

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

NOTE TO USERS

This reproduction is the best copy available.

UMI

UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

**MAPPING OF MULTICAPABLE AND INTERDEPENDENT RESOURCE UNITS IN
PERT/CPM NETWORKS**

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

By

MILAN MILATOVIĆ

Norman, Oklahoma

2000

UMI Number: 9968101

UMI[®]

UMI Microform 9968101

Copyright 2000 by Bell & Howell Information and Learning Company.

**All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.**

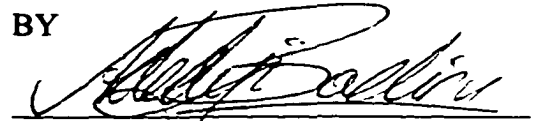
**Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346**

**MAPPING OF MULTICAPABLE AND INTERDEPENDENT RESOURCE UNITS IN
PERT/CPM NETWORKS**


A DISSERTATION

APPROVED FOR THE SCHOOL OF INDUSTRIAL ENGINEERING

BY



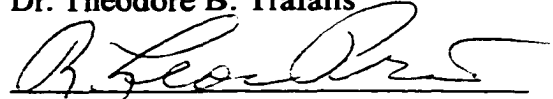
Dr. Adedeji B. Badiru



Dr. Shivakumar Raman



Dr. Theodore B. Trafalis



Dr. R. Leon Price



Dr. S. Lakshmivarahan

**© Copyright by Milan Milatović
All Rights Reserved**

ACKNOWLEDGMENTS

What a relief! If you have gone through all of this, you know what I mean... Otherwise, stretch your imagination.... In addition to the perfect harmony between *Reason and Randomness*, this work would have definitely not converged to its current form without the presence and full support from few people. Certainly the person that has taken the most punches is Dr. Adedeji Badiru, my long time mentor, and now a very good friend. His invaluable “lunch consultations”, expertise, and guidance into all aspects of academic life has provided me with a priceless *career toolbox*. My sincere appreciation also goes to Dr. Shivakumar Raman for friendly chats and advises in situations when I needed them the most. Needless to say, I am honored to have had the opportunity to attend and survive courses by Dr. Theodore Trafalis and Dr. S. Lakshmivarahan. Finally, I am grateful to Dr. R. Leon Price for his time and comments which have enhanced this work.

Wait, I am not finished!!! A good work is well presented work. With that in mind, I am indebted to Dr. Milorad Novicević for sharing many of his magic tricks with me. I also feel very fortunate to have met and collaborated with many of my fellow students and colleagues, especially Dr. Suat Kasap, Dr. Alexander Malyscheff and Dr. Danko Nikolić. Finally, without a kind administrative help from Allison Richardson, Jean Shingledecker, and Jane Smith this work would have been prolonged for several months.

At last, I am blessed to have been introduced to Olga’s heavenly cuisine that has prevented me from completely digesting my spine during long working hours. I extend my thanks to Iričanin family for their friendship throughout all these years, and my parents who still ultimately hold the credit for my being and your reading pleasure of this document.

TABLE OF CONTENTS

ABSTRACT.....	x
I. INTRODUCTION.....	1
II. LITERATURE REVIEW.....	6
2.1 Knowledge Based Systems in Scheduling.....	7
2.2 Uncertainty in Scheduling.....	12
2.3 OR And Dynamic Programming Applications In Project Scheduling.....	15
2.4 Resource Constrained Project Scheduling.....	18
2.5 Branch and Bound Applications in Project Scheduling.....	23
2.6 Cost Considerations in Project Scheduling.....	24
2.7 Activity Duration Issues in Project Scheduling.....	27
2.8 Resource Leveling and Balancing in Project Networks.....	28
2.9 Resource Preferences and Discrimination of Resource Units in Project Scheduling.....	30
III. RESEARCH BACKGROUND.....	35
3.1 Problem Statement.....	41
IV. METHODOLOGY.....	43
4.1 <i>Project Resource Mapper</i> : Classification, Representation and Interdependencies among Project Resources and their Mapping to Project Activities.....	43
4.1.1 Modeling of Resource Characteristics and their Interdependencies.....	46
4.1.2 Dynamic and Resource Type-Specific Varying of Mapping Utility Function.....	55
4.1.3 Resource Time Effective Capabilities and Interdependencies.....	58
4.1.4 Resource Costs and Resource Interdependencies based on Costs.....	64
4.1.5 Resource Preferences and Resource Interdependencies based on their Preferences.....	66
4.1.6 Resource Availability in Resource-Activity Mapping.....	67
4.2 <i>Project Activity Scheduler</i> : Prioritizing and Scheduling Project Activities.....	72
4.2.1 Initial Estimation of Project Activities Duration.....	73
4.2.2 Computing and Dynamic Updating of Activity Priorities....	77
4.2.3 Formulation of the Objective Function for Activity Scheduling and Resource Balancing.....	85

V. SUMMARY.....	93
5.1 Conclusions.....	93
5.2 Major Research Contributions.....	92
5.3 Future Research.....	95
VI. REFERENCES.....	97
APPENDIX A. Complete Heuristic for Dynamic Mapping Resource Units to Project Activities.....	104
APPENDIX B. Overview of PROMAP (Project Resource Mapper) Software....	106
APPENDIX C. Examples of PROMAP Project Input and Output.....	123
Example Project # 1: Input Data.....	125
Example Project # 1: Output.....	147
Example Project # 2: Input Data.....	164
Example Project # 2: Output.....	179
APPENDIX D. Computer Codes for PROMAP Implementation.....	185

LIST OF FIGURES

Figure 1. CRD Network Analysis.....	36
Figure 2. Resource Schedule Chart Based on Earliest Start Times.....	37
Figure 3. Modified Critical Resource Diagram.....	40
Figure 4. Incorporating Resource Availability into Mapping Constraints.....	70
Figure 5. Example of Unbalanced Resource Loading Graphs.....	83
Figure 6. Example of a Project Schedule with the Loading Graph of Resource Type Two Fully Balanced.....	84
Figure B1. PROMAP's Main Menu.....	106
Figure B2. <i>Project</i> Menu Items.....	106
Figure B3. Window for Entering the Basic Project Data.....	107
Figure B4. Use of List Boxes to Display Project Activities and Resource Types.....	108
Figure B5. Text Box for Entering Availability of Resource Types.....	108
Figure B6. Pull-Down Menu that Facilitates the Entering of Activity Precedence Relations.....	109
Figure B7. Window for Entering Functional Dependencies among Resources....	110
Figure B8. Pull-Down Menu Items for Entering Specific Resource Data.....	111
Figure B9. Pull-Down Menu Items.....	112
Figure B10. Run Menu Items.....	113
Figure B11. Choices of User Selected Objectives According to which Resources are to be Mapped.	114
Figure B12. Dialog Box for Entering Optimizing Utility Functions.....	115
Figure B13. Dialog Box for Entering Resource Centralizing Level.....	116
Figure B14. List Box for Selection of Resource Types to be Centralized.....	116
Figure B15. Items under Graph Menu.	117
Figure B16. Example Gantt Chart by PROMAP.....	118
Figure B17. Example of a Resource Loading Graph.....	119
Figure B18. Example of Resource-Activity Mapping Grid.....	120
Figure B19. Resource Units Utilization Bar Chart.....	121
Figure B20. Example Cost Chart for Units of a Project Resource Type.....	122
Figure C1. Example of a Project Manager's Mapping Strategy Input.....	148
Figure C2. Resource-Activity Mapping Grid for the Units of Resource Type 1...	149
Figure C3. Resource-Activity Mapping Grid for the Units of Resource Type 2...	149
Figure C4. Resource-Activity Mapping Grid for the Units of Resource Type 3...	150
Figure C5. Resource-Activity Mapping Grid for the Units of Resource Type 4...	150
Figure C6. Percentage of Resource Units Utilization for Type 1.....	151
Figure C7. Percentage of Resource Units Utilization for Type 2.....	152
Figure C8. Percentage of Resource Units Utilization for Type 3.....	152
Figure C9. Percentage of Resource Units Utilization for Type 4.....	153
Figure C10. Total Relative Resource Units Costs for Resource Type 1.....	154
Figure C11. Total Relative Resource Units Costs for Resource Type 2.....	154
Figure C12. Total Relative Resource Units Costs for Resource Type 3.....	155
Figure C13. Total Relative Resource Units Costs for Resource Type 4.....	155

Figure C14. Resource Loading Graph for Resource Type 1.....	156
Figure C15. Resource Loading Graph for Resource Type 2.....	157
Figure C16. Resource Loading Graph for Resource Type 3.....	157
Figure C17. Resource Loading Graph for Resource Type 4.....	158
Figure C18. Project Activity Gantt Chart.....	158
Figure C19. Project Gantt Chart After Simplifying the Scheduling and Mapping Strategies.....	159
Figure C20. Resource Type 1 Loading Graph After Simplifying the Scheduling and Mapping Strategies.	160
Figure C21. Modified Mapping Strategy.....	161
Figure C22. Project Gantt Chart when Resource Preferences Prevail over Resource Capabilities.....	161
Figure C23. Resource-Activity Mapping Grid for Type 1 when Resource Preferences Prevail over Resource Capabilities.....	162
Figure C24. Resource-Activity Mapping Grid for Type 2 when Resource Preferences Prevail over Resource Capabilities.....	163
Figure C25. Example Mapping Strategy.....	179
Figure C26. Project Gantt Chart for a Schedule Emphasized on Resource Availability.....	180
Figure C27. Resource-Activity Grid for Type 1 of Strategy Emphasized on Resource Availability.....	180
Figure C28. Resource-Activity Grid for Type 2 of Strategy Emphasized on Resource Availability.....	181
Figure C29. Resource-Activity Grid for Type 3 of Strategy Emphasized on Resource Availability.....	181
Figure C30. Relaxed Resource Mapping Strategy Results in Shorter Project Duration.....	183
Figure C31. Resource Group 1 Assignments Resulting from a Change in Strategy.....	183
Figure C32. Resource Group 2 Assignments Resulting from a Change in Strategy.....	184
Figure C33. Resource Group 3 Assignments Resulting from a Change in Strategy.....	184

LIST OF TABLES

Table 1. Example Project Data.....	39
Table 2. Example Representation of Time-Effective Capabilities and Interdependencies to Seven Project Activities.....	74
Table 3. Initially Estimated Activity Durations.....	76
Table C1. Basic Project #1 Data.....	125
Table C2. Time-Effective Capabilities For Resource Type 1.....	126
Table C3. Time-Effective Capabilities For Resource Type 2.....	127
Table C4. Time-Effective Capabilities For Resource Type 3.....	129
Table C5. Time-Effective Capabilities For Resource Type 4.....	132
Table C6. Preferences for Resource Type 1.....	134
Table C7. Preferences for Resource Type 2.....	135
Table C8. Preferences for Resource Type 3.....	137
Table C9. Preferences for Resource Type 4.....	139
Table C10. Costs for Resource Type 1.....	141
Table C11. Costs for Resource Type 2.....	142
Table C12. Costs for Resource Type 3.....	144
Table C13. Costs for Resource Type 4.....	146
Table C14. Basic Project #2 Data.....	164
Table C15. Time-Effective Capabilities for Resource Group 1.....	165
Table C16. Time-Effective Capabilities for Resource Group 2.....	166
Table C17. Time-Effective Capabilities for Resource Group 3.....	168
Table C18. Preferences for Resource Group 1.....	170
Table C19. Preferences for Resource Group 2.....	171
Table C20. Preferences for Resource Group 3.....	173
Table C21. Time Availability for Resource Group 1.....	176
Table C22. Time Availability for Resource Group 2.....	177
Table C23. Time Availability for Resource Group 3.....	178

ABSTRACT

Globalization of business activities, deregulation of industries, and technological advances have greatly contributed to the increasing importance of project scheduling approaches in knowledge rich economy. In this new economy, multifunctional capabilities are becoming one of the most critical resource attributes that need effective appropriation in resource constrained scheduling. As a result, the traditional scheduling of project activities must be complemented with attentive mapping of human, social and technical resources to interact in value creating ways, while still meeting the cutting edge of both analytical rigor and managerial relevance. Therefore, the primary objective of this research is the development of a generic project scheduling model that incorporates 1) resource characteristics, such as preferences, time-effective capabilities, costs and availability of project resources, 2) possible performance interdependencies among different resource groups, and proceeds to map the most relevant resource units to each newly scheduled project activity. The principal challenge in this generic model development is to make it applicable to realistic project environments which often involve resource units with characteristics which may vary across activities, as well as within a single activity relative to specific interactions among resources. The scope of this research challenge increases when the actual duration, cost, and successful completion of a project activity are considered to be potentially resource driven and dependent on the choice of particular resource units assigned to it. Such successive consideration of resource characteristics in resource allocation to activities is of extreme

practical relevance because it may likely also improve overall project duration, quality, and cost.

The model developed in this study first schedules qualifying activities at each decision instance, and then dynamically maps available and the most relevant resource units to them. Before the resource-activity mapping occurs, resource units are classified into groups based on their interactive dependencies. Those units, whose preferences or performance on an activity depend on their interaction with units from other groups, are mapped last. The actual mapping of resource units to activities is accomplished according to a pre-specified arbitrary utility function which incorporates one or more of the above resource characteristics. Due to the dynamic nature of project schedules, the utility function may be held fixed throughout the mapping or be allowed to vary with time by filtering out some of its additive components not associated with current scheduling time. Similarly, the utility may be allowed to differ for different resource groups by filtering out its components not associated with currently mapped resource group. The procedure progresses until all project activities are scheduled and resource units assigned to each of them. This model represents a crucial initial step towards a comprehensive resource-activity based integration in project scheduling, which is a particularly valuable managerial tool in knowledge-intensive industries.

