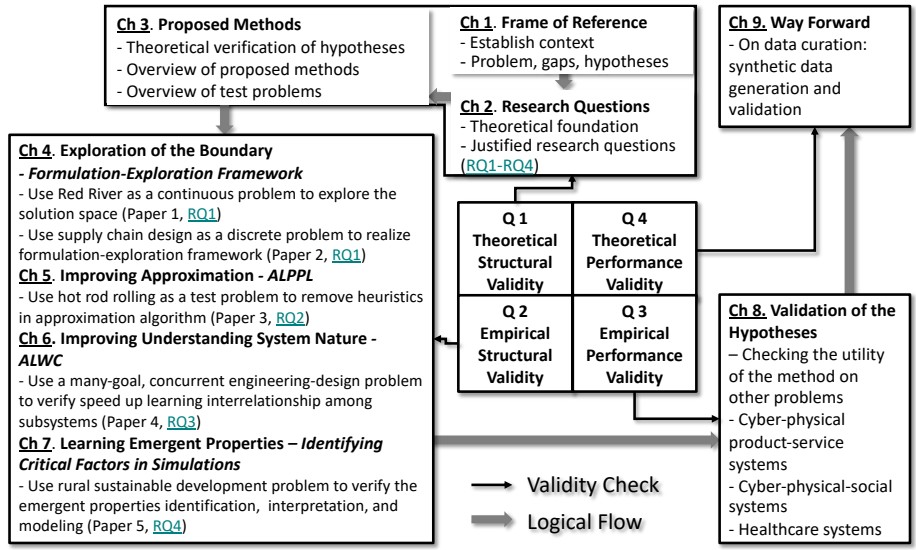
Contribution and Way Forward

23 / 28

Here are the four test problems and their uncertainties that are managed in this dissertation.



### Verification and Validation (Section 1.6, 9.1.3)



Frame of Reference      A Test Problem      Contribution and Way Forward      24 / 28

How do I know the proposed methods are correct and useful? I use validation square to illustrate the verification and validation. (This is the verification and validation of the whole dissertation, not for a particular test problem.) In the first three chapters, I give theoretical structural validity by framing the reference, posing research questions, and proposing methods and demonstrate they are theoretically validated. From Chapter 4 to 7, I use test problems to empirically validate the methods. In Chapter 8, I identify the utility and application scope of the methods. In Chapter 9, I show the theoretical performance validity by indicating the way forward.



### Relevant Publications and Manuscripts under Review

Theories	Applications	Publications and Manuscripts
Multicriteria decision making	Dam-network design	Guo, L., Chen, S., Allen, J.K., Mistree, F., 2020, "A Framework for Designing the Customer Order Decoupling Point to Facilitate Mass Customization," <i>ASME Journal of Mechanical Design</i> , 143(2): 022002. Guo, L., Zamanisabzi, H., Neeson, T.M., Allen, J.K., Mistree, F., 2018, "Managing Conflicting Water Resource Goals and Uncertainties in a Dam-Network by Exploring the Solution Space," <i>ASME Design Automation Conference</i> , Quebec City, Quebec, Canada. Paper Number DETC2018-86018.
	Supply chain design	Guo, L., Chen, S., Allen, J.K., Mistree, F., 2020, "A Framework for Designing the Customer Order Decoupling Point to Facilitate Mass Customization," <i>ASME Journal of Mechanical Design</i> , 143(2): 022002. Guo, L., Chen, S., Allen, J.K., Mistree, F., 2019, "Designing the Customer Order Decoupling Point to Facilitate Mass Customization," <i>ASME Design Automation Conference</i> , Anaheim, CA, USA. Paper Number DETC2019-97379.
Improving algorithm performance using Machine Learning	Improving approximation algorithms	Guo, L., Nellippallil, A.B., Smith, W.F., Allen, J.K., Mistree, F., 2020, "Adaptive Linear Programming Algorithm with Parameter Learning," <i>ASME Design Automation Conference</i> , Online. Paper Number DETC2020-22602. Guo, L., Nellippallil, A.B., Smith, W.F., Allen, J.K., Mistree, F., 2021, "A Smart Linear Programming Algorithm," <i>ASME Journal of Mechanical Design</i> , Paper Number MD-21-1436, <i>under review</i> .
	Managing interactions among subsystems	Guo, L., Milisavljevic-Syed, J., Wang, R., Huang Y., Allen, J.K., Mistree, F., 2021 "Managing Many-Goal, Concurrent Design Problems using Adaptive Leveling-Weighting-Clustering Algorithm," <i>Structural and Multidisciplinary Optimization</i> , <i>under review</i> . Wang, R., Guo, L., Huang, Y., Wang, G., 2021, "Decision Guidance Method for the Knowledge Discovery and Reuse in Many-Goal Engineering Design Problems," <i>Advanced Engineering Informatics</i> , <i>under review</i> .
Multiscale simulation	Designing smart communities	Guo, L., Mohebbi, S., Das, A., Allen, J.K., Mistree, F., 2020, "A Framework for the Exploration of Critical Factors on Promoting Two Season Cultivation in India," <i>ASME Journal of Mechanical Design</i> , 142(12): 124503.
	Knowledge management on cyber-physical systems	Wang, R., Milisavljevic-Syed, J., Guo, L., Huang, Y., & Wang, G., 2021, "Knowledge-Based Design Guidance System for Cloud-Based Decision Support in the Design of Complex Engineered Systems," <i>ASME Journal of Mechanical Design</i> , 143(7), 072001.

Frame of Reference      A Test Problem      Contribution and Way Forward      25 / 28

Here are the fields of my theoretical study, relevant application fields, and publications from this dissertation. The black ones are published. The gray ones are under review.

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
 GATEWAY COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
DU-UNIVERSITY OF OKLAHOMA

## Outline of the Proposed Monograph to Springer

**Abstract**  
In the design of complex systems, to overcome the lack of data, heuristics are used. In this dissertation, a design evolution loop – formulation, approximation, exploration, and evaluation – is proposed. It is hypothesized that by running the design evolution loop iteratively, a designer can improve the design by

- Evaluating the heuristics and replacing them with insight obtained from exploration of the solution space.
- Managing the complexity and uncertainty of module structure.
- Interpreting the behavior and the property to support decision-making.
- Capturing and managing emergent properties.

The design evolution loop is verified using test problems that span various disciplines, for example, supply chains (ISE), hot rod rolling (ME), water resource management (geography), rural development (social science), and parameter learning (data science).

**Chapter 1 Establishing Context**  
Designing complex systems | *Satisficing* strategy | Four types of robust design | Model evolution

**Chapter 2 Evolving the Model Boundary**  
The formulation-exploration framework | Managing water resource | Designing a supply chain

**Chapter 3 Improving the Approximation**  
Approximation algorithms | Using parameter learning | Designing the hot rod rolling process

**Chapter 4 Learning Subsystems**  
Concurrent design | Many-goal problems | Using unsupervised learning | Designing a thermal system

**Chapter 5 Capturing Emergent Properties**  
Cyber-physical-social systems | Scenario planning in simulations | Capturing and quantifying emergent properties

**Chapter 6 Closing Remarks**  
Contributions | More applications of model evolution in different disciplines

Frame of Reference
A Test Problem
Contribution and Way Forward
26 / 28

And I am writing a monograph with my advisors to Springer. Here is the outline of the monograph.

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
 GATEWAY COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
DU-UNIVERSITY OF OKLAHOMA

## Summary of Contributions

1. Theoretical foundation
  - a) Prove the utility of *satisficing* methods in complex-system design (Section 2.1)
  - b) Demonstrate how cDSP+ALP+DSIDES realize *satisficing* strategy (Section 2.2)
2. Four types of robust design (Section 2.4.1, 8.1.4)
  - a) Type I – robust to noise factors (uncertainty in water inflow of a dam network, order fluctuation in a supply chain, etc.)
  - b) Type II – robust to control factors (productivity fluctuation in a supply chain, effort and timing of a promotion, etc.)
  - c) Type III – robust to variation in model structure (uncertainty in model approximation)
  - d) Type IV – robust to uncertainties brought by managing the first three types of uncertainty (uncertainty in model simplification, mathematical abstraction of the physical world, etc.)
3. New Knowledge
  - a) A method allows evolving model boundary – formulation-exploration framework (Section 3.3.1, 4.2.3, 4.3.3)
  - b) A method allows heuristics updating in approximation – Adaptive Linear Programming algorithm with Parameter Learning (ALPPL) (Section 3.3.2, 5.3)
  - c) A method facilitates subsystems awareness and reorganization – Adaptive Leveling-Weighting-Clustering algorithm (ALWC)
  - d) A method helps capture emergent property – scenario planning in simulations (Section 3.3.4, 7.3) (Section 3.3.3, 6.3)
4. Applications in multiple disciplines (Section 8.2, 9.2.3)  
Supply chains (ISE), hot rod rolling (ME), water resource management (geography), rural development (social science), and parameter learning (data science).

Frame of Reference
A Test Problem
Contribution and Way Forward
27 / 28

In summary, the contributions are in four categories.

First, I establish the theoretical foundation by demonstrating the utility of *satisficing* methods in complex-system design and how cDSP+ALP+DSIDES can realize *satisficing* strategy for

designers. Second, I manage four types of uncertainty. Third, the new knowledge in this dissertation comes as four methods. Each of them allows us to answer one research question. Last but not the least, there are applications in multiple disciplines.

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

### Research Thrusts and Application in My Early Career (Section 9.2)

**- Response to Dr. Neeson: how does the Red River project help or relevant to your future work?**

Research Thrust	Applications	Foundation
What is the mathematics that supports the directed evolution of the data curation methods and processes?	Designing lean process chains to support fail-safe healthcare networks and cyber-physical product-service systems	Chapter 4
How can we improve algorithms by replacing heuristics with insight and automate the process?	Realizing the customization of decision workflows for cyber-physical product-service systems (CPPSS).	Chapter 4 and 7
	Design fail-safe supply networks for healthcare systems	Chapter 4 and 6
What are the mechanisms and modeling strategies to support information sharing between multi-scale simulations?	Managing emergent properties of self-organizing systems	Chapter 7

Frame of Reference

A Test Problem

Contribution and Way Forward

28 / 28

Now, let me answer Dr. Neeson's question, "how does the Red River project help or relevant to your future work?" Here are my research thrusts and relevant applications in my early career. They are all based on my dissertation. The Red River problem is in Chapter 4. I identify the directed evolution of the data curation methods and algorithm improvement through replacing heuristics as my future research that can be a step forward of the Red River project. The applications include fail-safe healthcare networks and cyber physical product service systems.