I. INTRODUCTION

Traditional project scheduling techniques generally provide graphical and analytical solutions which are primarily based on project activities. Resources, if limited in quantity or availability, then impose appropriate constraints in scheduling of activities. The actual assignment of resources to activities depends on the type and functionality of resources themselves. In cases when resources have pre-specified assignments and responsibilities towards one or more activities, their allocation is concurrently performed with the scheduling of applicable activities. In other cases, an activity may only require a certain number of (generic) resource units of particular type(s), which are assigned after the scheduling of the particular activity. These two approaches coarsely represent the dominant paradigms in project scheduling. The objective of this research is to propose a new model and strategy which will shift these paradigms to facilitate a more refined guidance for allocation and assignment of project resources. In other words, there is a need for tools which will take into account behavior, multi-capability, interdependencies, and bundling of resources and provide for effective resource tracking, control, interaction, and, most importantly, resource-activity mapping.

The methodology developed in this research is based on several elemental modeling assumptions. The principal assumption is that project environments often involve multi-capable resource units with different characteristics. This is especially the case in knowledge intensive settings and industries which are predominantly staffed with highly trained personnel. The specific characteristics considered were resource preferences,

time-effective capabilities, costs, and availability. Each resource unit's characteristics may further vary across project activities, but also within a single activity relative to interaction among resource units. Finally, resource preferences, cost, and time-effective capabilities may also independently vary with time due to additional factors, such as learning, forgetting, weather, type of work, etc. Therefore, although we don't exclude a possibility that an activity duration is independent of resources assigned to it, in this research, we assume that it is those resource units assigned to a particular activity that determine how long it will take for the activity to be completed. This is, again, somewhat contrary to a common practice, where an activity duration is pre-specified before having any resource units assigned to it.

Based on the above assumptions, a comprehensive model has been developed and implemented in this research to schedule projects by alternatively executing two specific procedures. The first one prioritizes and schedules activities based on the current availability of resources. The second procedure then immediately maps the most relevant of the available resource units to the newly scheduled activities. The *activity scheduler* prioritizes and schedules activities based on some of their basic attributes, which may include attempts to centralize selected resource loading graphs based on activity resource requirements. The particular attributes considered are the number of activity successors, initially estimated shortest expected activity duration, and dynamically updated amount of depleted activity slack. In addition to their attributes, activities may also be prioritized and scheduled based on their resource requirements with respect to a manager's attempt to centralize certain pre-specified resource loading graphs. The *resource mapper* then

considers the above resource characteristics, incorporates interdependencies among resource groups or types, and maps the available resource units to newly scheduled activities according to a project manager's or analyst's pre-specified utility (objective) function. Although the *activity scheduler* must ensure that enough resource units are available for each candidate activity before it is scheduled, the *resource mapper* decides which particular of those available units should be assigned to which activity. Since project scheduling is a dynamic process, this utility function may be held constant throughout the process, or allowed to vary with time. For example in the early scheduling stages, a project manager may be more interested in satisfying resource preferences as opposed to later project stages, where project's timely completion may require greater attention on resource time-effective capabilities. The utility function may further differ for various resource groups (types) or specific units. For this purpose, *Kronecker's delta* as well as *window functions* are used to keep the desired parts of the utility function and filter out those additive components of the utility which are not associated with a current time or resource group.

The scheduling strategy as illustrated above promotes a more balanced and integrated activity-resource mapping approach. Mapping the most qualified resources to each project activity, and thus preserving the values of resource, is achieved by proper consideration of resource time-effective capabilities and costs. By considering resource preferences and availability which may be entered in either crisp or fuzzy form, the model enables consideration of personnel's voice and its influence on a project schedule and quality. Furthermore, resource interactive dependencies may also be evaluated for each

of the characteristics and their effects incorporated into resource-activity mapping. Finally, by allowing flexible and dynamic modifications of scheduling objectives (utility), the model permits managers or analysts to incorporate some of their tacit knowledge and discretionary input into project schedules.

The model has been implemented in a software prototype, with its code, input format, and sample outputs illustrated in the appendices. The output consists of five types of charts. The more traditional ones include project *Gantt chart*, and *resource loading graphs* for all resource groups or types involved in a project. More specific graphs include *resource-activity mapping grids*, *resource utilization* and *resource cost* bar charts. Based on inputted resource characteristics, their interdependencies, and the form of the objective, the *resource-activity mapping grid* provides a decision support in terms of which units of each specified resource group should be assigned to which particular project activity(ies). Therefore, the *resource-activity grids* are, in effect, the main contributions of this study. *Unit utilization charts* track the resource assignments and provide a relative resource usage of each unit relative to the total project duration. *Resource cost* charts compare total project resource expenditures for each resource unit.

The remaining of this dissertation is organized as follows: Chapter II presents extended literature review that has been relevant and influential on this research. Chapter III discusses the research background and the need for new approaches and models. Chapter IV provides a detailed description of the model. Chapter V summarizes major research contributions and provides recommendations for future research directions. Appendix A

presents an algorithmic summary of the model proposed and implemented in this research. A brief overview of software developed to support the model is given in Appendix B. Appendix C presents two example projects, their input data, and elaboration of outputs relative to given objectives. Finally, the computer code used in the model implementation is listed in Appendix D.

II. LITERATURE REVIEW

The process of scheduling is one of the basic constituents of every manufacturing, production, management, and computer environment. Regardless of the environment in which it takes place, scheduling is defined as allocation of (usually limited) resources over time to perform a set of planned activities. A survey of some 400 top contractors in construction, showed that 96.2% of them still use Critical Path Method (*CPM*) to some degree for scheduling (Mattila and Abraham, 1998). Another survey of Associated General Contractors of America revealed that scheduling is still the most important technological component that needs improvement (Mattila and Abraham, 1998). During the development of an expert system for job-shop scheduling, it was discovered that human schedulers spend about 80-90% of their time in only identifying the constraints, and only about 10-20% for the actual scheduling (Liebowitz and Potter, 1995). Park et al. (1996) affirm that the main problems in automation of production scheduling is the lack of an explicit representation scheme of scheduling knowledge to aid in the communication between human schedulers and systems analysts.

In general, scheduling problems are associated with numerous conflicting objectives and constraints, and an immense number of combinatorial options and selections. It is traditionally an NP hard problem, that is, it cannot be solved by a polynomially bounded algorithm. Thus, the challenge for the researchers remains open.

Studies in both operations research (*OR*) and artificial intelligence (*AI*) have contributed their portion of techniques towards scheduling. Traditional *OR* scheduling methods involve linear programming, branch and bound, and Tabu search. Contributions by *AI* come from expert systems, fuzzy logic (as a special case of expert systems), neural networks, simulated annealing, genetic algorithms, constraint satisfaction, hill climbing, and connectionist methods. Thus, an additional problem a scheduler may face is having to make a choice of mapping a particular scheduling problem to an adequate technique. Tsang (1995) argues that the knowledge of which technique to apply and when, is at least as critical as the expertise in the individual technique itself.

Previous literature surveys on *AI* applications in scheduling can be found in Atabakhsh (1991), Tsang (1995), and Wiers (1996). The following sections discuss some of the recent applications of *AI* techniques, followed by the advances and applications of *OR* in scheduling. Expert and knowledge-based systems, including uncertainty in scheduling are discussed next.

2.1 Knowledge Based Systems In Scheduling

As one of the oldest of techniques, expert systems have been widely used in scheduling for many years. The popularity of expert systems stems primarily from the simplicity of their implementation and understanding, since their structure is almost solely rule based. A domain knowledge is generally embedded into an expert system in terms of rules and a scheduler. The rules indicate which of the tasks or resources are eligible for scheduling,

while the scheduler then attempts to resolve possible conflicts and satisfy any constraints. A major difficulty in the implementation of expert systems (not only in scheduling, but in general), is the knowledge extraction from human experts. In addition, the actual scheduling conflict resolver is also difficult and non trivial to develop. Many attempts, however, to develop expert systems to tackle specific and custom problems exist, and some of the latest attempts are described in this section.

As one of the primary and most executed operations at NASA sites, scheduling has prompted a great need for development of more generic expert scheduling systems. Liebowitz and Potter (1995) investigated objectives, requirements, resources, constraints, processes, and scheduling domains for development of a generic scheduling system for NASA centers, particularly for missions planning. Their previous survey of 250 papers on expert scheduling systems in 1993 enumerated about 24 significant scheduling approaches that were based on optimizing algorithms, about 20 different heuristics, and two hybrid methods that incorporated both heuristics and algorithms. In their literature review, they have come up with about 20 different objectives that are to be considered by NASA's mission scheduling.

Liebowitz and Potter (1995) stressed several points necessary for the development of a generic expert scheduling system for NASA purposes, but which are also relevant to other industrial areas. First, regarding the objectives of scheduling, it is imperative to maximize scheduled number of requests while minimizing "unhappiness" of a scheduler. In addition, all (or a vast majority of) constraints must be satisfied, while the safety and

performance is maintained. Some of the objectives included due dates satisfaction, satisfying maximum number of constraints, balancing loads among different stages of assembly operations, maximizing the scheduling of high priority events over low priority ones, minimizing the number of tardy jobs, minimizing inventory costs as well as project duration, optimization of resource allocation, etc. Some major scheduling requirements require a hierarchical architecture, ability to quickly, effectively and automatically perform rescheduling, need for good user and system interface and portability, and a need for having a variety of scheduling techniques available. Hierarchical architecture implies that a part of overall scheduling is propagated to lower level schedules who have a control of their own limited areas or departments. All requirements were grouped into eight groups, some of them being general requirements, resource/constraints requirements, activity requirements, output requirements, system interface requirements, etc. Resources were classified as spatial (ones where time is a significant factor, such as spacecraft orbits or viewing periods), and non-spatial such as cranes, crews, machines, etc. Constraints were classified as precedence constraints (due to ordering of activities), synchronization constraints, and non-time dependent constraints, such as capacity, safety, etc. A long list of resources and constraints is also provided in the paper.

Hori et al. (1995) show how a composable scheduling knowledge can be elicited from existing expert systems, thus enabling knowledge sharing and reuse. The authors propose three problem solving patterns as abstract templates for component elicitation: *divide and merge* (divide a given problem, invoke another component to receive solutions to divided subproblems and merge them into a schedule hypothesis); *transform and restore*

(reformulate a problem structure, invoke another component, and restore the schedule hypothesis obtained to the original problem space); *check and modify* (find an unexpected situation such as a constraint violation in a schedule hypothesis, and modify it).

Recently, Sauer and Bruns (1997) have proposed a generic framework to facilitate construction of knowledge based scheduling systems. Their framework is based on two design principles: (1) combination of standard computer science components with knowledge based concepts (heuristics, algorithms) and declarative knowledge representation, and (2) explicit and transparent representation of knowledge that allows for reuse and adaptation of scheduling algorithms. The authors argued that all scheduling systems must possess an easy adaptation and advocate for a reusable representation of scheduling knowledge. This stems from the fact that many advanced algorithms have been designed for only specific problem instances, which do not allow reusing of any components in future systems or transferring much of an algorithm into more general scenarios.

Ntuen and Park (1995) have experimented in merging *OR* and *AI* tools and proposed a hybrid scheduling model for approaching non-structured scheduling problems (*NSSP*). In *NSSP*, resources possess at least one, but generally more skills to perform a task. Example would include a car mechanic who does a variety of tasks from tire repair to engine rebuilding. Ntuen and Park (1995) have proposed their methodology to scheduling of aircraft turnaround functions (*ATF*). Examples of *ATF* are express plane inspection for leaks and/or damage, refueling, ammunition loading and arming, etc. It is of interest to

coordinate these functions in a minimum time span. To accomplish it, Ntuen and Park developed a model, named *Task Oriented Planner*, which is also of object oriented structure. During a job schedule, the knowledge processing environment dynamically creates a node for each resource which carries its class attributes. This method of dynamic node creation allows for potential job preemption, resumption, as well as dealing and assignment of idle resources. Thus, once a planning is achieved, the scheduling module is activated which creates sub-hierarchies of knowledge bundles to cluster jobs and resources according to priorities.

A joint project by Korea Advanced Institute of Science and Technology (*KAIST*) and Daewoo from 1991 to 1993 that involved development of an intelligent comprehensive scheduling system for shipbuilding has been documented in an article by Lee et al. (1995). The result was a Daewoo Shipbuilding Scheduling (*DAS*) expert system launched in January of 1994, which had significantly improved the production and quality of the facility. Similar to the previous papers, this model was also based on hierarchical system architecture.

Papers by Lee and Wu (1995) and Liou and Wu (1996) incorporated experts systems into scheduling of academic courses. Lee and Wu (1995) designed their scheduler based on a *desirability map* that indicates the degree of ‘wishfulness’ for a class to be assigned a specific time block. The number obtained is a combination of a preference degree, instructor’s priority, and a course weight itself. Conflicts were resolved by using a breath-first search in conflict trees. A finished schedule allowed for interactive changes.

Rules in the expert system were extracted from the knowledge of faculty and staff. The system was implemented in *CLIPS*, a C language based Integrated Production System, established on forward chaining principles. The system had a total of 556 rules and was actually tested at the National Sun-Yat University in Taiwan.

Liou and Wu (1996) proposed an alternative implementation of expert systems for academic course scheduling. Courses, instructors, classrooms, and time periods were represented as basic objects, each having a set of attributes assigned to it. The attributes of each instructor included name, I.D., position, mastering courses, list of preferences, etc. The authors further developed a scheme of depicting objects and relationships among them. The proposed scheme was graphically represented with relationships grouped as “pyramids”, with the vertices being particular instructors, courses, and time periods, and the edges being their interrelationships. Thus, each pyramid was interpreted as an assignment of a course to an instructor for a particular classroom during a particular time period. For example, credit hours taken by a particular instructor could easily be assessed by accessing all edges sharing a particular vertex representing that instructor.

2.2 Uncertainty in Scheduling

Uncertainty in scheduling parameters has been considered and modeled extensively within the past decade. Hapke et al. (1994) proposed a complete decision support system for software project scheduling. The purpose of the so called Fuzzy Project Scheduler (*FPS*) was to allocate resources, (primarily software engineers) among planned activities,

such as system design, GUI design, implementation of modular components, and subsequently their integration. The uncertainty was assumed in activity durations, ready times, and due dates. The actual system consisted of not only one scheduling heuristic. Instead, activities were chosen based on one out of 12 different heuristic rules. In addition, in order to generate even greater variety of feasible schedules, the authors also implemented five different mutations to each of the 12 priority lists. Thus, the total of 60 different schedules were obtained from which the authors suggest selecting one with the best solution. Since the solutions were represented in fuzzy numbers, one of the previously available means was used to compare the magnitude of fuzzy number obtained. Although the system results were characterized by possible high degrees of uncertainty, that was exactly the purpose of it. In other words, the system's solution did incorporate both optimistic and pessimistic scenarios, carried them all the way through, and accordingly, yielded similar output which contained a full possibility distribution.

Nasution (1994) proposed a more comprehensive method for carrying calculations in fuzzy *CPM*. As opposed to previous research on this matter which either considered earliest *or* latest allowable project times, Nasution proposed more relaxed methodology which incorporated interactive subtraction of fuzzy times in the backward *CPM* calculations, thus, enabling him to compute fuzzy slacks of all network activities. Since fuzzy numbers have areas associated with them, Nasution suggested that any negative parts of fuzzy numbers (obtained by fuzzy subtraction) should be ignored since they likely carry no useful information.

Lorterapong (1994) extended fuzzy scheduling heuristics to incorporate resource allocation within projects. The heuristic mainly breaks down the activities into subsets at each time instant when a resource conflict occurs. Then, a simple procedure based on activity slacks is used to evaluate each activity subset and determine its impact on project duration. The author then extended this concept into a fuzzy space and incorporated vagueness in the specification of time parameters.

Wu and Hadipriono (1994) used fuzzy logic to evaluate different factors on activity durations in construction projects and scheduling. One of the prime objectives in project management is to estimate duration of a project. On a smaller scale, estimation of activity duration within a project may also be a non trivial task. In construction scheduling, there are numerous factors that may and do affect activity durations. Some of the most important ones include site location and condition, climate and weather (weather being an instance of a climate), resources, management performance, material supply, equipment performance, labor performance, etc. Too optimistic schedule may result in project delays and penalties to the contractor. On the other hand, too pessimistic calculations may produce resource idleness and increase in overhead costs. Thus, the authors proposed an activity duration decision support system that applies fuzzy modus ponens (forward chaining or data driven inference) to capture the impact of the above factors in activity durations. It is interesting to note that the authors used a new representation of fuzzy numbers to quantify linguistic descriptions of the above factor values. More specifically, the authors proposed *angular fuzzy sets* to model the system.

Angular fuzzy sets were first proposed in 1990 by one of the authors, who used a semicircle from $-\pi/2$ to $+\pi/2$ to represent the true values in the universe of discourse (universal set over which fuzzy numbers are defined). The angle between a straight line from the center of the circle and the horizontal represents a particular truth value.' The authors do provide some operational and arithmetic possibilities using angular fuzzy numbers.

2.3 OR And Dynamic Programming Applications In Project Scheduling

In their recent review of current project scheduling models and methods, Brucker et al. (1999) attempt to standardize a common notation and a classification in project scheduling, as well as close the still open gap between project scheduling and job shop scheduling as its special case. The authors divided the methods into single-mode cases, multi-mode cases, resource constrained problems with time lags, models with nonregular objectives, and models with stochastic activity durations.

Branch-and-bound and heuristic approaches were the most common methods for solving single-mode cases. Patterson et al. (1989) proposed a case of branch-and-bound algorithm commonly referred to as the precedence tree. At each iteration, the procedure determines a set of currently scheduled activities and those that have just qualified for scheduling. One of the eligible activities is then selected and the next starting time is computed. Once the dummy termination node is encountered, a complete schedule is

said to be found, and the procedure backtracks to the previous level and selects an untested eligible activity. When all the eligible activities have been tested, the procedure backtracks again to the previous level, until each branch from the root to a leaf has been examined, and which in effect represents the permutations of the activity set that is precedence feasible.

Delay Alternatives is another branch-and-bound procedure proposed by Christofides et al. (1987), which at each time decision instance t_c , considers eligible activities, and subjects them to resource constraints. Those activities whose requirements may be satisfied given the current constraints and resource availabilities are scheduled, while the other activities are delayed until the next decision instance. Once the schedule is completed, the procedure backtracks and reconsiders the delayed activities. This method, as opposed to the *precedence tree*, considers scheduling of activities in batches (as opposed to one at a time), and it first computes the decision instance before deciding on eligible activities. Variations to the above procedures include the method of *Extension Alternatives* as proposed by Stinson et al. (1978) and the method of *Block Extensions* by Mingozzi et al. (1998).

Heuristic methods that were initially proposed were priority-rule based, and had (still do) advantage of being intuitive, easily implementable and of affordable computational effort. Recent heuristics, however, in order to improve the objective, are shifting more towards local constraint based analysis, truncated branch-and bound, and integer programming heuristics (Brucker et al., 1999).

When a project manager is in control of being able to vary a project duration according to how much penalty he or she is willing to pay for, we have a so-called time-cost trade off problem. In effect, this type of problems are a part of multiobjective set up with distinct budget and deadline problems merged together. In general, it is desirable to solve a multiobjective problem for all possible scenarios of costs and deadlines, before making a decision.

Fulkerson (1961) and Kelly (1961) proposed an activity on arc network and algorithm which iteratively calculates a project cost curve, by a maximum flow computation which takes the capacities as the slopes of linear cost functions of critical activities. Although many improvements to this procedure have been proposed by today, the currently most promising algorithms still rely on dynamic programming. Some of the alternative approaches have been proposed by Bein et al. (1992) and Demeulemeester et al. (1996).

In multi-mode cases, each activity may be executed in one of several modes. The number of different durations of a single activity that depend on the number of resource units assigned to that activity will define the number of modes. There are exact and heuristic procedures to approach problems of this sort. The exact algorithms are extensions of single mode algorithms, such as the *precedence tree* which was adapted to a multi-mode case by Sprecher and Drexel (1998). Modifications to *delay alternatives* method to accommodate for multi-modality were also proposed by Sprecher (1997).

Heuristic procedures have also been proposed for solving multi-mode scheduling problems. Some of the methods are documented in Drex1 (1991), and Slowinski et al. (1994).

In 1998, Herroellen et al. published another survey of resource constrained project scheduling techniques. Their emphasis was on depth-first branch-and-bound procedure for preemptive resource-constrained scheduling models with generalized precedence relations, and models that maximize the net present value of projects.

Some of the more significant papers and works on resource-constrained project scheduling are discussed next.

2.4 Resource Constrained Project Scheduling

Ulusoy and Ozdamar (1989) conducted a study in which they investigated the influence of actual project networks and/or resource characteristics on performance of heuristic rules. A factorial design was used to classify problem types successfully solved by particular heuristics. In addition to investigating six previously published heuristics, the authors also proposed a new heuristic, named *Weighted Resource Utilization Ratio And Precedence (WRUP)*, defined as:

$$Priority = w(p)n(ij) + w(r) \sum_k \frac{r(ijk)}{R(k)}$$

where:

$w(p) \equiv$ precedence weight

$w(r) \equiv$ resource utilization, $[1 - w(p)]$

$n(ij) \equiv$ number of immediate successors of activity ij (assuming activity on node network)

$R(k) \equiv$ units available of resource type k per period.

Network/resource characteristics investigated in the study were the *aspect ratio* (the ratio between the number of critical and non-critical activities), *complexity ratio* (the ratio of the number of activities to the number of network events), *resource utilization factor* which reflects global resource usage on a critical path, and *dominant obstruction value* as an indicator of resource shortage. The experiments showed that *WRUP* heuristic outperforms three out of six existing techniques, and has additional computational advantages over the remaining heuristics, mainly in terms of the number of times a *CPM* network needs to be resolved.

Khattab and Choobineh (1991) evaluated several of the existing priority rules and proposed eight new rules, which they incorporated into a new scheduling heuristic, referred to as the *Search* method. *Search* method solves each scheduling problem eight times, once for each of the eight proposed priority rules. The method would then recommend the schedule resulting in the shortest project duration. Due to its hybrid

nature, the method produced schedules of shortest duration the most often. The eight priority rules used in the *search* method are:

1.
$$\frac{\text{activity time} + \text{time of all successors}}{\text{activity resource} + \text{resources of all successors}}$$
2. *total time of all successors*
3. $(\text{activity time} + \text{time of all successors}) - (\text{total time of predecessors})$
4. *activity time + time of all successors*
5.
$$\frac{\text{activity time} + \text{time of immediate successors}}{\text{number of immediate successors}}$$
6.
$$\frac{(\text{time of immediate successors})/(\text{resources of immediate successors})}{(\text{activity resource})/(\text{activity time})}$$
7. *activity resource*
8.
$$\frac{\text{activity time}}{\text{activity resources}}$$

Although no priority rule above could be successfully used by itself, their combination did outperform other single rule priority measures investigated at the time.

Davis et al. (1992) formulated a multiple criteria project scheduling problem with objectives of minimizing project completion time as well as minimizing the over-utilization of resources. The authors introduced a decision support framework and the interactive procedure allowed a decision maker to iteratively observe and evaluate tradeoffs between different objectives. Although restricted in size of problems it could handle, the proposed procedure performed better than the existing goal programming methods, mainly because the interaction between the decision maker was facilitated and provided a better reflection of preferences and objectives.

Minciardi et al. (1994) proposed an event driven method and constructed a project schedule by solving a sequence of successive instances of the same subproblem. Then, additional heuristics were employed to generate feasible schedules for subproblem instances in consideration. This led to a final schedule which determined a set of decisions for assigning and sequencing of tasks over available resources. These decisions were further inputted as constraints in the final timetabling optimization.

Nowicki and Smutnicki (1994) presented an alternative decision support system, but its implementation involved both so-called soft and hard constraints. Soft constraints could be violated, and hard constraints were non-violated. The inclusive heuristic then computed in deterministic time increments the set of schedulable tasks.

Considering the limitation of daily consumption of project resources, Ulusoy and Ozdamar (1994) proposed a heuristic, referred to as the *Local Constraint Based Analysis*

(*LCBA*). *LCBA* is a two stage procedure, where the first stage checks whether all activities have a sufficiently wide time span during which they can be run, and the second stage employs a set of rules to prioritize qualifying activities and resolve any existing resource conflicts.

Boctor (1996) presented a heuristic for non-preemptive project scheduling problem with renewable resources and multiple execution modes. At each iteration, the proposed procedure (does not schedule one activity at the time, but rather) evaluated schedulable combinations of activities (including activity durations versus the number of resource units employed) and selected a combination that maximizes a prespecified objective. Whenever a feasible schedule existed, the procedure guaranteed its generation. The heuristic was tested on a set of 240 randomly generated projects and it outperformed four of the previously most acknowledged procedures.

Morse et al. (1996) evaluated resource constrained project networks by applying combinations of at least two heuristics that would produce minimum project duration. The authors selected 10 simple and existing priority rules and applied to the set of 108 previously generated project network problems. The heuristics used were *shortest job first*, *first come first served*, *latest finish time*, *minimum slack first*, *minimum early finish*, *maximum slack first*, *longest activity first*, *ACTIM*, *ACTRES*, and *resource over time (ROT)*. The project durations were computed by a package network program with separate subroutines that allocated resources. An additional algorithm was then utilized to

determine which combination or subset of the above priority rules would yield the shortest project duration.

Icmeli-Tukel and Rom (1996) proposed two models for scheduling resource constrained projects with objectives of maximizing project quality. The quality was measured by the amount of rework required and associated additional cost corresponding to it. The two models were formulated as mixed integer programming problems except that they contained additional constraints and variables in the objective function.

One of the most exploited *OR* procedures in project scheduling is the application of branch-and-bound technique. A selection of papers in the area is briefly summarized in the next section.

2.5 Branch and Bound Applications in Project Scheduling

Drexl (1991) used a branch-and-bound dynamic programming method which incorporated Monte Carlo method for resolving conflicts between activities that compete for limited resources. Carraway et al. (1991) extended the notion of dynamic resource allocation to multiple interdependent projects. Li and Willis (1992) proposed an iterative project scheduling, which during the procedure, scheduled a project both forwards and backwards until the completion time could not be further improved. Initially, a project was scheduled forward to compute a “forward” schedule. The duration obtained was then used as a starting point for the backward schedule. The process continued until no further

improvement could be achieved. Belhe and Kusiak (1993) applied constrained project scheduling problem in scheduling of design activities. However, instead of resorting to the traditional branch-and-bound method, the authors approached the problem using the *beam search* heuristic. This method is similar to branch and bound, except that *beam search* heuristically determines the best paths and ignores the rest of the search space. De Reyck and Herroelen (1998) incorporated branch-and-bound method for solving resource constrained project networks with generalized precedence relations. Nazareth et al. (1999) applied breadth first approach to solving resource constrained networks. Additional dynamic programming approaches are also found in Elmaghraby (1993) and Brucker et al. (1999).