## Acknowledgements



Dr. Janet K. Allen  
 Dr. Farrokh Mistree  
 Dr. Theodore B. Trafalis  
 Dr. Charles D. Nicholson  
 Dr. Thomas M. Neeson  
 Dr. Shivakumar Raman  
 Dr. Warren F. Smith  
 Dr. Anand Balu Nellippallil  
 Dr. Ashok Das  
 Dr. Shima Mohebbi  
 Dr. Ru Wang  
 Dr. Jelena Milisavljevic-Syed  
 And many others



My advisors' chair funds  
 L.A. Comp Chair and the John and Mary Moore Chair at  
 the University of Oklahoma  
 Other financial support from the University of Oklahoma,  
 INFORMS, ASME, and Midwestern Association of  
 Graduate Schools



Cheryl L. Carney  
 Melodi A. Franklin  
 Ashley Herndon  
 My family and friends  
 And many others

29 / 28

*I sincerely thank the people, funds, and organizations who help me in finishing my Ph.D.*



## More Questions from Dr. Trafalis

**TT** Trafalis, Theodore B. Today at 5:07 AM  
 To: Mistree, Farrokh; Cc: Guo, Lin; Allen, Janet K.; Nicholson, Charles D.

You replied to this message on 7/12/21, 7:09 AM. [Show Reply](#)

Lin,

I have read the part related to my questions and I have some extra questions.

1. What are the connections of satisficing solution and robust optimization solution( see formulations of Ben Tal, Bertsimas e.t.c)? I suggest to try a simple toy problem with box constraints.
2. What about Taguchi method and your approach? What about fuzzy optimization ? What about flexible optimization (there is a lot of literature since the seventies)? Can you create a simple toy example and show the resulting solutions?
3. Need to define precisely the concept of robust solution (see other formulations in literature, e.g Bertsimas).
4. In eq. 1.6 what is P calligraphic. It is not defined. In eq. 1.9 what is d and d\*?
5. What do you mean by degree of convexity?
6. Need to discuss more carefully the difference of goal programming solution and satisfying solution (goal programming is strongly related with distance methods in multiobjective optimization and there is a huge literature in this topic).

Backup Slides

30

*Now let me address Dr. Trafalis's new questions from this morning.*

*First, what are the connections of satisficing solutions and robust optimization solutions? The answer is robust optimization solutions are obtained by minimizing (or maximizing) an objective (or multiple objectives) that is consisted of decision variables. When using those methods, people attempt to seek optimal solutions that meet both the necessary and sufficient KKT conditions. We*

have shown the differences between “meeting both necessary and sufficient KKT conditions” and “only meeting the necessary KKT conditions” in Slide 37 and 46:

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

GALATICA COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
UNIVERSITY OF ORLANDO

## How is *satisficing* different from optimization? (Section 2.1)

### Optimizing vs. *Satisficing*

- Difference in the assumptions regarding the KKT conditions

**Second-order sufficient conditions**

For the Lagrangian:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x)$$

$$\Rightarrow s^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) s \geq 0, \text{ where } s \neq 0$$

And

$$[\nabla_x g_i(x^*), \nabla_x h_j(x^*)]^T s = 0$$

**Physical meaning:**

At the solution point  $x^*$ , there exists a nonzero vector  $s$  that is orthogonal to the gradient matrix of all active inequality and equality constraints, such that the second-order matrix of the Lagrange's equation with respect to decision variables  $x^*$  and Lagrange multipliers  $\lambda^*$  and  $\mu^*$  is conditionally positive semidefinite:  $s^T \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*) s \geq 0, \forall s \in S$ .

In a small range around  $x^*$ , the convexity degree of the objective should not exceed the convexity degree of the constraints combined by Lagrange multipliers.

$\forall x \in [x^* - \epsilon, x^* + \epsilon]$ ,  
 $f(x)$  is concave, and  
 $\sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$  is convex.

$\forall x \in [x^* - \epsilon, x^* + \epsilon]$ ,  
 $f(x)$  is convex, and  
 $\sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$  is convex.  
 The degree of convexity of  $f(x)$  is no greater than that of  
 $\sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$ .  
 The second-order sufficient conditions can be met and the  
 local optimal solution  $x^*$  can be found.

$\forall x \in [x^* - \epsilon, x^* + \epsilon]$ ,  
 the degree of convexity of  $f(x)$  is greater than that of  
 $\sum_{i=1}^m \mu_i g_i(x) - \sum_{j=1}^{\ell} \lambda_j h_j(x) = 0$ .  
 The second-order sufficient conditions cannot be met and  
 the local optimal solution  $x^*$  cannot be found.

Back to Slide 7  
Optimizing vs Satisficing  
regarding KKT Conditions

Figure 2.3 The convexity requirements for satisfying the second-order sufficient KKT conditions

Figure 2.4 Lagrange multipliers fail to identify an optimal for a highly convex objective

Backup Slides 37

When the convexity degree of the objective is larger than that of the constraints combined by Lagrange multipliers, meeting sufficient KKT conditions has a higher chance in failing us to get a solution. That is one difference between robust optimization solutions and satisficing solutions.

Systems Realization Laboratory @ OU

Model Evolution for the Realization of Complex Systems  
Lin Guo

GALATICA COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
UNIVERSITY OF ORLANDO

## How is *satisficing* different from optimization? (Section 2.2)

- b) How can designers obtain *satisficing* solutions using cDSP (formulation construct) + ALP (approximation and solution algorithm) + DSIDES (implementation)?

Optimizing

Satisficing

The first-order Lagrange equations is a function of parameters  $\mathcal{P}$ , decision variables  $x$ , and Lagrange multipliers  $\mu, \lambda$

$$\nabla_x L(x, \mu, \lambda) = \psi(\mathcal{P}, x, \mu, \lambda)$$

The second-order Lagrange equation is a functions of parameters  $\mathcal{P}$  and decision variables  $x$

$$\nabla_{xx}^2 L(x, \mu, \lambda) = \nabla_x \psi(\mathcal{P}, x, \mu, \lambda) = \psi'(\mathcal{P}, x)$$

If any uncertainty with probability  $P$  takes place (to at least one item of  $\mathcal{P}$  or  $x$  or  $\mu$  or  $\lambda$ ), the probability  $\mathbb{P}$  that an optimal solution is still optimal is

$$\mathbb{P}(x^* | \mathcal{P}) \approx \prod_{q=1}^Q [1 - p(\bar{\mathcal{P}}_q | \mathcal{P})] \prod_{n=1}^N [1 - P(\bar{x}_n | \mathcal{P})]$$

$$\prod_{i=1}^m [1 - P(\bar{\mu}_i | \mathcal{P})] \prod_{j=1}^{\ell} [1 - P(\bar{\lambda}_j | \mathcal{P})]$$

The first-order Lagrange equation is a function of the coefficients  $\mathcal{p}$  of the deviation variables  $d$  in the goal function

$$\nabla_d L(x^s, d, \mu, \lambda) = \nabla_d z(d) + \sum_{i=1}^m \mu_i \nabla_d g_i(x^s) - \sum_{j=1}^{\ell} \lambda_j \nabla_d h_j(x^s) - \lambda_{\ell+1} \nabla_d G(x^s, d)$$

$$= \nabla_d z(d) + 0 - 0 \pm 1 = \nabla_d z(d) \pm 1 = \psi(\mathcal{p}) \pm 1$$

The second-order Lagrange equation degenerates to zero

$$\nabla_{dd}^2 L(x^s, d, \mu, \lambda) = \nabla_{dd}^2 z(d) \equiv 0$$



If any uncertainty with probability  $P$  takes place (to  $\mathcal{p}$ ), the probability  $\mathbb{P}$  that a satisficing solution is still satisficing is

$$\mathbb{P}(x^s | \mathcal{P}) \approx \prod_{r=1}^R [1 - p(\bar{\mathcal{P}}_r | \mathcal{P})]$$


$\Rightarrow \mathbb{P}(x^* | \mathcal{P}) \ll \mathbb{P}(x^s | \mathcal{P})$

Backup Slides 46

Another difference is the chance of losing a solution under uncertainty, as it is shown in Slide 46. Satisficing strategy allows us to have a higher chance to maintain a solution under uncertainties. In addition, using optimization methods, designers have assumptions on the distribution or stochasticity of the uncertainty. Those assumptions can be wrong, and the uncertainty must be parameterizable. However, in engineering design, we often face some unparameterizable uncertainties, we cannot manage them using robust optimization.

 Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
 SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

## More Questions from Dr. Trafalis



**Trafalis, Theodore B.**

Today at 5:07 AM

**To:** Mistree, Farrokh; **Cc:** Guo, Lin; Allen, Janet K.; Nicholson, Charles D. ✕

You replied to this message on 7/12/21, 7:09 AM. Show Reply

Lin,

I have read the part related to my questions and I have some extra questions.

1. What are the connections of satisficing solution and robust optimization solution( see formulations of Ben Tal, Bertsimas e.t.c)? I suggest to try a simple toy problem with box constraints.
2. What about Taguchi method and your approach? What about fuzzy optimization ? What about flexible optimization (there is a lot of literature since the seventies)? Can you create a simple toy example and show the resulting solutions?
3. Need to define precisely the concept of robust solution (see other formulations in literature, e.g Bertsimas).
4. In eq. 1.6 what is P calligraphic. It is not defined. In eq. 1.9 what is d and d\*?
5. What do you mean by degree of convexity?
6. Need to discuss more carefully the difference of goal programming solution and satisfying solution (goal programming is strongly related with distance methods in multiobjective optimization and there is a huge literature in this topic).

Backup Slides
30

The second question. We carry on and expand Taguchi's method (Slide 23:)

## Manage Four Types of Uncertainties (Section 2.4.1, 3.4.2)

**Type I** – noise factors. *Noise factors* are not under a designer's control

(Taguchi, 1980)

**Type II** – design variables

(Chen, Allen, and Mistree 1996)

**Type III** – variations in the *mathematical models*

(Choi, Allen, and Mistree 2005)

**Type IV** – variability introduced by a *hierarchical, multiscale or multidisciplinary formulation of the product*.

(Seepersad, Allen, and Mistree 2005)

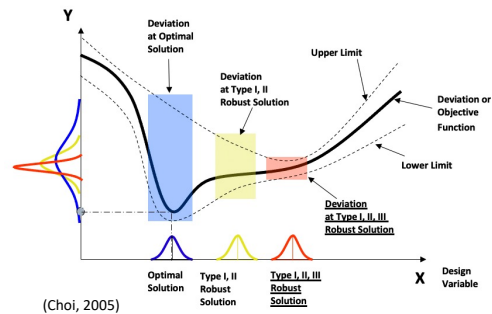


Figure 2.26 Four Types of Robust Solution

Taguchi manages Type I uncertainty, that is noise factors. In 1996, Chen and coauthors expanded it to Type II uncertainty, that is the design (or decision) variables. In engineering design, in a physical system, although we set a certain value to a variable that we thought we can fully control, unparameterizable uncertainties may bring errors or deviations to the variable or to other relevant parts of the system. For example, in a dam network, when we set the water released amount to a certain value and at a certain time, due to some factors not considered into the decision model (and they are usually unable to be captured or quantified, that is why we do not take them into account in the model), such as sudden change in seepage due to climate change, silt variation due to long-term operation, unavoidable tiny operating error, etc., the physical system may not output the same result as the mathematical gets to us. Type II Robust Design is to using EMI (error margin index) to manage the uncertainty in decision variables. Type III uncertainty is the variation in model structure. Choi and coauthors use DCI (design capacity index) to manage it. Type IV uncertainty is the uncertainty in the process chain, which is the uncertainty brought by managing the first three types of uncertainty. The details of the four types of Robust Design in in Section 4.3.1 and 5.3.2. In summary, our method is built on Taguchi's method and we expand it to more types of uncertainty.



## More Questions from Dr. Trafalis



Trafalis, Theodore B.

Today at 5:07 AM

To: Mistree, Farrokh; Cc: Guo, Lin; Allen, Janet K.; Nicholson, Charles D.

You replied to this message on 7/12/21, 7:09 AM.

Show Reply

Lin,

I have read the part related to my questions and I have some extra questions.

1. What are the connections of satisficing solution and robust optimization solution( see formulations of Ben Tal, Bertsimas e.t.c)? I suggest to try a simple toy problem with box constraints.
2. What about Taguchi method and your approach? What about fuzzy optimization ? What about flexible optimization (there is a lot of literature since the seventies)? Can you create a simple toy example and show the resulting solutions?
3. Need to define precisely the concept of robust solution (see other formulations in literature, e.g Bertsimas).
4. In eq. 1.6 what is P calligraphic. It is not defined. In eq. 1.9 what is d and d\*?
5. What do you mean by degree of convexity?
6. Need to discuss more carefully the difference of goal programming solution and satisfying solution (goal programming is strongly related with distance methods in multiobjective optimization and there is a huge literature in this topic).

Backup Slides

30

*As to fuzzy optimization and flexible optimization. The answer is the same with the previous question. Using optimization methods, the objective is consisted of decision variables and the solutions meet sufficient KKT conditions. The differences between optimizing and satisficing also comply with the difference between fuzzy optimization and flexible optimization and satisficing.*



## More Questions from Dr. Trafalis



Trafalis, Theodore B.

Today at 5:07 AM

To: Mistree, Farrokh; Cc: Guo, Lin; Allen, Janet K.; Nicholson, Charles D.

You replied to this message on 7/12/21, 7:09 AM.

Show Reply

Lin,

I have read the part related to my questions and I have some extra questions.

1. What are the connections of satisficing solution and robust optimization solution( see formulations of Ben Tal, Bertsimas e.t.c)? I suggest to try a simple toy problem with box constraints.
2. What about Taguchi method and your approach? What about fuzzy optimization ? What about flexible optimization (there is a lot of literature since the seventies)? Can you create a simple toy example and show the resulting solutions?
3. Need to define precisely the concept of robust solution (see other formulations in literature, e.g Bertsimas).
4. In eq. 1.6 what is P calligraphic. It is not defined. In eq. 1.9 what is d and d\*?
5. What do you mean by degree of convexity?
6. Need to discuss more carefully the difference of goal programming solution and satisfying solution (goal programming is strongly related with distance methods in multiobjective optimization and there is a huge literature in this topic).

Backup Slides

30

*Thank you for the question. I added the definition of robust design in the “definition of terms.”*  
**Robust design** – In this dissertation, robust design means the design that is relatively insensitive to one or more types of uncertainty. Type I uncertainty – the uncertainty brought by noise factors,

for example, parameters. Type II uncertainty – the uncertainty brought by control factors such as decision variables. Type III uncertainty – the variation in the model structure. Type IV – the uncertainty brought by managing the first three types of uncertainty. We are aware of other definitions of robust design, but in this dissertation, in the context of designing complex systems, we define robust design as the above.

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
GALILEO COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

## More Questions from Dr. Trafalis

TT

**Trafalis, Theodore B.** Today at 5:07 AM

To: Mistree, Farrokh; Cc: Guo, Lin; Allen, Janet K.; Nicholson, Charles D. ✕

You replied to this message on 7/12/21, 7:09 AM.
Show Reply

Lin,

I have read the part related to my questions and I have some extra questions.

1. What are the connections of satisficing solution and robust optimization (see formulations of Ben Tal, Bertsimas e.t.c)? I suggest to try a simple toy problem with box constraints.
2. What about Taguchi method and your approach? What about fuzzy optimization? What about flexible optimization (there is a lot of literature since the seventies)? Can you create a simple toy example and show the resulting solutions?
3. Need to define precisely the concept of robust solution (see other formulations in literature, e.g Bertsimas).
4. In eq. 1.6 what is P calligraphic. It is not defined. In eq. 1.9 what is d and d\*?
5. What do you mean by degree of convexity?
6. Need to discuss more carefully the difference of goal programming solution and satisficing solution (goal programming is strongly related with distance methods in multiobjective optimization and there is a huge literature in this topic).

Backup Slides
30

Here I explain in the next slide, which is a screenshot from Section 1.4.2.

Systems Realization Laboratory @ OU
Model Evolution for the Realization of Complex Systems  
Lin Guo
GALILEO COLLEGE OF ENGINEERING  
SCHOOL OF INDUSTRIAL  
AND SYSTEMS ENGINEERING  
UNIVERSITY OF OKLAHOMA

## 4. The “P” in the equations

**Given**

$$f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n \quad \text{Equation 1.4}$$

$$\Omega = \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k, \} \quad \text{Equation 1.5}$$

**Find**

$$x^*: \mathcal{P}_{x \in \Omega}(f(x) = \text{Target}) \quad \text{Equation 1.6}$$

The merit function of a decision model using *satisficing* strategy is identifying the nearest projection of the objective function  $f(x)$  onto the feasible space bounded by constraints. Or in other words, the aim is minimizing the deviation between the target and the actual achieved value of a goal, as Equation 1.7-1.9

**Given**

$$f: \mathbb{R}^n \rightarrow \mathbb{R}, \Omega \subseteq \mathbb{R}^n \quad \text{Equation 1.7}$$

$$\Omega = \{x \in \mathbb{R}^n | g_i(x) \geq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, k, f(x) + d^- - d^+ = \text{Target}\} \quad \text{Equation 1.8}$$

**Find**

$$d^*: d^*(x) \leq d(x) \quad \text{Equation 1.9}$$


[https://en.wikipedia.org/wiki/Projection\\_\(mathematics\)](https://en.wikipedia.org/wiki/Projection_(mathematics))

Minimum deviation. Same as  $x^*: f(x^*) \geq f(x), \forall x \in \Omega$

Backup Slides
31

The  $P$  calligraphic means the nearest projection of the set in the parentheses onto the set  $\Omega$ .  
 $d^*(x)$  means the minimum deviation among all deviations.

## More Questions from Dr. Trafalis



**Trafalis, Theodore B.**

Today at 5:07 AM

**To:** Mistree, Farrokh; **Cc:** Guo, Lin; Allen, Janet K.; Nicholson, Charles D. ✕

You replied to this message on 7/12/21, 7:09 AM. Show Reply

Lin,

I have read the part related to my questions and I have some extra questions.

1. What are the connections of satisficing solution and robust optimization solution( see formulations of Ben Tal, Bertsimas e.t.c)? I suggest to try a simple toy problem with box constraints.
2. What about Taguchi method and your approach? What about fuzzy optimization ? What about flexible optimization (there is a lot of literature since the seventies)? Can you create a simple toy example and show the resulting solutions?
3. Need to define precisely the concept of robust solution (see other formulations in literature, e.g Bertsimas).
4. In eq. 1.6 what is  $P$  calligraphic. It is not defined. In eq. 1.9 what is  $d$  and  $d^*$ ?
5. What do you mean by degree of convexity?
6. Need to discuss more carefully the difference of goal programming solution and satisficing solution (goal programming is strongly related with distance methods in multiobjective optimization and there is a huge literature in this topic).

Backup Slides

30

*In this dissertation, we simplify the degree of convexity (or convexity degree) of an equation as the average value of the diagonal terms of its Hessian matrix, as it is shown in the next slide.*

## 5. Degree Convexity

- A constraint  $g_i$  is convex with respect to  $X_j$  if

$$\left( \frac{\partial^2 g_i}{\partial X_j^2} \right) < 0$$

- It is adequate to estimate the **degree of convexity** as

$$\text{degree of convexity} = \frac{1}{n} \sum_{j=1}^n \left( \frac{\partial^2 g_i}{\partial X_j^2} \right)$$

- A constraint is accumulated if its **degree of convexity** is negative (i.e., more negative than -0.015) and if it was active in the immediately preceding iteration.

These three rules coupled with a special normalized form of the constraint function {9a}, have been found to be adequate for obtaining the solution of a wide variety of compromise DSPs. Furthermore, a constraint will only be accumulated if it is an inequality constraint. System goals therefore are never accumulated since they are always equalities in the compromise DSP formulation.

Mistree, F., Hughes, O. F., Bras, B., & Kamat, M. P. (1993). Compromise decision support problem and the adaptive linear programming algorithm. *Progress in Astronautics and Aeronautics*, 150, 251-251.

Backup Slides

32

*This definition is from Farrokh Mistree's 1993 paper.*



## More Questions from Dr. Trafalis



Trafalis, Theodore B.

Today at 5:07 AM

To: Mistree, Farrokh; Cc: Guo, Lin; Allen, Janet K.; Nicholson, Charles D.

You replied to this message on 7/12/21, 7:09 AM.

Show Reply

Lin,

I have read the part related to my questions and I have some extra questions.

1. What are the connections of satisficing solution and robust optimization solution( see formulations of Ben Tal, Bertsimas e.t.c)? I suggest to try a simple toy problem with box constraints.
2. What about Taguchi method and your approach? What about fuzzy optimization ? What about flexible optimization (there is a lot of literature since the seventies)? Can you create a simple toy example and show the resulting solutions?
3. Need to define precisely the concept of robust solution (see other formulations in literature, e.g Bertsimas).
4. In eq. 1.6 what is P calligraphic. It is not defined. In eq. 1.9 what is d and d\*?
5. What do you mean by degree of convexity?
6. Need to discuss more carefully the difference of goal programming solution and satisficing solution (goal programming is strongly related with distance methods in multiobjective optimization and there is a huge literature in this topic).

Backup Slides

30

In Section 2.28, I use Toy Problem IV to demonstrate the satisficing strategy we use (cDSP+ALP+DSIDES) is different from Goal Programming.