2.6 Cost Considerations in Project Scheduling

Many papers address the issue of minimizing costs in project networks. Wu and Li (1994) proposed a strategy of applying the cut set theory of networks in order to determine activity sets to be shortened and the maximal shortening time such that the overall project duration is reduced at minimal costs. The authors first applied the minimal cut set method to select the set of activities to be crashed. This was accomplished by first computing the conventional critical path, then eliminating all non-critical activities, and finally identifying the minimal flow cut set. After the crashing activity set was identified, Wu and Li proposed a new application of *cut set parallel network*, where they used the cut set parallel difference method to determine the maximal permitting crashing time.

Analyzing instances of high interest rates and limited capital, Sung and Lim (1994) considered scheduling resource constrained project networks with availability restrictions on capital and renewable resources. The authors considered resource-duration interactions in order to maximize the net present value of a project. Their proposed heuristic consisted of two phases. In the first phase, the initial schedule was determined and its associated net present value. The next phase then attempted to improve the initial solution by solving all decomposed subproblems.

Demeulemeester (1995) further presented an optimal technique for minimizing resource availability costs in time constrained project networks. The author perturbed the basic *resource constrained project scheduling problem (RCPSP)* which searched for a solution to the shortest project duration constrained to given project data and resource availability. The newly defined problem presets the actual project duration and attempts to find a feasible schedule subject to project data and available constraints. Demeulemeester (1995) formulated the problem as follows:

$$\min \sum_{k=1}^m c_k(a_k)$$

subject to:

$$f_i \leq f_j - d_j \quad \text{for all } (i, j) \in H$$

$$f_1 = 0$$

$$f_n \leq T$$

$$\sum_{i \in S} r_{ik} \leq a_k \quad \text{for } k = 1, \dots, m \quad \text{and} \quad t = 1, \dots, f_n$$

The problem as formulated above is referred to as the *resource availability cost problem (RACP)*. The traditional resource-constrained project scheduling problem (*RCPS*) would not have imposed $f_n \leq T$ as its constraint, and the objective would be to minimize f_n . The author however, did employ existing techniques for iterative solving *RCPS* and proposed their modification for solving *RACP*.

Demeulemeester et al. (1996) further presented two algorithms for optimally solving discrete time/cost trade off problems. The algorithms were based on dynamic programming, and were implemented with respect to three different objectives: (1) completing the project as early as possible given the limitations of a single nonrenewable resource; (2) minimizing resource usage given the constraints on total project duration; and (3) computing total project time/cost trade off function, given the constraints on both resources and total project duration.

De Reyck and Herroelen (1997) extended previous ideas and considered scheduling problems with generalized precedence relations with the objective of maximizing net present value. As a solution, the authors explored a depth-first branch-and-bound algorithm in which the original project network is represented by nodes in the search tree which also incorporated additional precedence relations. Resource conflicts were

approached through the concept of minimal delaying modes, and rules were employed to filter out portions of the search tree.

2.7 Activity Duration Issues in Project Scheduling

The effects of variable or erroneous activity durations on project networks have also been addressed in the literature. Sipos (1992) gave a thorough set of definitions and concepts behind the analysis of activity durations in projects. Leachman and Kim (1993) proposed and developed procedures that compute earliest and latest intensity curves of dependent activities for correct modeling of variable duration activities and generalized precedence relations. Yang (1996) identified uncertainties in projects as difficulties in estimation of work contents of activities, unexpected wear conditions and delays, need for rework, delivery failures and absenteeism. The author then formulated a research study to examine the effects of erroneous estimation of activity durations in three project environments, depending on the strength of the precedence relations, level of resource availability, and magnitude of errors in estimating activity durations. Yang also quotes the statistics that by the time of middle 1980's many companies used less than 10% of the advanced features available in their project management software, and out of 35 project management software packages available by 1986, only two were found capable of automatically generating feasible project schedules.

2.8 Resource Leveling and Balancing in Project Networks

Probably the two most important elements of any type of project, regardless of its scope and area, are the proper planning and the amount and availability of resources necessary for its completion. Thus, one of the most popular approaches to efficient resource handling and cost reduction is the reduction in variability of resource usage. High fluctuations in resource loading and frequent hiring and firing of employees traditionally reduces short term project feasibility. Many overhead costs, such as administrative procedures and training periods occur when hiring resources which may not get a proper chance and time to generate pay-off revenues if being fired not long enough after being initially contracted. Finally, a management practicing a frequent "hire-fire" policy might not be able to attract as much of high quality resources (Seibert and Evans, 1991).

Resource leveling and allocation have been the focus of project management studies for almost four decades now. A pioneering work in this area has been presented by Burgess and Killebrew in 1962 who proposed and implemented a heuristic that minimizes the sum of squares of activity levels. Later, a model that enumerated all possible solutions and found an optimum was presented by Ahuja in 1976. An obvious problem with this approach was that as the number of non-critical activities increased, the combinatorics of the problem became too complex. For example, a simple project of only 15 non-critical activities, each having a slack of only 10 time units would have exactly 10^{15} possible combinations! More lately, Easa (1989) formulated an integer *LP* which guaranteed optimal solution but only for small to medium sized projects. In his work, the objective

was to minimize absolute deviations between resource requirements and desired rectangular loading level.

Seibert and Evans (1991) considered several serial methods for time constrained resource leveling. Serial methods rank activities based on some user defined rules and then attempt to schedule them within the allowed resource constraints. If that is not possible, the methods do exceed the constraints (since the overall project duration is held fixed), but as uniformly as possible. They also propose a simple measure of how successful a particular resource leveling is, by defining a utilization factor as a ratio of resource usage level versus initial (unleveled) loading.

As a follow up on the above two articles, Bandelloni et al. (1994) proposed a resource leveling technique based on *non-serial dynamic programming* modeling. Although not completely relying on full enumeration, this method is also limited to small or medium sized project networks.

Recently, another application of integer *LP* formulation in resource leveling for linear schedules was developed by Matilla and Abraham (1998). *Linear projects* (frequently arising in construction) contain repetitive activities which need to be performed on several different locations. Thus, a proper distribution and work continuity of resources must be obtained. Konstantinidis (1998) further proposed a model that would balance resource loading graphs for nonrenewable, renewable, and doubly constrained resources by eliminating as many interruption periods and costs and shifting activities between

their earliest start and latest completion times. Renewable resources are limited per period, but are becoming again available each new period. Nonrenewable resources have a fixed number of units allocated for the entire project. Doubly constrained resources are constrained with respect to both per period and per total project basis.

2.9 Resource Preferences and Discrimination of Resource Units in Scheduling

Literature also presents work on scheduling projects by accounting for worker (resource) preferences, qualifications, and skills, as decisive factors to their allocation. Roberts (1992) argued that information sources for project planners and schedulers are increasingly nonhuman, and stressed that planners must keep computerized tools for project management and scheduling in line and perspective with human resources used by projects. In other words, the author warns that too much technicalities may prompt and mislead the managers into ignoring human aspects of management.

Franz and Miller (1992) considered a problem of scheduling medical residents to rotations, and approached it as a large scale multi-period staff assignment problem. The objective of the problem was to maximize the residents' schedule preferences while meeting the hospital's training goals and contractual commitments for staffing assistance (Franz and Miller, 1992). Thus, each resident's schedule is different depending on particular interests and departmental requirements. The authors formulated a problem as a zero-one integer problem with a linear objective function indicating the preference weight of a doctor i being assigned to rotation j during month k . The constraints were

the following: there must be a specific number of residents assigned to rotation j each month k ; each resident must serve a certain number of months in rotation j ; all residents must be assigned one rotation each month; and certain residents must serve in pairs. To solve the problem, the authors proposed a decision support system built around the above linear programming model. However, the solution was found by a continuous LP , after which it was rounded to binary integers using a heuristic developed by the authors. The heuristic measured the “tightness” of each constraint set, and used it to calculate the so-called *rounding indicator ratio* which indicated the direction towards which the variables were to be rounded.

Gray et al. (1993) discussed the development of an expert system to schedule nurses according to their scheduling preferences. Assuming consistency in nurses’ preferences, an expert system was proposed and implemented to produce feasible schedules considering nurses’ preferences, but also accounting for overtime needs, desirable staffing levels, patient acuity, etc. The effort was driven by a previous study which revealed that creating a 12-week schedule for 16 nurses may take up to 40 hours of a human manual scheduling time (Kostreva and Geneviev, 1989). In addition, scheduling satisfaction was found to be one of eight most important measures of overall job satisfaction (Mueller and McCloskey, 1990).

A more specific problem was addressed by Yura (1994), where the objective was to satisfy worker’s preferences for time off as well as overtime, but under due date constraints. The author broke down the problem into two subproblems. One was a

relaxed version where the objective is to satisfy worker's preferences for days off, but it excluded any overtime, while the second attempted to minimize the total overtime while trying to satisfy worker's preferences for days off. Both problems were formulated as linear goal programming problems. The first one assumed that the overtime is undesirable for employers, while the second one extended the idea by including the overtime and was applied in cases of heavy work loads.

Badri et al. (1998) also utilized advantages of goal programming, but used it to formulate a multiobjective problem to account for faculty preferences in university course scheduling. The model provides a one-stage assignment using a zero-one goal programming model, which was an improvement over the previously proposed model by Badri (1996) that consisted of two stages (first one assigned faculty to courses at the departmental level, the second one distributed these combinations to available time slots). The model proposed by Badri et al. (1998), not only produced solutions in one stage, but also attempted to accommodate for faculty preferences to teach certain courses and during certain time intervals. The data structure was presented in the form of a matrix with rows indicating course priorities, and entries with priorities for specific time blocks. The constraints were classified into seven categories: a set of goals to ensure that all required courses were offered; available teaching loads for each instructor; limitations in classroom availability; faculty preferences for courses; limiting one of the preferences per combination; and ensuring that an instructor was assigned to only one course per time block. The model was successfully applied to course scheduling at the United Arab

Emirates University, and solved a problem of 252 decision variables, 66 goal constraints, and 167 system constraints.

Campbell (1999) further considered allocation of cross-trained resources in multidepartment service environment. Employers generally value more resource units with various skills and capabilities for performing greater number of jobs. It is in those cases when managers face challenges of allocating these workers such that the utility of the assignment of workers to a department is maximized. The author used fractional values, c_{id} to describe capabilities of each worker i to work in department d . In other words, a c_{id} was set to one if a worker i is fully qualified to work in the department d , or zero if the worker cannot work in the department d at all, and a fractional value between zero and one if the worker can be assigned to a department, but he or she is not fully qualified for the tasks involved. A binary value x_{id} then indicated whether a worker was assigned to a department or not. The author also defined a sum of capability values of workers assigned to work in department d as:

$$\omega_d = \sum_{i=1}^I c_{id} x_{id}$$

The utility of assigning workers to a department d was then simply a function of ω_d , i.e.,

$u_d(\omega_d)$. The overall objective was to maximize $U = \sum_{d=1}^D u_d(\omega_d)$, subject to constraints

that each worker must be assigned to a single department as well as that all x_{id} 's must be zeros or ones. The results of experiments showed that the benefits of cross-training

utilization may be significant. In most cases only a small degree of cross-training captured the most benefits, and tests also showed that beyond a certain amount, the additional cross-training adds little additional benefits.

III. RESEARCH BACKGROUND

The literature survey in the previous chapter showed an obvious need for a tool that would effectively schedule, track, and control resource allocation to projects, but from the perspectives of resource units themselves. Badiru (1993) proposed *Critical Resource Diagramming (CRD)* which is a simple extension to traditional *CPM* graphs. In other words, criticalities in project activities may also be reflected on resources. Different resource types or units may vary in skills, supply, or be very expensive. This discrimination in resource importance should be accounted for when carrying out their allocation in scheduling activities.

Unlike activity networks, *CRD*'s use nodes to represent each resource units. Also, unlike activities, a resource unit may appear more than once in a *CRD* network, specifying all different tasks for which a particular unit is assigned to. Similar to *CPM*, the same backward and forward computations may be performed to *CRD*'s. Figure 1 illustrates some of the *CRD* properties and features. Notice that resource unit 1 and resource unit 4 appear twice in the graph, meaning that they work on more than one project activity. Thus, the actual interpretation of any computations may be different than that of a conventional *CPM* network. Since units 1 and 4 worked on two different activities each, that could also imply that the units may have been cross-trained to perform a variety of tasks.