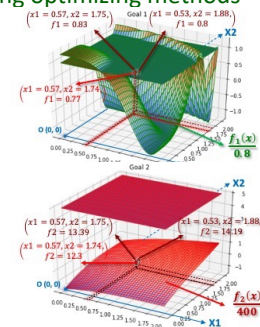
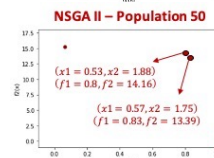
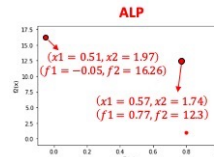


## 6. Difference between Goal Programming solution and satisficing solution

Toy Problem IV: we formulate a goal programming problem and solve it using optimizing methods

Table 2. 10 The Compromise DSP of the TP-IV

TP	The Compromise DSP
IV	<b>Given</b>
	$x_1, x_2, d_1^{\pm}, d_2^{\pm}$
	$f_1(x) = \cos(x_1^2 + x_2^3)$
	$f_2(x) = 25 \cdot (x_1 - 2)^3 + 50 \cdot (x_2 - 2)^3 + 50 \cdot x_1 \cdot x_2^2$
	<b>Find</b>
	$x_1, x_2, d_1^{\mp}, d_2^{\mp}$
	<b>Satisfy</b>
	<b>Goals:</b>
	$\frac{f_1(x)}{1.2} + d_1^- = 1$
	$\frac{f_2(x)}{400} + d_2^- = 5$
<b>Constraints:</b>	
$x_1 \cdot x_2 \leq 1$	
<b>Bounds:</b>	
$d_i^- \cdot d_i^+ = 0, i = 1, 2$	
$0 \leq x_1, x_2 \leq 2$	
$0 \leq d_1^{\pm}, d_2^{\pm} \leq 1$	
<b>Minimize</b>	
$Z = \sum_{i=1}^2 w_i \cdot (d_i^- + d_i^+)$	



**Observation:** cDSP and ALP are designed to formulate and explore engineering-design problems with various completabilities of the goals:

$$\frac{f_i(x)}{T_i} \gg \frac{f_j(x)}{T_j}$$

Equation 2. 24

Using the ALP, good enough solutions (comparing with the solutions from NSGA II) can be obtained, whereas using optimizing algorithms, no feasible solutions are returned.

Backup Slides

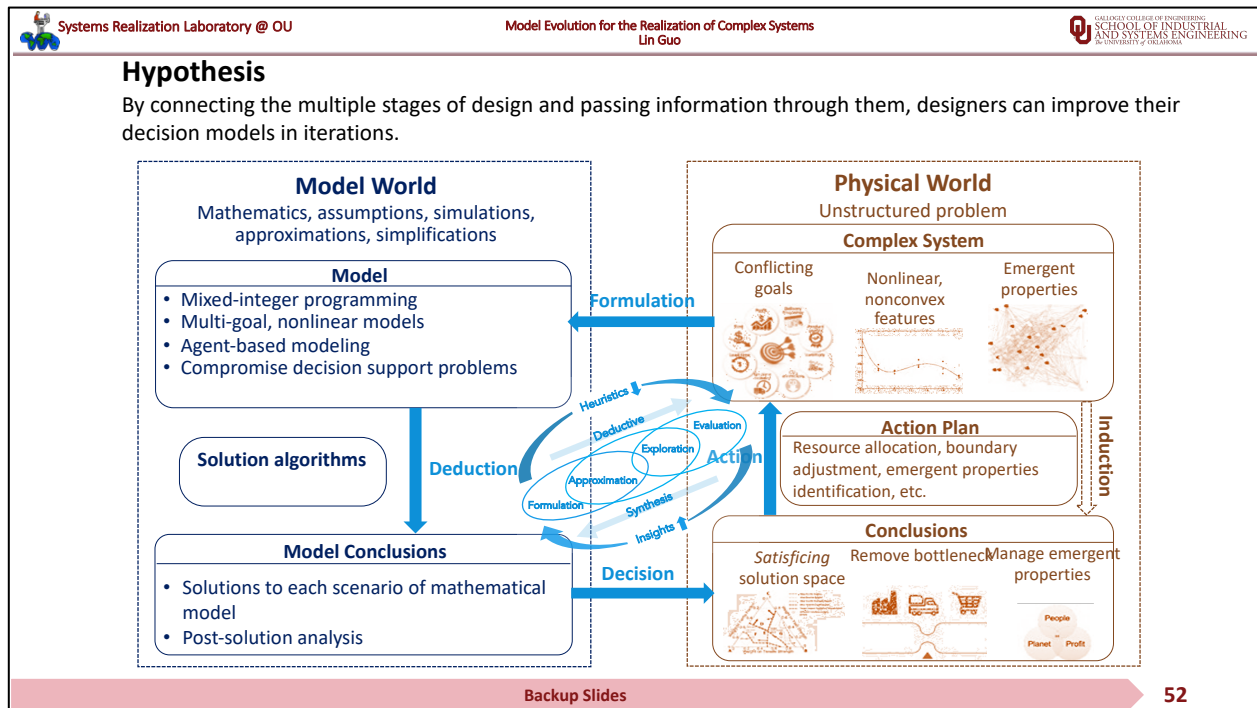
34

For Toy Problem IV, we formulate one cDSP (the same as Goal Programming regarding the formulation) and use both optimization methods and satisficing strategy to solve it. However, none of the optimization solution algorithms in the scipy.optimize package can return any feasible solutions, whereas the ALP returns solutions. Why does this happen?



Because only when we use the ALP to solve a cDSP, can we get a feasible solution that violates the upper bound of the deviation variables. That means if the target of a goal is too ambitious – the feasible space bounded by constraints are far away from the goal, we have to violate the upper bound of the deviation variable to obtain a solution. Such violation is not allowed in Goal Programming when using optimization solution algorithms. Only using ALP, can we allow this kind of violations. Therefore, if we use optimization methods to solve a cDSP (or a Goal Programming problem), the solutions are not satisficing solutions. The difference between Goal Programming and cDSP is not in the formulation format, but in the solution algorithm. Those are the answer to the six questions from Dr. Trafalis.

Here are all the other backup slides:



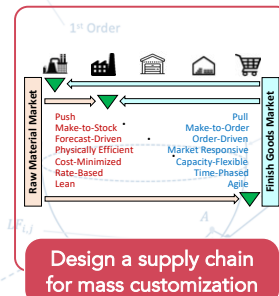
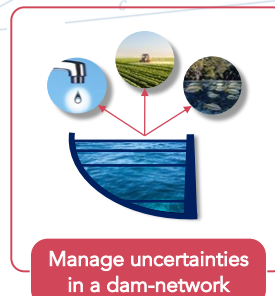
**Research Questions (RQ) and Specified Hypotheses (SH) (2/2)**

Chapter	Ch1	Ch2	
<b>Actions</b>	Research Gap: How can designers realize model evolution using satisficing strategy so that they can manage chaos in the physical world, reduce the risk of losing an optimal solution, and discover domain-independent knowledge to update metaheuristics? Hypothesis: By the multiple stages of design and passing information through them, designers can improve their decision models in iterations.	RQ1: What is the method to evolve model <b>boundary</b> ?	SH1: Explore the sensitivity of the segments of the model boundary and improve accordingly.
		RQ2: What is the method to evolve model to <b>update metaheuristics</b> ?	SH2: Learn, evaluate, and update metaheuristics to improve model performance.
		RQ3: What is the method to speed up <b>learning the system nature</b> ?	SH3: Learn system nature such as interrelationship among subsystems and reorganize them based on it.
		RQ4: What is the method that allows model evolution by <b>incorporating emergent properties</b> ?	SH4: Capture and quantify emergent properties through scenario planning in simulation.



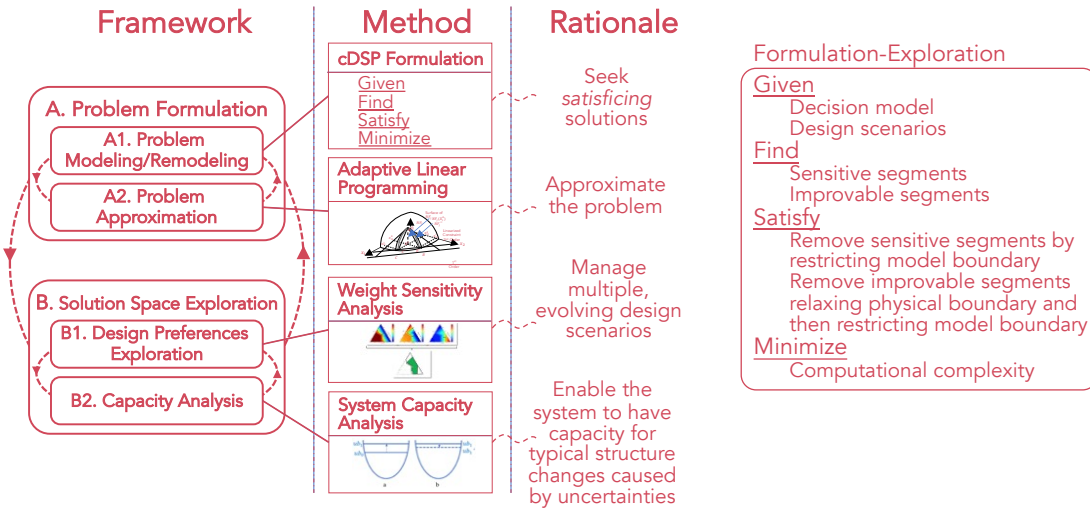
Methods  
Applications  
Contributions

CONSIDERATIONS

**Hypothesis 1 – Expansion**  
 Evolving the model boundary to manage uncertainties


## Manage Uncertainties in a Dam-Network

Proposed method: **Formulation-Exploration Framework**



Methods  
Applications  
Contributions

Given (System parameters)

$$FT_d^t, AT_d^t, (E_d^t + P_d^t), CF_d^t, CM_d^t, (I_d^t + Pr_d^t), w_i, \text{ where } i = 1, 2, 3$$

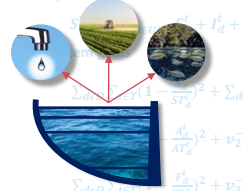
Find

### Hypothesis 1

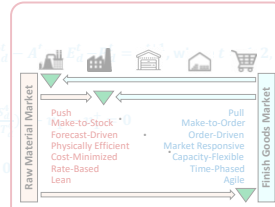
Evolving a model to manage uncertainties

Satisfy  $v_i^t, v_i^t, \text{ where } i = 1, 2, 3$

System constraint



Manage uncertainties in a dam-network



Design a supply chain for mass customization

Minimize (The deviation function)

$$z = \sum_{i=1}^3 w_i \cdot v_i^t, \text{ where } 0 \leq w_i \leq 1, \text{ and } \sum_{i=1}^3 w_i = 1$$

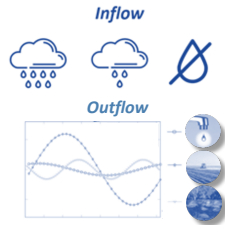
# Manage Uncertainties in a Dam-Network

## Problem statement

14-dam-network on Red River Basin



### Uncertainties



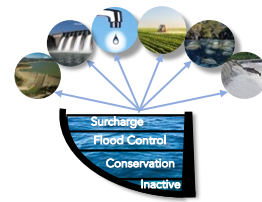
Expected functions of a dam-network



### Decisions

Water flow plans  
Model improvement

Goals



### Features of complex systems

- Nonlinear, discrete, and non-convex
- Fewer factors to control but a lot of requirements
- Need to explore the ways to combine the multiple goals
- Manage multiple types of uncertainty
- Need to know more about the model robustness and ways to improve the model formulation

Backup Slides

57

# Manage Uncertainties in a Dam-Network

## Model and evolution

Given (System parameters)

$$FT_d^t, AT_d^t, (E_d^t + P_d^t), CF_d^t, CM_d^t, (I_d^t + Pr_d^t) w_i, \text{ where } i = 1, 2, 3$$

Find

System variables

$$S_d^t, A_d^t, F_d^t$$

Deviation variables

$$v_i^+, v_i^-, \text{ where } i = 1, 2, 3$$

Satisfy

System constraint

$$S_d^t + \sum_{v \in U D_d} F_d^t + I_d^t + Pr_d^t - F_d^t - A_d^t - E_d^t - P_d^t = S_d^{t+1}, \text{ where } t = 1, 2, 3$$

System goals

$$\sum_{d \in D} \sum_{t \in T} (1 - \frac{S_d^t}{ST_d^t})^2 + \sum_{d \in D} (1 - \frac{S_d^t}{ST_d^t})^2 + v_1^- - v_1^+ = 0$$

$$\sum_{d \in D} \sum_{t \in T} (1 - \frac{A_d^t}{AT_d^t})^2 + v_2^- - v_2^+ = 0$$

$$\sum_{d \in D} \sum_{t \in T} (1 - \frac{F_d^t}{FT_d^t})^2 + v_3^- - v_3^+ = 0$$

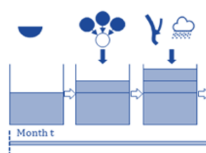
Bounds

$$CM_d \leq S_d^t \leq CF_d, F_d^t \geq 0, A_d^t \geq 0$$

$$0 \leq v_i^- \leq 0, v_i^- \cdot v_i^+ = 0 \text{ where } i = 1, 2, 3$$

Minimize (The deviation function)

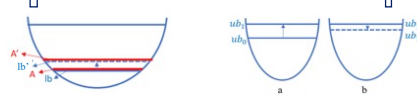
$$z = \sum_{i=1}^3 w_i \cdot v_i^-, \text{ where } 0 \leq w_i \leq 1, \text{ and } \sum_{i=1}^3 w_i = 1$$



#	Infl.			
1	No			
2	Ext			
3	Dry			
4	Not			
5	Rai			
6	Ext			
7	Flo...			
8	Rain Unevenly I	0%	200%	100%
9	Rain Unevenly II	400%	0%	0%
10	Rain Unevenly III	400%	0%	400%

Table 9 suggestions of model improvement

#	Improvement	Constraints or Bounds functions change, i=1,2,3
1	Raise the lower bound of storage volume in Reservoir 7 by 1%	S7M >= -40900 => S7M >= -40900*1.01
2	Raise the upper bound of storage volume in Reservoir 6 by 1%	S6M2 <= -74799 => S6M2 <= -74799*1.01

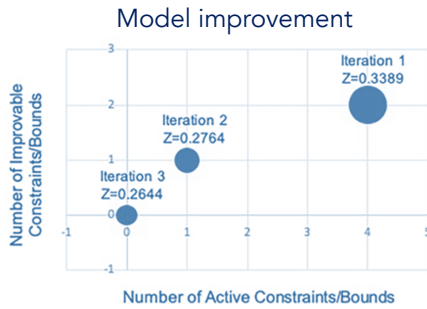


Backup Slides

58



## Manage Uncertainties in a Dam-Network Results and outcome



### New Knowledge

#### Formulation-Exploration Framework

##### Given

- Decision model
- Design scenarios

##### Find

- Sensitive segments
- Improvable segments

##### Satisfy

- Remove sensitive segments by restricting model boundary
- Remove improvable segments relaxing physical boundary and then restricting model boundary

##### Minimize

- Computational complexity

### Publications

Guo, L., Zamanisabzi, H., Neeson, T.M., Allen, J.K., Mistree, F., 2019, "Managing Conflicting Water Resource Goals and Uncertainties in a Dam-Network by Exploring the Solution Space," *ASME Journal of Mechanical Design*, 141(3): 031702.

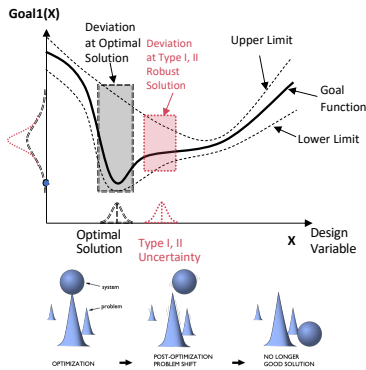
Guo, L., Zamanisabzi, H., Neeson, T.M., Allen, J.K., Mistree, F., 2018, "Managing Conflicting Water Resource Goals and Uncertainties in a Dam-Network by Exploring the Solution Space," *ASME 44th Design Automation Conference*, Quebec City, Quebec, Canada. Paper Number DETC2018-86018.

Backup Slides

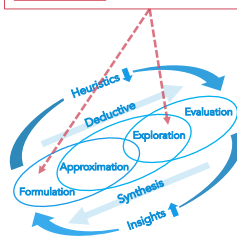
59



## Answer to Research Question 1



RQ1: What is the method to evolve model to adapt to uncertainties in parameters and variables?



### New Knowledge

#### Formulation-Exploration Framework

##### Given

- Decision model
- Design scenarios

##### Find

- Sensitive segments
- Improvable segments

##### Satisfy

- Remove sensitive segments by restricting model boundary
- Remove improvable segments relaxing physical boundary and then restricting model boundary

##### Minimize

- Computational complexity

### Potential Applications

- Detecting hidden bottleneck of healthcare networks
- Lean process design for cyber-physical product-service systems

### Potential funding sources

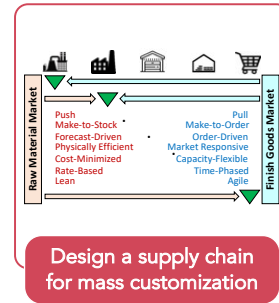
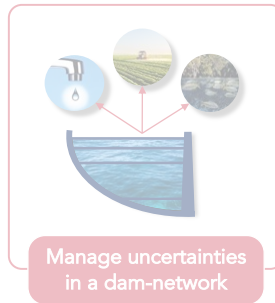
- NSF Engineering – UKRI Engineering and Physical Sciences Research Council (ENG-EP SRC)
- NSF Advanced Manufacturing (AM) Program

Backup Slides

60

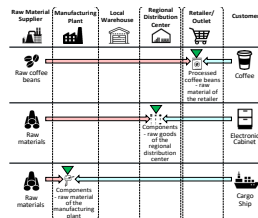
Methods  
Applications  
Contributions

Hypothesis 1  
Evolving a model to manage  
uncertainties

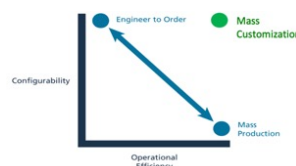


Design A Supply Chain for Mass Customization  
Problem statement

Position the customer order  
decoupling point (CODP)



Expected functions of a  
supply chain for mass  
customization



Goals and  
constraints



Uncertainties

Uncertainty in demand  
and supply through a  
product life cycle



Decisions

CODP position  
Service level  
Reliability

Features of complex systems

- Nonlinear, discrete, and non-convex
- Fewer factors to control but a lot of requirements
- Need to explore the ways to combine the multiple goals
- Manage multiple types of uncertainty
- Need to know more about the model robustness and ways to improve the model formulation



## Design A Supply Chain for Mass Customization

### Research gaps in literature

Method	Gaps
Queue model	Assumptions of order / customer arrival distribution; <b>over-simplified</b> using single objective with only cost as the unit
Time scheduling	<b>No scaling</b> for multi-objective problems; assumptions in demand
Stochastic programming / dynamic programming	Assumptions of order / customer arrival distribution; decrease the frequency but not the severity of the failures
AHP / Network simulation	Relying on <b>domain expertise</b> which can be <b>subjective</b>

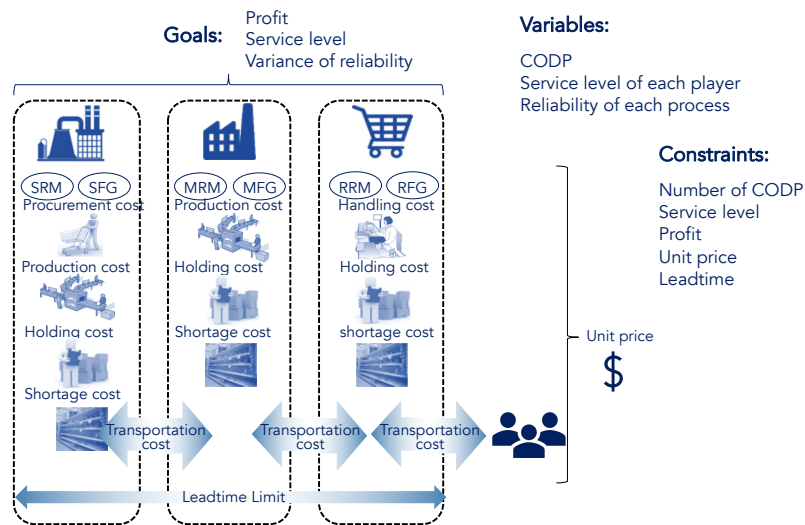
#### Summary of gaps:

- Assumptions in the distribution of non-deterministic parameters can be wrong, yet no evaluation and improvement mechanics regarding the assumptions
- Lacking mechanism of continuous improvement regarding system bottleneck



## Design A Supply Chain for Mass Customization

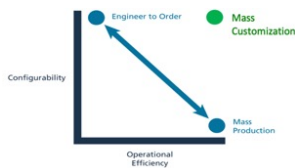
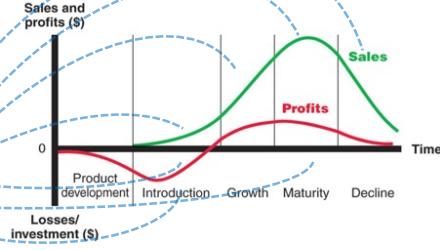
### Model and evolution



## Design A Supply Chain for Mass Customization

### Results and outcome

WS	Icons	Profit, Performance stability, and Product life cycle
1		Low profit, unstable - decline
2		High profit, unstable - maturity
3		Hi profit, stable - growth
4		Low profit, stable - introduction
5		Low profit, stable - introduction
6		High profit, unstable - maturity
7		Low profit, stable - introduction



### Publications

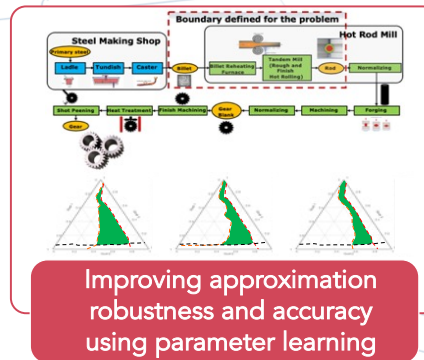
Guo, L., Chen, S., Allen, J.K., Mistree, F., 2020, "A Framework for Designing the Customer Order Decoupling Point to Facilitate Mass Customization," *ASME Journal of Mechanical Design*, 143(2): 022002  
 Guo, L., Chen, S., Allen, J.K., Mistree, F., 2019, "Designing the Customer Order Decoupling Point to Facilitate Mass Customization," ASME 45th Design Automation Conference, Anaheim, CA, USA, Paper Number DETC2019-97379.