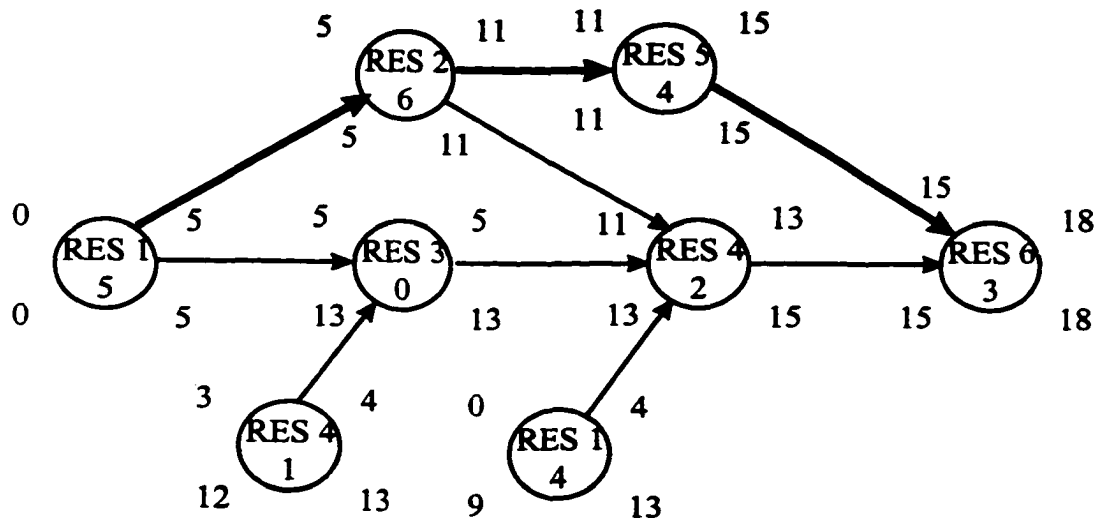


Figure 1. CRD Network Analysis (Badiru, 1993).

Critical resource path in Figure 1 is indicated by bolded arrows. Resource units on the critical path have no slack time left for performing their jobs, and their delay would delay the whole project. Badiru (1993) proposed several node classifications for analysis of CRD's: a node at which more than one arrow merges is defined as a *bottleneck* node; a node whose task depends on the task(s) of its immediate predecessors is defined as a *dependent* node; should such a node be on the critical path, it is referred to as the *critically dependent* node; a node from which more than one arrow points out is defined as a *burst* node. Obviously, delaying burst nodes increases chances of delaying the whole project. RES 3 serves as an example of a bottleneck resource node. RES 6 is an example of a critically dependent node.

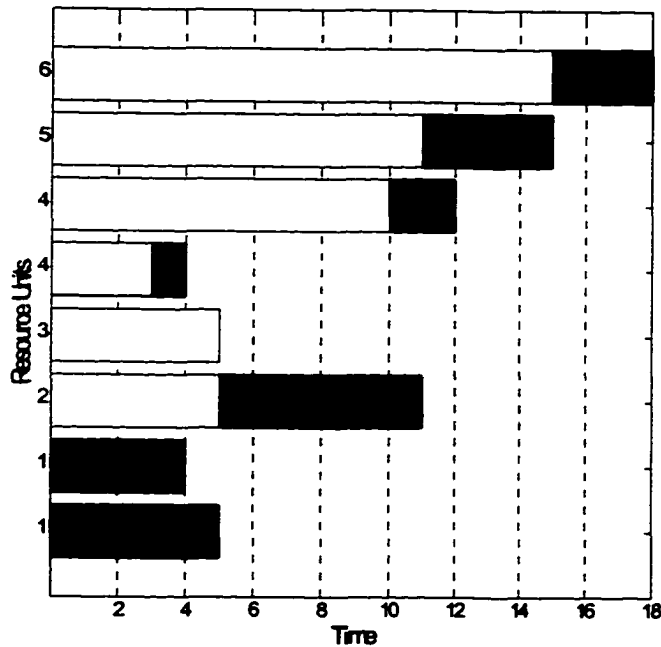


Figure 2. Resource Schedule Chart Based on Earliest Start Times (Modified from Badiru, 1993).

Badiru (1993) further defined a *resource scheduling chart* as shown in Figure 2. Each resource unit is represented by a horizontal bar, with a dark region indicating the interval of a resource unit's work. Badiru (1993) distinguished the above graph from a conventional Gantt chart, in a sense that resource units do not have slack times since they are assumed to be engaged throughout the project. In addition, it is pointed out that two tasks for resource unit 1 have jobs which overlap for a four time unit period. On the other hand, the two tasks for resource unit 4 are six time units "away" from each other. This could be an indication that resource unit four might end up being idle for a period of time.

In a sense, resource scheduling chart increases the resolution of resource loading graphs, such that it enables jobs and tasks of each particular resource unit to be monitored and recorded.

CRD's are a simple extension and a complementary tool to the traditional *CPM* graphs, that enhance the information on resource conflicts, and provide alternative insights into resource distribution to jobs, project tracking and control. However, the model as presented by Badiru (1993) and illustrated in Figures 1 and 2 is easily implementable only in cases when resource units are *pre-determined* to work on specific activities only. In any other case, when resource units are cross-trained or with varying qualifications, it is very hard to define the precedence relationships as illustrated in Figure 1. Due to the combinatorial nature of the problem, the model in Figure 1 is hard to reconstruct when scheduling resource units without prior knowledge of their exact assignments.

Consider an example as partially adopted from Badiru and Pulat (1995), where a project data is presented with only seven activities and two resource types. There are 10 total resource units of type one units and 15 units of type two available for the project. The activity precedence relations and resource requirements are given in Table 1.

Table 1. Example Project Data.

Activity	Predecessor	Resource Types	
		Type 1	Type 2
<i>A</i>	-	3	0
<i>B</i>	-	5	4
<i>C</i>	-	4	1
<i>D</i>	<i>A</i>	2	0
<i>E</i>	<i>C</i>	4	3
<i>F</i>	<i>A</i>	2	7
<i>G</i>	<i>B, D, E</i>	6	2

Assuming that resource units of both types are expected to perform differently if assigned to different activities, we cannot presume the duration of any activity before we actually decide which particular units of each type will be assigned to it. In the most complex case, a project manager or analyst would have a table of size $(10+15) \times 7$ with its entries representing preferences, costs, or time each resource unit would need to complete any of the seven activities. Having the project in a form as presented above, the construction of a *CRD* similar to the one in Figure 1 would be an enormous task.

Once each of the 25 resource units (10 units of type one and 15 of type two) are given specific assignments as to which activities each of them is going to carry out, a modification of the original *CRD* may be graphed as shown in Figure 3.

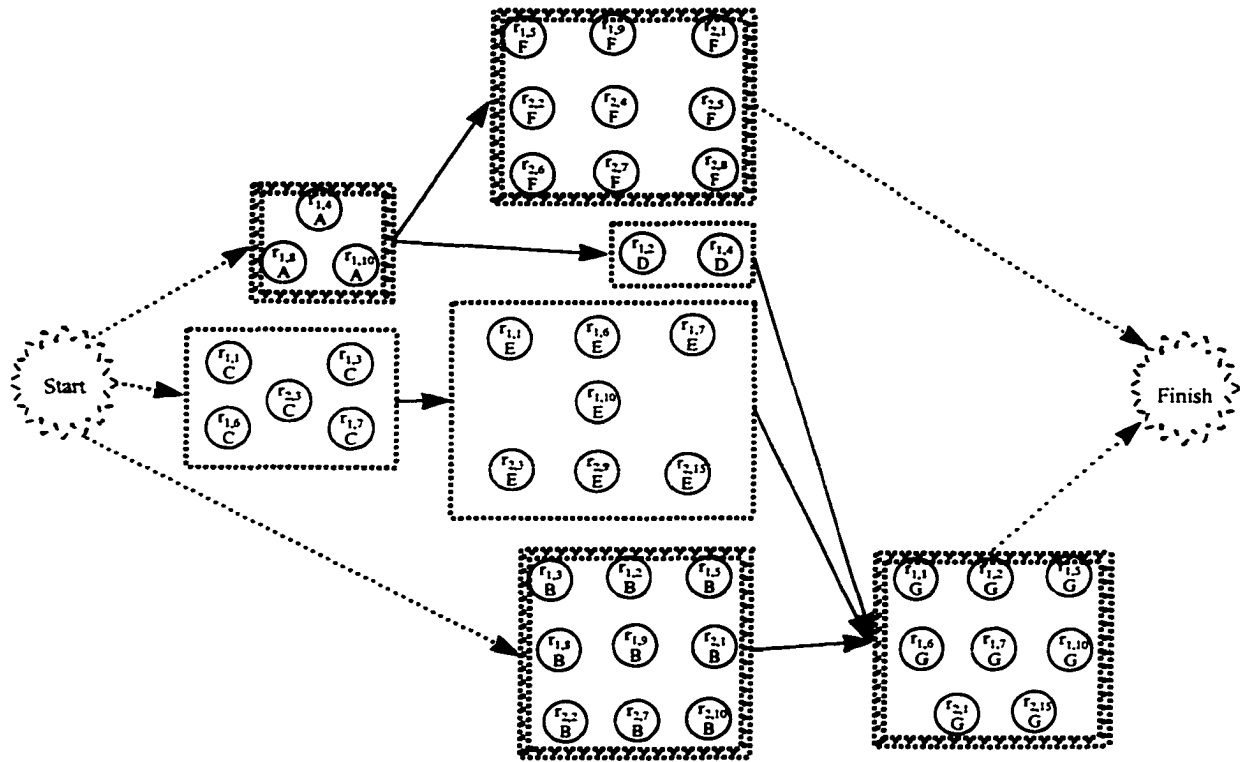


Figure 3. Modified Critical Resource Diagram.

The square nodes in Figure 3 represent activities, while the circles inside each activity block illustrate particular resource units that are assigned to each activity. Activities on the critical path are illustrated by the reinforced block boundaries. This implies that all resource units assigned to activity *G* (i.e., circular nodes with a subscript *G*) are critically dependent resource units (since they are all inside a block activity on a critical resource path at which more than one arrow merges). In addition, notice that each of the resource units assigned to activity *G* are also assigned to one of its immediate predecessors. In other words, units 1, 6, 7, and 10 of type one and unit 15 of type two are also assigned to activity *E*, while the units 2 and 5 of type one and unit 1 of type two are also selected to work on activity *B*. Both activities, *B* and *E* are the immediate predecessors of activity *G*.

In addition, activity *B* is also on a resource constrained critical path. Thus, to avoid any resource idleness and depressions in resource utilization graph, activity *E* should be well planned and completed at about the same time as activity *B*.

Consistently with the definitions by Badiru (1993), resource units working on activity *A* are all referred to as the *burst* units, since activity *A* precedes more than one other activity. Thus, these resource units bear somewhat greater responsibility for completing their tasks on time in order to avoid delays in total project duration.

3.1 Problem Statement

The objective of this research is the development of a generic project scheduling model capable of both effective and efficient mapping of multi-capable resource units to project activities. Besides resource-activity mapping, the model must also be able to incorporate a project manager's tacit or discretionary knowledge which is provided *ex ante* and may involve variables exogenous to the project itself. This is facilitated through a pre-specified utility function which may be held constant during project scheduling or allowed to vary across project parameters such as time, activity, resource type, and/or resource characteristics (capabilities, preferences, cost, availability). The model performing the above functions has been developed and implemented in a software prototype.

The following chapter presents a model which facilitates an easier construction of networks as shown in Figure 3. The methodology consists of an *activity scheduler* which prioritizes activities, and a *resource mapper* which assigns the most adequate resource units to each of the newly scheduled project activities. The actual implementation of the model presented in the methodology is discussed in the appendices. Figure B18 from Appendix B is an example of one of the outputs provided by the developed software prototype. The particular so-called *resource-activity grid* in Figure B18 conveys the same type of information as the *CRD* shown in Figure 3.

IV. METHODOLOGY

The methodology of this research represents an analytical extension of *CRD* discussed in the preceding chapter. As previously mentioned, the design considerations of the proposed model consist of two distinct procedures: activity scheduling and resource mapping. At each decision instance during a scheduling process, the *activity scheduler* prioritizes and schedules some or all candidate activities, and then the *resource mapper* iteratively assigns the most adequate resource units to each of the newly scheduled activities. Since the actual modeling of the *resource mapper* represents a true kernel of this research, it will be discussed first.

4.1 *Project Resource Mapper: Classification, Representation and Interdependencies among Project Resources and their Mapping to Project Activities*

Project resources are generally categorized into groups or types according to their similarities and functionality. Dreger (1992) discusses five main types of resources: capital, personnel, plant and equipment, materials and supplies, and space. Slowinski (1981) further considers classification of resources into renewable and non-renewable. Personnel, equipment and space are typically regarded as renewable since they can be re-engaged as soon as activities that are currently employing them are completed. Capital and, in many cases, materials and supplies are regarded as non-renewable since they are usually available in fixed amounts for the total project.

The methodology in this research is primarily focused on renewable resources. In addition, resources are not necessarily or solely categorized into types or groups according to their similarities (i.e., into personnel, equipment, space, etc.), but more according to hierarchy of their interdependencies. In other words, we assume that time-effective capabilities, preferences, or even cost of any particular resource unit assigned to work on an activity may be dependent on other resource units also assigned to work on the same activity. Some or all of these other resource units may, in the similar fashion, be also dependent on a third group of resources, and so on. Based on the above assumptions, we model competency of project resources in terms of following four resource characteristics: time-effective capabilities, preferences, cost, and availability. Time-effective capability of a resource unit with respect to a particular activity is the amount of time the unit needs to complete its own task if assigned to that particular activity. Preferences are relative numerical weights that indicate personnel's degree of desire to be assigned to an activity, or manager's perception on assigning certain units to particular activities. Similarly, each resource unit may have different costs associated with it relative to which activities it gets assigned to. Finally, not all resource units may be available to some or all activities at all times during project execution. Thus, times during which a particular unit is available to some or all activities are also incorporated into the mapping methodology. Each of the characteristics described may vary across different project activities. In addition, some or all of these characteristics (especially time-effective capabilities and preferences) may also vary within a particular activity

relative to resource interaction with other resources that are also assigned to work on the same activity.

In this research, resources whose performance is totally independent of their interaction with other units are grouped together and referred to as the type or group “one” and allocated first to scheduled activities. Resource units whose performance or competency is affected by their interaction with the type or group “one” units are grouped into type or group “two” and assigned (mapped) next. Resource units whose competency or performance is a function of type “two” or both types “one” and “two” are grouped into type “three” and allocated to scheduled activities after the units of the first two types have been assigned to them.