Methods  
Applications  
Contributions

CONSIDERATIONS

## Hypothesis 2 Discover knowledge to update metaheuristics using deep learning

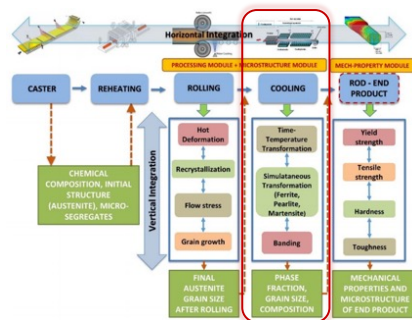




# Improve Approximation Using Parameter Learning

## Problem statement (1/2)

### The Hot Rolling Process Chain Problem



#### Variables

- Cooling rate CR
- Austenite grain size (AGS)  $D$
- Carbon concentration
- Manganese concentration after rolling Mn

#### Goals

- Ferrite Grain Size  $\frac{D_{\alpha Target}}{D_{\alpha}(X_i)} - d_1^- + d_1^+ = 1$
- Ferrite Fraction  $\frac{X_f(X_i)}{X_{f Target}} + d_2^- - d_2^+ = 1$
- Interlamellar Spacing  $\frac{S_o}{S_o(X_i)} - d_3^- + d_3^+ = 1$

Nellippallil, A. B., Rangaraj, V., Gautham, B., Singh, A. K., Allen, J. K., and Mistree, F., 2018, "An Inverse, Decision-Based Design Method for Integrated Design Exploration of Materials, Products, and Manufacturing Processes," Journal of Mechanical Design, 140(11), p. 111403.

#### Ferrite Grain Size

$$D_{\alpha} = (1 - 0.45e_r^{0.5}) \times \{(-0.4 + 6.37C_{eq}) + (24.2 - 59C_{eq})CR^{-0.5} + 22[1 - \exp(-0.015D)]\}$$

for  $C_{eq} < 0.35$

$$D_{\alpha} = (1 - 0.45e_r^{0.5}) \times \{(22.6 - 57C_{eq}) + 3CR^{-0.5} + 22[1 - \exp(-0.015D)]\}$$

for  $C_{eq} > 0.35$



Hardness  
Yield strength  
Tensile strength

#### Ferrite Fraction

$$X_f = 1 - (0.206 - 0.117[Mn] - 0.0005CR - 0.00113D + 0.248[C] + 0.00032[Mn]CR + 0.000086[Mn]D + 0.9539[Mn][C] - 4.259 \times 10^{-6}CR * D + 0.00726CR[C] + 0.0023D[C] - 0.0305[Mn]^2 - 0.0000056CR^2 + 4.859 \times 10^{-6}D^2 + 0.79[C]^2)$$

$X_f \uparrow$  toughness  $\downarrow$   
elongation  $\downarrow$   
 $X_f \downarrow$  chloride stress corrosion cracking  $\uparrow$

#### Interlamellar Spacing

$$S_o = 0.1307 + 1.027[C] - 1.993[C]^2 - 0.1108[Mn] + 0.0305CR^{-0.52}$$



Hardness  
Yield strength

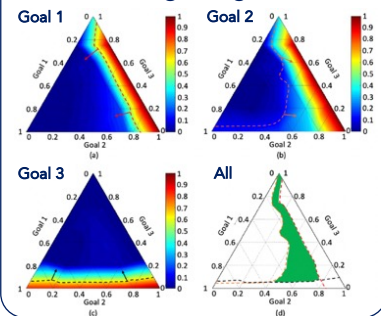
Backup Slides

67

# Improve Approximation Using Parameter Learning

## Problem statement (2/2)

### The Satisficing Weight Set

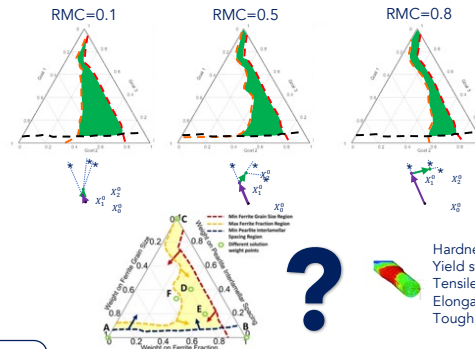


#### Goals

- Achieve Ferrite Grain Size Target  $\frac{D_{\alpha Target}}{D_{\alpha}(X_i)} - d_1^- + d_1^+ = 1$
- Achieve Ferrite Fraction Target  $\frac{X_f(X_i)}{X_{f Target}} + d_2^- - d_2^+ = 1$
- Achieve Interlamellar Spacing Target  $\frac{S_o}{S_o(X_i)} - d_3^- + d_3^+ = 1$

Nellippallil, A. B., Rangaraj, V., Gautham, B., Singh, A. K., Allen, J. K., and Mistree, F., 2018, "An Inverse, Decision-Based Design Method for Integrated Design Exploration of Materials, Products, and Manufacturing Processes," Journal of Mechanical Design, 140(11), p. 111403.

Using Adaptive Linear Programming (ALP) algorithm  
Linearize the problem based on Reduce Move Coefficient



Why?

- No information passing
- Rely on domain expertise
- Rely on metaheuristics to make rules
- No mechanism to update metaheuristics

Hardness  
Yield strength  
Tensile strength  
Elongation  
Toughness

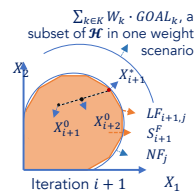
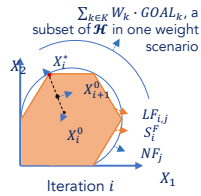
Backup Slides

68

## Improve Approximation Using Parameter Learning

### Research gaps in literature

Metaheuristics in the ALP – determining the reduced move coefficient (RMC)



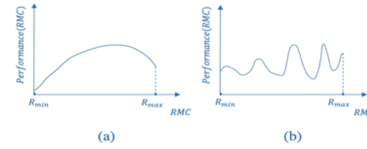
### Golden Section Search



if  $Performance(A) \gg Performance(B)$



if  $Performance(A) \ll Performance(B)$



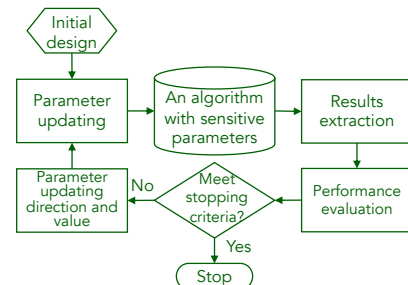
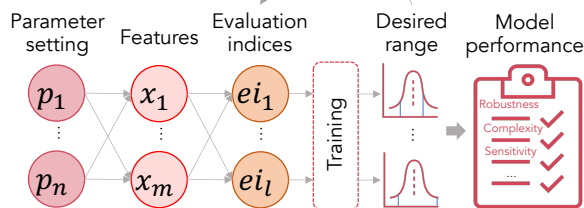
1. The performance of the algorithm may not be convex with the RMC value
2. There is not a definition of the "performance" of the algorithm

#### Summary of gaps:

- Lacking criteria to evaluate the approximation performance associated with parameter value
- No mechanism to make the approximation relatively insensitive to the parameter

## Improve Approximation Using Parameter Learning

### Proposed method: Adaptive Linear Programming algorithm with Parameter Learning (ALPPL)



#### ALPPL

##### Given

Decision Model  
Design scenarios (DS)

##### Find

Value of key parameters  
Evaluation Indices (EIs)  
Desired range of EIs (DEI)

##### Satisfy

EIs in DEI

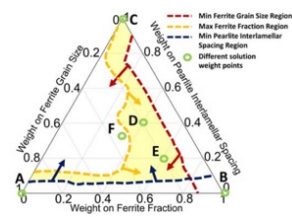
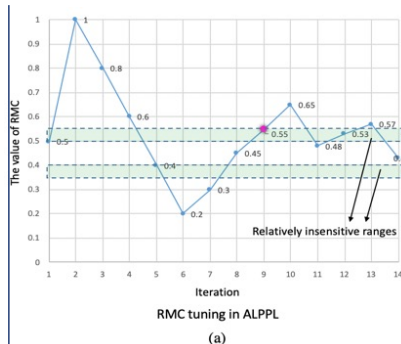
##### Minimize

$\sigma_{EIs}$  for all DS

## Improve Approximation Using Parameter Learning

### Results and verification

- Parameter learning is better than golden section search in
- More evaluation standards
  - Robustness of the solutions
  - Exploring the insensitive range more sufficiently



We are confident that it is good enough

Method	Parameter learning
Evaluation Standard	EIs (Goal achievement, robustness)
Performance	RMC: 0.55 in insensitive range
Insensitive range exploration	Explore relatively sufficiently (29%)

Backup Slides

71

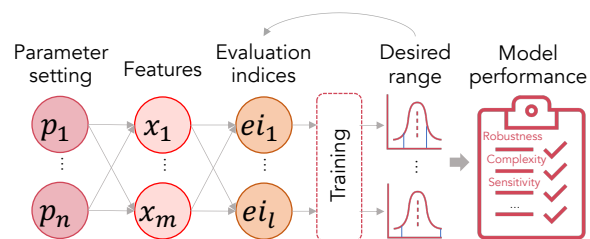
## Improve Approximation Using Parameter Learning

### Contribution and outcome

Parameter learning procedure for a variety of linear algorithms

```

1 Given:  $DEI$ , the best  $RMC$  sample value,  $DEI$  updating rules
2 Initialize:  $t < 0$ ,  $best <$  the best  $RMC$  sample value,  $RMC_0 <$ 
the best  $RMC$  sample value, the maximum iteration number  $T$ ,
stopping criterion 2  $<$  { $best$  has not been updated in  $n$  iterations}
3 While  $t \leq T$  do // Define stopping criterion 1
4    $RMC_t <$  Next  $RMC$ 
5   Run synthesis cycle
6   Calculate  $EI[RMC_t]$ 
7   if  $EIs[RMC_t] > EIs[best]$ 
8      $best <$   $RMC_t$  // Update the best RMC
9   if  $DEI$  should be updated
10    update  $DEI$ 
11 if stopping criterion 2 is met
12   break
13 else
14   Next  $RMC <$ 
Determine  $next\{EIs[RMC_t], EIs[RMC_{t-1}], EIs[RMC_{t-2}], best\}$ 
// Use  $EIs$  of the  $t^{th}$ ,  $(t-1)^{th}$ ,  $(t-2)^{th}$ , and  $best$  to determine the  $RMC$ 
of the next iteration9
15    $t <$   $t+1$ 
16 return  $best$ 
    
```



Parameter tuning?  
 Parameter calibration?  
 Neural networks?