As previously indicated, these resource characteristics and interdependencies enable modeling of personnel’s voice and/or manager’s apriori knowledge and propensity of available resources. Prior to any assignment of resources to project activities, a manager may specify a utility or objective function that incorporates some or all of the above characteristics. Then, throughout the process of scheduling project activities, the model will attempt to map specific resource units to each newly scheduled activity such that the pre-specified utility or objective function is maximized. An example of a realistic utility function would be manager’s desire to maximize personnel’s preferences while still keeping the costs and project completion time as low as possible. Furthermore, this utility function may be more accented or discriminatory towards one or more resource types. For example, a manager may wish to maximize time-effective capabilities for all

resource groups in order to reduce project total time, but minimize cost of only contract workers which have been classified as resource type “three”. This type of utility function would contain a component which would be nonzero only when units of resource type three are mapped to newly scheduled activities.

Besides a possibility of being resource type-specific, a utility or objective function may also vary with time. For example, in the beginning of a project, a manager’s objective may consist primarily of cost and personnel preferences. In the later stages of the project, however, timely project completion may become the most important factor. To facilitate for this, a window function is used to filter out temporarily irrelevant additive components of the utility function and hold them at zero.

The modeling of the above resource characteristics is discussed in the following sections.

4.1.1 Modeling of Resource Characteristics and their Interdependencies

After candidate activities at each scheduling decision instance have been scheduled, we proceed to map available resource units to them such that a pre-specified utility or objective function is locally optimized. This utility function may consist of only one of the four resource characteristics (i.e., time-effective capabilities, preferences, costs, and availability), but is usually a blend of two or more of them.

As previously discussed, all resources are grouped into types (or categories), not necessarily or solely according to their similarities as traditionally done, but rather according to certain interdependencies that may exist among some or all resource characteristics. Those resource units whose characteristics are either constant or varied, but only across different activities are grouped into type “one”. Resource units belonging to higher indexed types may have their characteristics depend on units belonging to lower indexed types. In this study, resources whose characteristics are independent of their interaction with other units and vary only across activities are referred to as the *drivers*. Resources of higher indexed types whose characteristics do vary not only across different activities, but also within a single activity relative to their interaction with the *drivers*, are referred to as the *dependents*. Notice that a particular resource unit may at the same time be a *driver* to the units grouped in higher indexed types and be a *dependent* on those units grouped in the lower indexed types. It should be also noted that no interdependencies may exist among the resource units of the same type or group. Should that occur, the particular resource type should be split such that the dependent units are placed into a new subtype of higher index. All resource characteristics and interdependencies relevant to the pre-specified utility function must be evaluated before any units are assigned to any of the newly scheduled activities at each decision instance.

The most commonly used variables in this study are defined as follows:

$i \equiv$ project activity i , such that $i = 1, \dots, I$

$I \equiv$ number of activities in project network.

$t_c \equiv$ decision instance, i.e., time moment at which one or more activities qualify to be scheduled since their predecessor activities have been completed.

$PR(i) \equiv$ Set of predecessor activities of activity i .

$Q(t_c) \equiv$ Set of activities qualifying to be scheduled at t_c , i.e., $Q(t_c) = \{i \mid PR(i) = \emptyset\}$.

$j \equiv$ resource type j , $j = 1, \dots, J$.

$J \equiv$ number of resource types involved in the project.

$R_j \equiv$ number of units of resource type j available for the project.

$\langle j, k \rangle \equiv$ notation for k -th unit of type j .

$\rho_i^j \equiv$ number of resource units type j required by activity i .

$u_{t_c}^{j,k} \equiv$ a binary variable with a value of one if k -th unit of type j is engaged in one of the project activities that are in progress at the decision instance t_c , and zero otherwise. All $u_{t_c}^{j,k}$'s are initially set to zero.

$t_i^{j,k} \equiv$ time-effective executive capability of k^{th} unit of resource type j if assigned to work on activity i .

$p_i^{j,k} \equiv$ preference of k -th unit of resource type j to work on activity i .

$c_i^{j,k} \equiv$ estimated cost of k -th unit of resource type j if assigned to work on activity i .

$\alpha_i^{j,k}(t_c) \equiv$ desired start time or interval availability of k -th unit of type j to work on activity i at the decision instance t_c . In many cases this parameter is invariant across activities, and the subscript i may often be dropped.

The last four variables above represent resource characteristics which, when evaluated, play decisive role in determining which units should be mapped to which project activities. A project manager may consider one, more than one, or all of the four characteristics when performing activity-resource mapping. For example, a manager may wish to keep project costs as low as possible, while at the same time attempting to use resources with the best time-effective capabilities, consider their availability, and even incorporate their voice (in case of humans) or his/her own perception (in cases of human or non-human resources) in the form of preferences. This particular case would require the manager to come up with a general mapping utility function which will reflect the trade-offs between these resource characteristics as objectives for each resource unit. Mapping objective for each unit with respect to each activity is simply then a function of temporal capabilities, costs, preferences, and temporal availability, represented as follows:

$$\mathcal{U}_i^{j,k} = f(t_i^{j,k}, c_i^{j,k}, p_i^{j,k}, \alpha_i^{j,k}(t_c))$$

In simpler cases when the information is, for example, available only on time-effective capabilities and costs, while the preferences are either not available or neglected, and assuming no restrictions on resource temporal availabilities, the mapping objective for each resource unit with respect to an activity is then a function:

$$\mathcal{U}_i^{j,k} = f(t_i^{j,k}, c_i^{j,k})$$

In general terms, a manager's goal is always to maximize his or her utility function. It should then be noted that the particular utility function above will only be maximized when $f(t_i^{j,k}, c_i^{j,k})$ is of such form that both costs and resource task times are minimized. An example of a simple utility which is represented by minimizing resource costs only would be:

$$u_i^{j,k} = -c_i^{j,k}$$

At each scheduling time instance, t_c , available resource units are mapped to newly scheduled activities. This is accomplished by solving J number of zero-one linear integer problems (i.e., one for each resource type), where the coefficients of the decision vector correspond to evaluated utility or objective function for each unit of the currently mapped resource type:

$$\max \sum_{h \in \Omega(t_c)} \sum_{k=1}^{R_j} u_h^{j,k} \cdot y_h^{j,k} \quad \text{for } j = 1, \dots, J$$

where:

$y_h^{j,k} \equiv$ binary variable of the decision vector.

$\Omega(t_c) \equiv$ set of newly scheduled activities at decision instance t_c .

A $y_i^{j,k}$ resulting in a value of one would mean that k -th unit of resource type j is mapped to i -th ($i \in \Omega(t_c)$) newly scheduled activity at t_c . The above objective in each of J number of problems is subjected to four types of constraints, as illustrated below.

I) The first type of constraints ensure that each newly scheduled activity receives its required number of units of each project resource type:

$$\sum_{k=1}^{R_j} y_i^{j,k} = \rho_i^j \quad \text{for } i \in \Omega(t_c) \quad \text{for } j = 1, \dots, J$$

II) The second type of constraints prevent mapping of any resource units to more than one activity at the same time at t_c :

$$\sum_{i \in \Omega(t_c)} y_i^{j,k} \leq 1 \quad \text{for } k = 1, \dots, R_j \quad \text{for } j = 1, \dots, J$$

III) The third type of constraints prevent mapping of those resource units that are currently in use by activities in progress at time t_c :

$$\sum_{k=1}^{R_j} u_{t_c}^{j,k} \cdot y_i^{j,k} = 0 \quad \text{for } i \in \Omega(t_c) \quad \text{for } j = 1, \dots, J$$

IV) The fourth type of constraints ensures that the variables in the decision vector $y_i^{j,k}$ take on binary values:

$$y_i^{j,k} = 0 \text{ or } 1 \quad \text{for } k = 1, \dots, R_j, \quad i \in \Omega(t_C), \quad \text{for } j = 1, \dots, J$$

Therefore, in the first of the total of J runs at each decision instance t_C , available units of resource type “one” compete (based on their characteristics and pre-specified utility function) for their assignments to newly scheduled activities. In the second run, resources of type “two” compete for their assignments. Some of their characteristics, however, may vary depending on the “winners” from the first run. Thus, the information from the first run is used to refine the mapping of type or group “two” resources. Furthermore, the information from either or both of the first two runs is then used in tuning the coefficients of the objective function for the third run when resources of type “three” are mapped. Mapping of the J -th type of resources may be affected by the outcome of any of the previous $J-1$ runs. Since there may be up to I number of such instances (if at each decision instance, only one candidate activity is scheduled), the total of $I \times J$ mapping binary integer problems may have to be solved for a project. This is in addition to up to I problems necessary to concurrently schedule candidate activities by the *activity scheduler* (see Section 4.2).

It should be noted again that this model may only support interactive dependencies between units of different resource types. Thus, *dependent* units must be in higher

indexed types, since their dependencies may be evaluated and incorporated into a utility function only after their *drivers* (units in the lower indexed types or groups) have been mapped. This is necessary in order to eliminate any non-linearities in the model. Should a manager discover any interdependencies among resource units of the same type, the type must be split in a manner that sub-*dependents* are regrouped into a higher indexed subtype and all other higher indexed types shifted accordingly.

The solution to the above zero-one integer formulation is found using the Balas algorithm (Rao, 1983), which takes advantage of the special structure of zero-one problems to generate optimal solutions more efficiently. Although the procedure still relies on enumeration, it pursues a smart approach to explicitly enumerate only a few solutions explicitly, while the others are either automatically enumerated implicitly or the problem proves infeasible. Balas subroutine used in this research is from the *Tomlab* toolbox at <http://www.ima.mdh.se/tom/>

The algorithm starts by converting a general form of an *LP* zero-one problem to a more standardized form, by forcing the objective function to be minimizing (i.e., changing its sign, if it is a maximizing one), replacing all equality constraints by two inequality ones of opposite types, multiplying all inequalities of type “ \geq ” by negative one to convert them to the form of the “ \leq ” type, perturbing the decision variables from x_i to $(1-x_i)$ when the corresponding coefficients are negative in the objective vector, and finally introducing an m -component nonnegative slack vector Y . The problem then becomes:

$$\min f(X) = C^T X$$

s.t.

$$AX + Y = B$$

$$x_i = 0 \text{ or } 1$$

$$Y \geq 0$$

The algorithm starts with an initial partial solution with all free variables set to zero. A partial solution is defined as the one with some (but not all) of the n variables of the decision vector being assigned a value of one or zero. The variables not included in a partial solution are referred to as the free variables. If each of the free variables of a partial solution are assigned values, the partial solution becomes complete. An integer problem with two or three binary variables may easily be enumerated explicitly to find an optimal solution. Problem with more than three variables, however, would require an explicit enumeration of 2^n solutions. Balas method (Rao, 1983), starting with an initial partial solution, tries to assign binary values to one free variable at a time and generate a new series of partial solutions. When a completion of a partial solution gives a feasible solution of objective function smaller than the current best solution, or when a completion of a partial solution that will improve the infeasibility in the current solution cannot be found, then the current partial solution is fathomed. Once a partial solution is fathomed, all of its completions are also implicitly enumerated and can be discarded from future iterations. Thus, as soon as a new partial solution is generated, the algorithm

attempts to fathom it, and proceeds to generate a new partial solution using the so-called backtracking procedure, which simply refers to replacing one of the variables in the current partial solution (which is fathomed) with its complement to generate a new partial solution. The complete details of the algorithm are provided in full by Rao (1983).

The utility or objective function was previously introduced as common for all resource types and throughout the entire project duration. In some instances, however, a manager may wish to map resources according to a utility that varies with time. For example, she or he may place a greater emphasis on preferences in the early stages, and timely project completion in the later stages of a schedule. Similarly, some resource types are more expensive than others. This may require a manager to pay a particular attention to cost in mapping some resource type(s), and worry only about time-effective capabilities for all other resource types. A combination, where a utility may vary with respect to both time and different resource types is also possible. More detailed modeling and illustration of varying utility functions is discussed in the next section.

4.1.2 Dynamic and Resource Type-Specific Varying of Mapping Utility Function

Mapping units of all resource types according to the same utility function or objective may often be impractical and unrealistic. Cost issues may be of greater importance in mapping some, while inferior to time-effective capabilities of other resource types. If a utility function is fixed for all resource types, mapping may eventually produce undesired

assignments and results. Therefore, to accommodate the need for a resource-specific utility function as mapping objective, we may formulate the utility function as additive (Keeney and Raiffa, 1992). In such a case, each of its components pertains to a particular resource type and is multiplied by a *Kronecker's delta* function (Bracewell, 1978). Kronecker's delta then detects resource type whose units are currently being mapped and filters out all utility function components, except the one that pertains to the currently mapped resource type. Kronecker's delta is represented as:

$$\delta(j, s) = \begin{cases} 1 & \text{if } j = s \\ 0 & \text{if } j \neq s \end{cases}$$

One of the most general forms that a *resource-type driven utility function* may take is then as follows:

$$u_i^{j,k} = f_g(t_i^{j,k}, c_i^{j,k}, p_i^{j,k}, \alpha_i^{j,k}(t_c)) + \sum_{s \in \mathcal{S}} f_s(t_i^{j,k}, c_i^{j,k}, p_i^{j,k}, \alpha_i^{j,k}(t_c)) \cdot \delta(j, s)$$

where:

$f_g \equiv$ Component of the utility that is common to all resource types.

$f_s \equiv$ Component of the utility that pertains to a *specific* resource type.