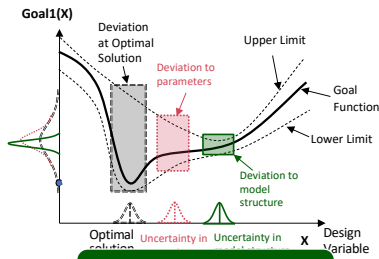
Publication

Guo, L., Nellippallil, A.B., Smith, W.F., Allen, J.K., Mistree, F., "Adaptive Linear Programming Algorithm with Parameter Learning," ASME 46th Design Automation Conference, Paper Number DETC2020-22602.

Backup Slides

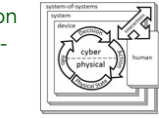
72

## Answer to Research Question 2

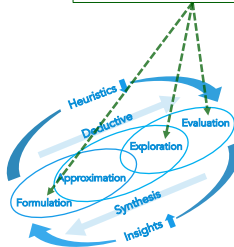


### Potential Applications

- Directed evolution of cyber-physical-social systems
- Designing large-scale, multidisciplinary systems using reinforcement learning



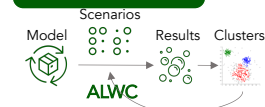
**RO2** What is the method to speed up learning the system nature?



### Potential funding sources

- NSF Smart and Connected Communities (S&CC) program
- NSF Engineering Design and System Engineering (EDSE) program

### New Knowledge



- Given**  
Initial design scenarios (DS)
- Find**  
Interrelationship among goals ( $I_D$ )  
Clusters of the goals  $C_G$
- Satisfy**  
Improving goal achieved value  
Updating DS based on  $C_G$
- Maximize**  
Goal achievement  
Diversity of solutions

### Publication

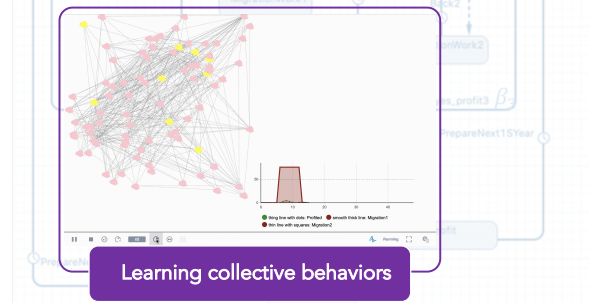
Guo, L., Milisavljevic-Syed, J., Wang, R., Huang Y., Allen, J.K., Mistree, F., "Managing Many-Goal, Concurrent Design Problems using Adaptive Leveling-Weighting-Clustering Algorithm," Advanced Engineering Informatics, under review.



Methods  
Applications  
Contributions

CONSIDERATIONS

**Hypothesis 4**  
Capture, quantify, and model emergent properties through scenario planning



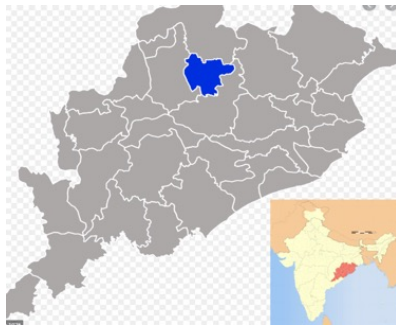
$$\beta_3 \ll \beta_1$$

$$\beta_3 = f(\alpha, \beta_1, \beta_2)$$



## Learn collective behaviors

### Problem statement



A social entrepreneur needs to promote second-season cultivation using underground water in a relatively isolated village.

### Questions

- What are the critical factors that affect the collective behaviors under interventions?
- How can we identify the critical factors and select the appropriate scenario to reach an expected result?

Backup Slides

75



## Learn collective behaviors

### Research gaps in literature

Author and Year	Problem Description	Method	Results	Contribution
Opiyo, 2019 [11]	Study the neighborhood influence and social pressure on temporal diffusion of solar home system	Agent-based modeling with the data from a survey	Visibility of newly installed SHS and increasing influence radius leads to growth in SHS installations	The survey development is helpful to acquire relatively quantifiable data for a social problem
Qiu, 2018 [10]	Simulate urban land development and population dynamics	Use an agent-based and spatial genetic algorithm framework (PDULD)	The government policies have dominated the process of land development	Community cohesion theory is introduced into the model; historic data are used to verify the results
Irsyad, 2019 [12]	Estimate the effects of four solar energy policy interventions on photovoltaic (PV) investments, government expenditure, economic outputs, etc.	Use a hybrid energy agent-based modeling	Results call for PV donor gift policy, the improvement of production efficiency, after-sales services and rural financing institutions	Integrate the input-output analysis, environmental factors and socioeconomic characteristics of households in Indonesia
Rossoshanskaya, 2019 [13]	Simulate labor potential reproduction; among the scenario forecast, provide decision support on management actions	Agent-based modeling with multi-agent and multi-scenario	The result is the integrated agent-based model of labor potential reproduction at the municipal level	The model is filled with real sociological and statistical data and has a user-friendly interface

#### Summary of gaps:

There is lack of a method that enables capturing critical factors for interventions such as promotion activities at a community level and gives decision support on scenario selection of the critical factors.

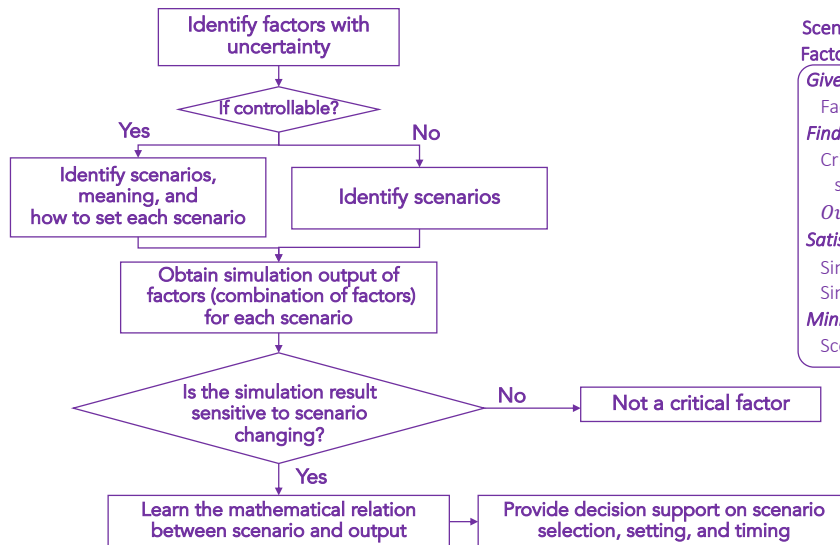
Backup Slides

76



## Learn collective behaviors

### Proposed method: Scenario Planning for Identifying Critical Factors in Simulation



#### Scenario Planning to Identify Critical Factors

**Given**

Factors [Scenarios]

**Find**

Critical Factors [Appropriate scenarios];

$$Output_{Factor_i} = f(Scenario_j)$$

**Satisfy**

Simulation goals

Simulation constraints

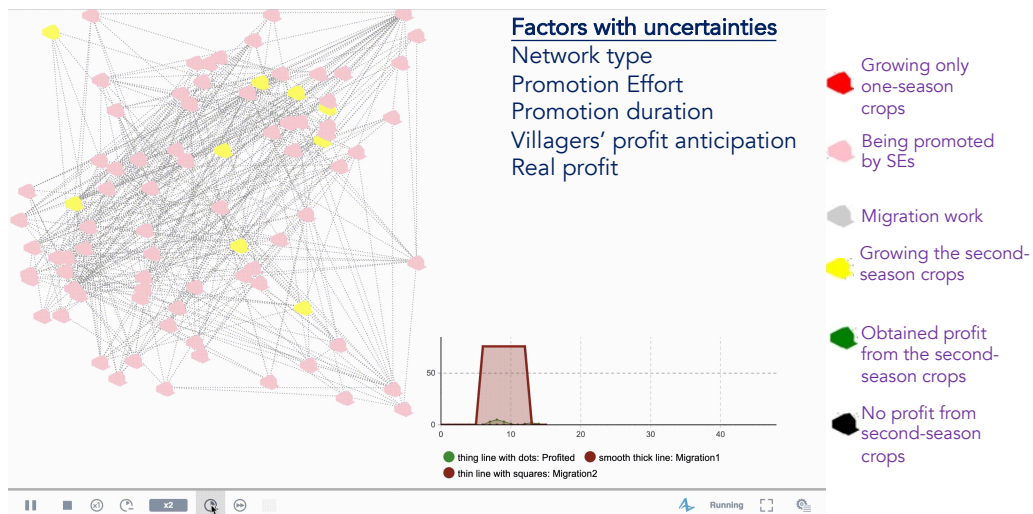
**Minimize**

Scenario combination



## Learn collective behaviors

### Model and scenario planning





## Learn collective behaviors

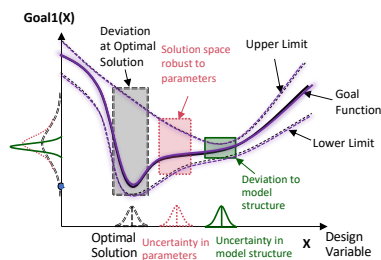
### Results and outcome

Factor	Controllable or not	Scenario	Meaning	Whether critical or not
Network type <sup>4</sup>	No	Distance-based	One community with strong neighborhood influences but weak or zero distant interactions	No
		Scale-free	One community with asymmetric influences between any two connected households and the influence does not depend on distance	
Promotion effort	Yes – SEs can control the promotion effort by reaching out to different numbers of households	Reaching different numbers of households: 10%, 30%, 50%, 75%, 100%	The promotion condition and target can be different from village to village, so searching for an appropriate rate for each village is necessary to reduce the promotion cost	Yes, more critical for short-term effects
Promotion duration	Yes – SEs can perform the promotion for different lengths of time	The promotion duration may last from one month to four months during the rainy season	A short promotion means relatively short direct promotion but long indirect promotion. This is suitable for a village with a strong mutual influence among households and vice versa.	Only critical for short-term effects
Villagers' anticipation and real profit – $\beta_2$ of households anticipates a better profit through second-season cultivation, $\alpha$ of them gain real profit as anticipated	Initially, $\alpha$ and $\beta_2$ are uncontrollable, but as the project goes on, SEs can control them by improving productivity, developing more market demand, etc.	$\beta_2 = 95\%$ $\alpha = 95\%$	Optimistic profit anticipation; good economy, weather condition, or soil fertility	Profit anticipation is critical for short-term effects; real profit is critical for long-term effect
		$\beta_2 = 95\%$ $\alpha = 75\%$	Optimistic profit anticipation; acceptable economy, weather condition, or soil fertility	
		$\beta_2 = 75\%$ $\alpha = 95\%$	Conservative anticipation; good economy, weather condition, or soil fertility	

**Publication**  
 Guo, L., Mohebbi, S., Das, A., Allen, J. K., & Mistree, F. (2020). A Framework for the Exploration of Critical Factors on Promoting Two-Season Cultivation in India. *Journal of Mechanical Design*, 142(12)



## Answer to Research Question 4



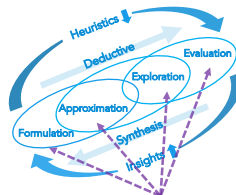
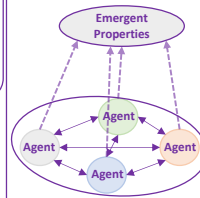
### Publication

Guo, L., Mohebbi, S., Das, A., Allen, J.K., Mistree, F., 2020, "A Framework for the Exploration of Critical Factors on Promoting Two Season Cultivation in India," ASME Journal of Mechanical Design, 142(12): 124503.