$\mathcal{S} \equiv$ Set of resource types whose mapping requires a specific utility.

As an example, consider again a case where all resource types would be mapped according to their time-effective capabilities, except in the case of resource types “two” and “three” where costs would also be of consideration, and in the case of type “five”, resource preferences and availabilities would be considered:

$$\mathcal{U}_i^{j,k} = f(t_i^{j,k}) + f_2(c_i^{j,k}) \cdot \delta(j,2) + f_3(c_i^{j,k}) \cdot \delta(j,3) + f_5(p_i^{j,k}, \alpha_i^{j,k}(t_c)) \cdot \delta(j,5)$$

The above example illustrates a case where mapping of resource units is performed according to filtered portions of a manager’s utility function, according to grouping of resources into types. Similarly, a utility function may be dynamically adaptive and varying with project scheduling time. As previously indicated, some resource characteristics may be of greater importance to a manager in the early scheduling stages of a project rather than in the later stages. Such a utility function may be modeled as follows:

$$\mathcal{U}_i^{j,k} = f_g(t_i^{j,k}, c_i^{j,k}, p_i^{j,k}, a_i^{j,k}(t_c)) + \sum_{s \in \mathcal{T}} f_s(t_i^{j,k}, c_i^{j,k}, p_i^{j,k}, a_i^{j,k}(t_c)) \cdot w(t_{LO}^s, t_{HI}^s, t_c)$$

where:

$f_g \equiv$ Component of the utility that is common to all resource types.

$f_s \equiv$ Component of the utility that pertains to a *specific* project scheduling interval.

$t_{LO}^s, t_{HI}^s \equiv$ Specific time interval during which resource mapping must be performed according to a unique function.

$\mathcal{T} \equiv$ Set of above defined time intervals for a particular project.

$w(t_{LO}^s, t_{HI}^s, t_c) \equiv$ Window function with a value of one if t_c falls within the interval $[t_{LO}^s, t_{HI}^s)$, and zero otherwise:

$$w(t_{LO}^s, t_{HI}^s, t_c) = \begin{cases} 1 & \text{if } t_{LO}^s \leq t_c < t_{HI}^s \\ 0 & \text{otherwise} \end{cases}$$

As an example, consider a case where resource mapping in the early project stages is performed considering time-effective capabilities, costs, as well as their activity preferences. However, as the scheduling progresses, a manager's objective may shift largely towards timely completion of the project, rather than worrying as much about costs, and especially preferences. In that case, the only important characteristic left to be considered would be time-effective capabilities. The overall utility then may be modeled as follows:

$$u_i^{j,k} = f(t_i^{j,k}) + f(c_i^{j,k}, p_i^{j,k}) \cdot w(0,30, t_c)$$

or alternatively, depending on a manager's actual objective:

$$\mathcal{U}_i^{j,k} = f(c_i^{j,k}, p_i^{j,k}, t_i^{j,k}) \cdot w(0,30, t_c) + f(t_i^{j,k}) \cdot w(30,90, t_c)$$

where $[0,30)$ and $[30,90)$ are examples of the time ranges.

Finally, it is also possible to map different resource types according to different objectives *and* at different times simultaneously, by simply combining the two concepts above. For example, assume again that a manager forms his objective in the early stage of the project based on resources' temporal capabilities, costs, and preferences. In the later stage, the manager drops the costs and preferences and considers only resource capabilities, with the exception of resource type “three” whose costs should still remain in consideration for mapping. An example of a utility that would account for this scenario may be as follows:

$$\mathcal{U}_i^{j,k} = f(c_i^{j,k}, p_i^{j,k}, t_i^{j,k}) \cdot w(0,30, t_c) + \left(f(t_i^{j,k}) + f(c_i^{j,k}) \cdot \delta(j,3) \right) \cdot w(30,90, t_c)$$

As previously stated, the actual resource characteristics, that is, time-effective capabilities, costs, preferences, and resource availability may also be invariant for each resource unit regardless of its interaction with other units on a particular activity. On the other hand, some of the characteristics may largely vary relative to resource interaction

with units of lower indexed resource types. Modeling resource characteristics and their interactive dependencies for each are discussed in the following sections.

4.1.3 Time Effective Capabilities and Interdependencies

For resource units whose performance on a particular activity is independent of their interaction with other units, that is, for the *drivers*, $t_i^{j,k}$ is defined as the time it takes k^{th} unit of type j to complete its own task or process when working on activity i . Thus, different resource units, if multi-capable, can be expected to perform differently on different activities. Each *dependent* unit, on the other hand, instead of $t_i^{j,k}$, generally has a set of interdependency functions associated with it. Each function describes unit's interactive dependency on a particular *driver* for a particular activity. Thus, the maximum possible number of dependency functions of any *dependent* resource unit equals the number of activities times the total number of *driver* units for each activity.

Although time-effective interactive dependencies among resources may be expressed in various forms, in this research we pay a particular attention to two forms, which due to their simplicity, are expected to be the most commonly used ones: *additive* and *percentual* interactive resource dependencies. *Additive* interaction between a *dependent* and each of its *driver* resource unit indicates the amount of time that the *dependent* will need to complete its own task if assigned to work in conjunction with a particular driver.

This is in addition to the time the driver itself needs to spend working on the same activity:

$$(T_i^{j,k})_z \equiv (t_i^{j_D,k_D} + \tilde{t}_i^{j,k}) \cdot y_i^{j_D,k_D}$$

where:

$\langle j_D, k_D \rangle \in D^{j,k}$, where $D^{j,k}$ is a set of *driver* units (each defined by an indexed pair

$\langle j_D, k_D \rangle$ for a particular resource unit $\langle j, k \rangle$.

$(T_i^{j,k})_z \equiv z$ -th interactive time-effective dependency of k -th unit of type j on its *driver*

$\langle j_D, k_D \rangle$, $z = 1, \dots, \text{size}(D^{j,k})$. The actual number of these dependencies will depend on a manager's knowledge and familiarity with his/hers resources.

$\tilde{t}_i^{j,k} \equiv$ time needed in addition to $t_i^{j_D,k_D}$ for k -th *dependent* unit of type j to complete its task on activity i if it interacts with its *driver* unit j_D, k_D .

$y_i^{j_D,k_D} \equiv$ binary (zero-one) variable indicating mapping status of the *driver* unit $\langle j_D, k_D \rangle$. It equals one if the unit $\langle j_D, k_D \rangle$ is assigned to activity i , and zero if the unit $\langle j_D, k_D \rangle$ has been assigned to activity i . Therefore, each $(T_i^{j,k})_z$ will have a nonzero value only if $y_i^{j_D,k_D}$ is also nonzero (i.e., if the *driver* resource unit $\langle j_D, k_D \rangle$ has been previously assigned to activity i).

The percentual interactive dependency is similarly defined as:

$$(T_i^{j,k})_Z = t_i^{j_D,k_D} \cdot (1 + \tilde{t}_i^{j,k} \%) \cdot y_i^{j_D,k_D}$$

where $\tilde{t}_i^{j,k} \%$ is the percentage of time by which $t_i^{j_D,k_D}$ will be prolonged if the unit k of type j interacts with its *driver* $\langle j_D, k_D \rangle$.

It should be noted that other interactive dependencies, besides additive and percentual, are also possible and have been investigated in software implementation of the methodology. For instance, dynamic dependencies, where values of $T_i^{j,k}$ vary with time are possible with an example model as follows:

$$(T_i^{j,k}) = (t_i^{j_D,k_D} + \tau_1) \cdot y_i^{j_D,k_D} \cdot w(t_{LO}^1, t_{HI}^1, t_c) + (t_i^{j_D,k_D} + \tau_2) \cdot y_i^{j_D,k_D} \cdot w(t_{LO}^2, t_{HI}^2, t_c) + \dots$$

where:

$w(t_{LO}^s, t_{HI}^s, t_c) \equiv$ Window function with a value of one if t_c fall within the interval $[t_{LO}^s, t_{HI}^s]$, and zero otherwise, as discussed in the previous section.

This dynamic representation of resource capabilities is especially useful in modeling the effects of learning and forgetting in project scheduling and resource allocation.

Not all units of a dependent resource type need to have defined dependencies. Some units may simply have fixed $t_i^{j,k}$. If neither $t_i^{j,k}$ nor any dependency functions are provided for a particular resource unit $\langle j, k \rangle$, then the $t_i^{j,k}$ of the unit is set to infinity and the unit will not be assigned to activity i . As previously mentioned, the actual number of interactive dependencies for a given resource unit generally depends on a manager's experience with the particular unit, and his/her knowledge of its interactions on previous projects. When the number of interactive dependencies of a resource unit is nonzero, we need to evaluate all of the dependencies and take their maximum for $t_i^{j,k}$:

$$t_i^{j,k} = \max_{n=1, \dots, \text{size}(D^{j,k})} \{(T_i^{j,k})_n\}$$

The actual procedure that evaluates all T 's to obtain a single value for $t_i^{j,k}$, for each unit of a *dependent* resource type j , is implemented as follows:

For each newly scheduled candidate activity, i , at t_c , DO
 For each resource unit, k of the current dependent resource type, j , DO
 $max_function \leftarrow \max\{(T_i^{j,k})_n\}$ for $n = 1, \dots, size(DI^{j,k})$
 If $t_i^{j,k}$ is \emptyset
 $t_i^{j,k} \leftarrow max_function$
 Else If $t_i^{j,k}$ is nonempty
 $t_i^{j,k} \leftarrow \max(max_function, t_i^{j,k})$
 End If
 End If
 If $t_i^{j,k}$ is \emptyset $(T_i^{j,k})_f$ is undefined for all f
 $t_i^{j,k} \leftarrow \infty$
 End If
 End DO
End DO

The above procedure is repeated for each newly scheduled project activity as many times as there are resource types. $t_i^{j,k}$ is, as previously mentioned, evaluated first for lower indexed resource types, since it is those types that may serve as *drivers* to higher indexed resource types or groups.

4.1.4 Resource Costs and Resource Interdependencies Based on Costs

Modeling cost characteristics follows a similar logic used for representation of temporal capabilities and interdependencies. In place of $t_i^{j,k}$, we now define a variable $c_i^{j,k}$, which represents the cost (say, in Dollars) of k -th unit of resource type j if it gets assigned to work on activity i . This value of $c_i^{j,k}$ may be invariant regardless of a unit's

interaction with other resources, or it may vary relative to interaction among resources, and thus, implying cost interdependencies which need to be evaluated before any mapping is performed (provided that the cost considerations are a part of a manager's utility or objective for mapping).

In cases when a cost of a resource unit for an activity varies depending on its interaction with units of other (lower indexed) types, we define cost dependencies as:

$$(C_i^{j,k})_z = \tilde{c}_i^{j,k} \cdot y_i^{j_D,k_D}$$

where:

$y_i^{j_D,k_D} \equiv$ a binary variable indicating the status of the particular *driver* resource unit $\langle j_D, k_D \rangle$, as defined in the previous section.

$\tilde{c}_i^{j,k} \equiv$ interactive cost of k -th unit of type j on its *driver* $\langle j_D, k_D \rangle$, with respect to activity i .

$(C_i^{j,k})_z \equiv$ z -th evaluated interactive cost dependency of k -th unit of type j on its *driver* $\langle j_D, k_D \rangle$, $z = 1, \dots, \text{size}(D^{j,k})$. The values of each $(C_i^{j,k})_z$ equals $\tilde{c}_i^{j,k}$ when $y_i^{j_D,k_D}$ equals one, and zero otherwise. The actual number of these interactive cost dependencies will again depend on a manager's knowledge and information about available resources.

Given a set of cost dependencies, we compute the overall $c_i^{j,k}$ as a sum of all evaluated $(C_i^{j,k})_z$'s as follows:

$$c_i^{j,k} = \sum_{z=1}^{|D^{j,k}|} (C_i^{j,k})_z$$

Once evaluated, each $c_i^{j,k}$ may be a part of a composite utility function as illustrated in the previous section, or a single objective coefficient, in cases when resources are mapped by minimizing costs only.

4.1.5 Resource Preferences and Resource Interdependencies Based on their Preferences

In pure economic analyses, preferences are often driven by monetary factors. In such cases, preferences may simply be modeled as negative costs. In many other instances, however, due to political, environmental, safety, or community standards, aesthetics, or other similar non-monetary reasons, pure monetary factors may not necessarily prevail in decision making. It is those other non-monetary factors that we wish to capture by introducing preferences in resource mapping to newly scheduled activities. The actual representation of preferences is almost identical to those of the costs. In other words, resources may have constant preferences on activities regardless of their interaction, or

their preferences may vary with respect to any particular activity relative to which units of other types have already been mapped to that activity. This latter scenario especially pertains to human resources, and is represented by the following form:

$$(P_i^{j,k})_z = \tilde{p}_i^{j,k} \cdot y_i^{j_D,k_D}$$

where $\tilde{p}_i^{j,k}$ is an interactive preference of k -th unit of type j on its *driver* $\langle j_D, k_D \rangle$, with respect to activity i . $(P_i^{j,k})_z$ is z -th evaluated interactive preference dependency of k -th unit of type j , with respect to activity i . Finally, again identically to modeling costs, $p_i^{j,k}$ is computed as:

$$p_i^{j,k} = \sum_{z=1}^{|D^{j,k}|} (P_i^{j,k})_z$$

Final resource characteristic, the availability, is discussed and modeled in the following section.