### New Knowledge

#### Scenario Planning to Identify Critical Factors

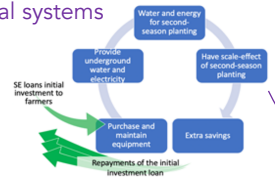
- Given Factors [Scenarios]
- Find Critical Factors [Appropriate scenarios];
- Output<sub>Factor<sub>i</sub></sub> = f(Scenario<sub>i</sub>)
- Satisfy Simulation goals
- Minimize Simulation constraints
- Scenario combination



**RO3:** What is the method that allows model evolution by incorporating emergent properties?

### Potential Application

- Policymaking for cyber-physical-social systems



### Potential funding source

- NSF Smart and Connected Communities (S&CC) program
- Community Service Center Trust Fund (CSCTF) Grant



## Way Forward

MSA FOLMSIQ

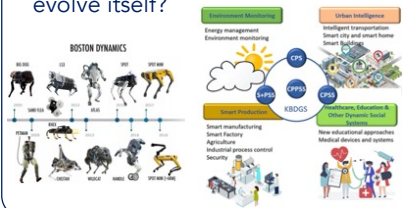
- Research in next 5 years
- Potential funding source
- Potential collaborations
- Relevant courses I can offer

## Algorithm Evolution

### Applications

Providing design guidance for cyber-physical product-service systems (CPPSS)

- Can a CPPSS “deep learn” and evolve itself?



### Potential Funding

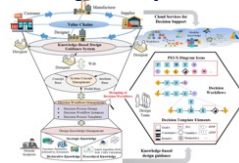
NSF Engineering Design and System Engineering (EDSE) program

[https://www.nsf.gov/funding/pgm\\_summ.jsp?pgm\\_id=505478](https://www.nsf.gov/funding/pgm_summ.jsp?pgm_id=505478)



### New Knowledge

A cloud-based knowledge management system



### Potential Collaboration

- How can we evolve algorithms to learn streaming data from sensors?



### Relevant Courses

*Introduction to Optimization (4XX/5XX level course)*

- In this course, students learn linear programming, integer programming, dynamic programming, stochastic programming, nonlinear programming, and other *fundamental knowledge and tools in optimization*.

*Designing for Open Innovation (4XX/5XX level course)*

- In this course, students learn how to *account for emergent properties* associated with designing for open innovation, such as a cyber-physical-social system.



### Multiscale Simulation

**Applications**

Designing smart connected communities

**Potential Funding**

NSF Smart and Connected Communities (S&CC) program  
[https://www.nsf.gov/funding/pgm\\_summ.jsp?pims\\_id=505364](https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505364)

Community Service Center Trust Fund (CSCTF) Grant  
<https://www.bhacf.org/community-service-center/>

System Dynamics (SD): top-down design (Organizational goals)

Sensitivity analysis Scenario planning

Streaming data

Agent-Based Modeling (ABM): bottom-up design

**New Knowledge**

A computational framework to learn emergent properties to manage chaos

**Potential Collaborations**

Lean process design leveraging new technologies

- Managing the evolving priority of the multiple drivers in sustainable development in rural communities

Adapted from the 2002 University of Michigan Sustainability Assessment

**Relevant Course**

*Advanced Modeling and Simulations (5XX/6XX level course)*

- In this course, students learn and practice *modeling and simulation tools*, methods, theories, and concepts in the context of managing *complex systems*. In this course, *case studies* and *team projects* are emphasized.

### Data Curation

**Applications**

Fail-safe healthcare network planning

- Dynamically positioning of the customer order decoupling point (CODP)
- Bottleneck detection
- Mass personalization in the healthcare

**Potential Funding**

NSF Engineering – UKRI Engineering and Physical Sciences Research Council (ENG-EPSC)

<https://www.nsf.gov/pubs/2020/nsf20510/nsf20510.htm>

Scenario Model Results Clusters

Chaos to Simplicity

**New Knowledge**

A synthetic data generation method

**Potential Collaborations**

Managing complexity of healthcare networks.

- Dealing with sparse data in data-driven methods

1.0	0.0	0.0	0.0	0.0
0.0	3.0	0.0	0.0	11.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	4.0	0.0	0.0
0.0	0.0	7.0	0.0	0.0
2.0	0.0	0.0	10.0	0.0
0.0	0.0	8.0	0.0	0.0
0.0	4.0	0.0	0.0	12.0

- Data curation for pattern learning based on sensor data

**Relevant Course**

*Data Curation for Designing Complex Systems (4XX level course)*

- In this course, two types of methods are introduced, namely, *data-driven* methods and *process-driven* methods. Methods for *verification and validation* are also introduced.

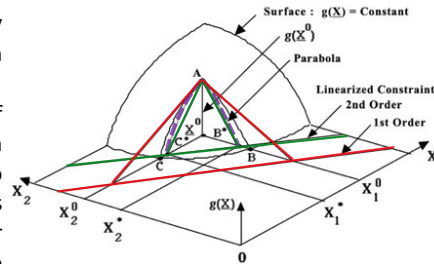


## Details of the Observations from Toy Problem II

Why does the ALP manage non-convex problems and return solutions close to the nondominated solutions (the solutions returned by NSGA II/III)?

Two mechanisms of the ALP allow it to linearize the non-convex function relatively accurately and converge with good enough solutions.

- First, "sequential linearization." The use of the second-order derivatives function (when the paraboloid being used to approximate the nonlinear function has two real roots) and the first-order derivatives (when the paraboloid has no real root) of the nonlinear functions make the linear problem relatively robust. The nonlinear equation is first approximated into a paraboloid and then approximated into a linear equation.



First-order Linearization  
Second-order Parabola  
Sequential Linearization

Backup Slides

85



## Details of the Observations from Toy Problem II

Why does the ALP manage non-convex problems and return solutions close to the nondominated solutions (the solutions returned by NSGA II/III)?

Two mechanisms of the ALP allow it to linearize the non-convex function relatively accurately and converge with good enough solutions.

- First, "sequential linearization." The use of the second-order derivatives function (when the paraboloid being used to approximate the nonlinear function has two real roots) and the first-order derivatives (when the paraboloid has no real root) of the nonlinear functions make the linear problem relatively robust. The nonlinear equation is first approximated into a paraboloid and then approximated into a linear equation.

Number of system constraints and goals  $m$ . Running variable  $i = 1, \dots, m$

Number of system variables  $n$ . Running variable  $j = 1, \dots, n$

Step 1: Rewrite constraints and goals as  $g_i(x) \geq 0$

Step 2: Choose initial point  $x^0$

Step 3: Evaluate

$$A = g_i(x^0)$$

$$B_j = \frac{\partial g_i(x)}{\partial x_j} \Big|_{x^0}$$

$$C_j = \frac{\partial^2 g_i(x)}{\partial x_j^2} \Big|_{x^0}$$

Step 4: Evaluate secant plane derivatives (equations 16a and 16b).

Check for positive discriminant and choose smallest root.

IF  $(B_j^2 - 2 \cdot A \cdot C_j) \geq 0$  THEN

$$\text{root1} = (A \cdot C_j) / (-B_j + \sqrt{B_j^2 - 2 \cdot A \cdot C_j})$$

$$\text{root2} = (A \cdot C_j) / (-B_j - \sqrt{B_j^2 - 2 \cdot A \cdot C_j})$$

IF  $|\text{root1}| < |\text{root2}|$  THEN

$$a_j = \text{root1}$$

ELSE

$$a_j = \text{root2}$$

ENDIF

ELSE

$$a_j = B_j$$

ENDIF

Step 6: Evaluate right hand side (RHS)  $b$  of linearized constraint/goal

$$b = \sum a_j x_j^0 - A$$

Step 7: Create final form of system constraint/goal

IF system constraint THEN

$$\sum a_j x_j \geq \text{or} = \text{or} \leq b$$

ELSE system goal

$$\sum a_j x_j + d \cdot d' = b$$

ENDIF

Step 8: Stop

Mistree, F., Hughes, O. F., Bras, B., & Kamat, M. P. (1993). Compromise decision support problem and the adaptive linear programming algorithm. *Progress in Astronautics and Aeronautics*, 150, 251-251.

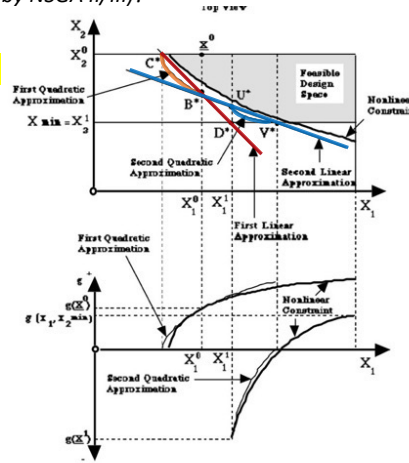
Backup Slides

86

## Details of the Observations from Toy Problem II

Why does the ALP manage non-convex problems and return solutions close to the nondominated solutions (the solutions returned by NSGA II/III)?

- Second, using ALP, the nonlinear, nonconvex equations are **sequentially linearized** in iterations
  - If the gradient of an equation at a local area around the starting point is  $\geq -0.015$  (slightly nonconvex), the equation is linearized around the starting point sequentially, and multiple linear constraints are used to substitute the nonlinear equation.



Backup Slides

87

## Details of the Observations from Toy Problem II

- Why does the ALP manage non-convex problems and return solutions close to the nondominated solutions (the solutions returned by NSGA II/III)?
- Two mechanisms of the ALP allow it to linearize the non-convex function relatively accurately and converge with good enough solutions.
- First, using ALP, the nonlinear, nonconvex equations are **sequentially linearized** in iterations
  - If the gradient of an equation at a local area around the starting point is  $\geq -0.015$  (slightly nonconvex), the equation is linearized around the starting point sequentially, and multiple linear constraints are used to substitute the nonlinear equation.
- Second, the use of the second-order derivatives function (when the paraboloid being used to approximate the nonlinear function has two real roots) and the first-order derivatives (when the paraboloid has no real root) of the nonlinear functions make the linear problem relatively robust. The nonlinear equation is **first approximated into a paraboloid and then approximated into a linear equation**.

Backup Slides

88

My Research Webpage



<http://srl.ou.edu/people/linguo/>

# THANK YOU

My YouTube Channel



[https://www.youtube.com/channel/UC\\_7q6ydcYCG4WObz0G42SVg](https://www.youtube.com/channel/UC_7q6ydcYCG4WObz0G42SVg)

**Lin Guo**

Ph.D. Candidate  
The Systems Realization Laboratory  
School of Industrial and Systems  
Engineering  
University of Oklahoma  
[lin.guo@ou.edu](mailto:lin.guo@ou.edu)

89