4.1.6 Resource Availability in Resource-Activity Mapping

Having certain number of resource units of each type available for a project does not necessarily imply that all of the units are available all the time for the project or any of its activities in particular. Due to transportation, contracts, learning, weather conditions,

logistics, or other factors, some units may only have *time preferences* for when they are available to start working on a project activity or the project as a whole. Others may have *strict time intervals* during which they are allowed to start working on a particular activity or the project as a whole. This latter, strictly constrained availability may be easily accommodated by the previously considered *window* function, $w(t_{LO}, t_{HI}, t_c)$.

Having too strictly defined intervals as above, during which resource units are available to take on their tasks or engage into project may be too rigid of a constraint. In many cases, especially for humans, resources may have a desired or “ideal” time when to start their work or be available in general. If that desired time is not achievable, then certain deviations are permissible and resources are flexible to become available at a time that may be “somewhat” earlier or later than initially desired. This flexible availability may simply be represented by fuzzifying the specified desired times using the following function:

$$\alpha_i^{j,k}(t_c) = \frac{1}{1 + a(t_c - \tau_i^{j,k})^b}$$

where:

$\tau_i^{j,k} \equiv$ desired time for k -th unit of resource type j to start its task on activity i . This desirability may either represent the voice of project personnel (as in the case of

preferences), or manager's perception on resource's readiness and availability to take on a given task.

$\alpha_i^{j,k}(t_c) \equiv$ fuzzy membership function indicating a degree of desirability of $\langle j, k \rangle$ -th unit to start working on activity i , at the decision instance t_c .

$a \equiv$ parameter that adjusts for the width of the membership function.

$b \equiv$ parameter that defines the extent of *start time* flexibility.

It should be noted that when no desirable times are specified, the value of $\tau_i^{j,k}$ is by default set to t_c , thus holding the membership function at unity.

The crisp and fuzzified desired start times are depicted in the upper and lower subplots in Figure 4, respectively. The effect of variations in the two membership parameters, a and b , is also shown in the lower subplot of Figure 4. Notice that variations in the parameter b , define the sharpness of the membership function's peak. Varying the parameter a will cause variation in the overall spread of the function.

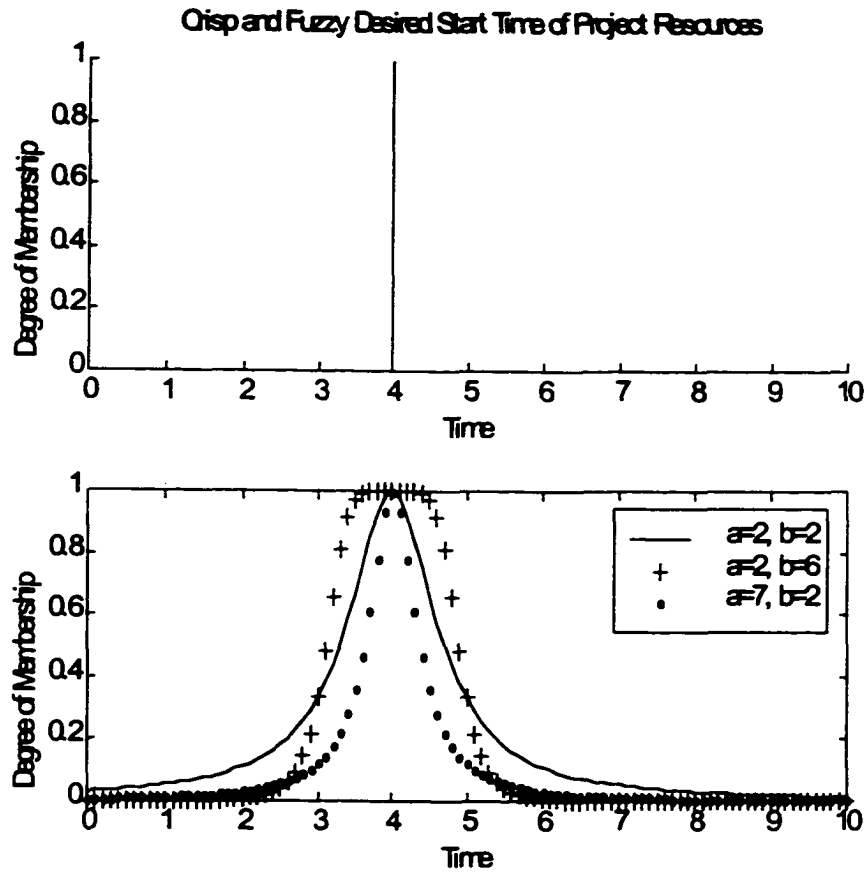


Figure 4. Incorporating Resource Availability into Mapping Constraints.

The membership function, $\alpha_i^{j,k}(t_c)$, is in effect a unimodal function with a peak and sides that approach, but never quite reach zero. This implies that a resource unit may be employed virtually at any time, but with the highest “desirability” at the moment where the function is at its peak. In cases when $\alpha_i^{j,k}(t_c)$ is modeled not as a fuzzy membership function, but as a previously discussed *window* function, the region outside the function indicates absolute unavailability of a resource unit to start a task or engage the project. It is obvious that $\alpha_i^{j,k}(t_c)$, once evaluated, serves as one of the resource characteristics that may be used as a part of an overall manager’s utility function for mapping of resources to

activities. This utility function is then used as a coefficient vector in the zero-one integer programming model that performs resource-activity mapping. Depending on the mathematical form of the utility, it may happen that a zero value of the evaluated $\alpha_i^{j,k}(t_c)$, if t_c falls outside the $[t_{LO}, t_{HI}]$ range, may cause zeros in some coefficients of the objective function. Due to the nature of linear programming, zeros in the coefficients of the objective do not imply that corresponding variables in the solution will also take the value of zero. In our case, that would mean that although we flagged off a resource unit as unavailable, the solution may still map it to an activity. Thus, we need to strictly enforce strict the interval (un)availability by adding information into constraints. For that we perturbed the third mapping constraint which was previously set to prohibit mapping of resource units at time t_c which are in use by activities in progress at that time. The constraint was originally defined as:

$$\sum_{k=1}^{R_j} u_{t_c}^{j,k} \cdot y_i^{j,k} = 0 \quad \text{for } i \in \Omega(t_c) \quad \text{for } j = 1, \dots, J$$

To now further prevent mapping of resource units whose $\alpha_i^{j,k}(t_c)$ equals zero at t_c , we modify the above constraint as follows:

$$\sum_{k=1}^{R_j} (u_{t_c}^{j,k} + (1 - \alpha_i^{j,k}(t_c))) \cdot y_i^{j,k} = 0 \quad \text{for } i \in \Omega(t_c) \quad \text{for } j = 1, \dots, J$$

This modified constraint now, not only filters out those resource units that are engaged in activities in progress at t_c , but also those units which were flagged as unavailable at t_c due to any other reasons.

So far, at each t_c , we map available resources categorized into types to newly scheduled activities, such that units of lower indexed types are mapped first. Then based on that outcome, units of higher indexed types are sequentially mapped by type by paying attention to their dependencies on units of lower indexed types. The next section discusses the actual activities, and how they are being prioritized and scheduled, before we start mapping resource units to them.

4.2 ACTIVITY SCHEDULER: PRIORITIZING AND SCHEDULING PROJECT ACTIVITIES

This chapter explains how activity duration is initially estimated before assigning resources to it and refining its duration. It also discusses prioritizing and scheduling project activities. Project activities are scheduled according to two criteria. The first one is based on basic activity attributes: initially estimated duration, resource requirements, and the dynamically updated amount of depleted slack at the decision instance t_c . The second criteria is a project manager's pre-specified level of attempt to balance (centralize) loading graphs of one or more resource types.

The following section discusses how durations of all project activities are initially estimated.

4.2.1 Initial Estimation of Project Activities Duration

Traditionally, a project manager estimates duration of each project activity first, and then assigns resources to it. In this study, although we don't exclude a possibility that an activity duration is independent of resources assigned to it, we assume that it is those resource units assigned to a particular activity that determine how long it will take for the activity to be completed. We further assume that resources even of the same functionality may vary among themselves in terms of qualifications, knowledge, skill level, and time-effective capabilities. Therefore, an activity duration may greatly be affected by our particular selection of different resource units, although they may all be capable of accomplishing the same type of work. Normally, more capable and qualified resource units are likely to complete their tasks faster, and vice versa. Thus, activity duration in this research is considered a *resource driven activity attribute*.

In this model, we first schedule activities, and then map resource units to them. However, since activity duration is assumed to be resource driven, we then cannot really schedule activities before knowing their duration. To resolve this issue, we initially only estimate the most optimistic activity duration using the available information on time-effective capabilities of *driver* resource units, that is, those whose performance is independent of their interaction with other units. This information is used for developing

a preliminary unconstrained *CPM* schedule which is later dynamically refined as resource units start to be mapped to activities and duration of each activity becomes more precise.

The initial duration d_i of a project activity is simply estimated by sorting the known $t_i^{j,k}$'s of all *driver* resource units and then computing d_i as following:

$$d_i = \max \{t_i^{j_D, k=\rho_i^j}, \text{ for } \forall j_D\}$$

The computations of d_i 's for a project of seven activities and two resource types is illustrated in Tables 2 and 3.

Table 2. Example Representation of Time-Effective Resource Capabilities and Interdependencies to Seven Project Activities.

Activities	$i=A$	$i=B$	$i=C$	$i=D$	$i=E$	$i=F$	$i=G$
ρ_i^j ($j=1$)	1	2	4	1	3	2	4
Unit 1			1.5		4.00	5.0	1.5
Unit 2	2.3	6.00	3.4	2.6	4.50	1.6	1.3
Unit 3	1.7	4.6	3.3		5.00	1.0	4.7
Unit 4	2.1		4.8		7.50	5.0	2.8
ρ_i^j ($j=2$)	0	4	1	0	3	3	2
Unit 1	-	2.0	$T_C^{2,1}$	-	$T_E^{2,1}$	$T_F^{2,1}$	3.0
Unit 2	-	2.5	$T_C^{2,2}$	-		4.8	$T_G^{2,2}$
Unit 3	-	5.1	$T_C^{2,3}$	-	$T_E^{2,3}$	4.0	
Unit 4	-		$T_C^{2,4}$	-		$T_F^{2,4}$	
Unit 5	-	4.8	$T_C^{2,5}$	-	5.0	6.0	2.7
Unit 6	-	5.2	$T_C^{2,6}$	-	$T_E^{2,6}$	$T_F^{2,6}$	$T_G^{2,6}$

Given the data in Table 2, we can easily compile it to estimate the initial duration, d_i of each activity, and tabulate the results as shown on the bottom of Table 3.

Table 3. Initially Estimated Activity Durations.

Activities	A		B		C		D		E		F		G		
ρ_i^j (j=1)	unit t	1	unit	2	unit t	4	unit	1	unit t	3	unit	2	unit	4	
t_i^{1,ρ_i^1}	3	1.7	3	4.6	1	1.5	2	2.6	1	4.0	3	1	2	1.3	
	4	2.1	2	6.0	3	3.3	1		2	4.5	2	1.6	1	1.5	
	2	2.3	1		2	3.4	3		3	5.0	1	5	6	2.8	
	1		4		4	4.8	4		4	7.5	4	5	10	4.7	
	1.7		6.00		4.8		2.6		5.0		1.6		4.7		
ρ_i^j (j=2)	unit t	0	unit	4	unit t	1	unit	0	unit t	3	unit	6	unit	2	
t_i^{2,ρ_i^2}		-	1	2.0		-		-		-		-		-	
		-	2	2.5		-		-		-		-		-	
		-	5	4.8		-		-		-		-		-	
		-	3	5.1		-		-		-		-		-	
		-	6	5.2		-		-		-		-		-	
		-	2	-		-		-		-		-		-	
t_i^{2,ρ_i^2}		-	5.1			-		-		-		-		-	
$\max(t_i^{1,\rho_i^1}, t_i^{2,\rho_i^2})$		$d_A=1.7$		$d_B=6.0$		$d_C=4.8$		$d_D=2.6$		$d_E=5.0$		$d_F=5.0$		$d_G=4.7$	

Once d_i is estimated for each project activity, we use it as information for prioritizing activities later in resource constrained scheduling. Modeling and strategy used in this research for activity prioritization is discussed in the following section.

4.2.2 Computing and Dynamic Updating of Activity Priorities

At each decision instance t_c (in resource constrained non-preemptive scheduling as investigated in this study), activities whose predecessors have been completed enter the set of qualifying activities, $Q(t_c)$. In cases of resource conflicts we often have to prioritize activities in order to decide which ones to schedule. In this methodology we prioritize activities based on two (possibly conflicting) objectives:

1. *Basic Activity Attributes*, such as the *current amount of depleted slack*, number of successors, and initially estimated optimistic activity duration, d_i .
2. Degree of manager's desire to *centralize* (or *balance*) *the loading* of one or more pre-selected project resource types.

Amount of Depleted Slack, $S_i(t_c)$, is defined in this research as a measure of how much total slack of an activity from unconstrained *CPM* computations has been depleted each time the activity is delayed in resource constrained scheduling due to lack of available resource units. The larger the $S_i(t_c)$ of an activity, the more its has been delayed from its unconstrained schedule, and the greater probability that it will delay the entire project.

Before resource constrained scheduling of activities (as well as resource mapping which is performed concurrently) starts, we perform a single run of *CPM* computations to determine initial unconstrained *Latest Finish Time*, LFT_i of each activity. Then, as the resource constrained activity scheduling starts, at each decision instance t_c , we calculate $S_i(t_c)$ for each candidate activity (from the set $\mathbf{Q}(t_c)$) as follows:

$$S_i(t_c) = \frac{t_c + d_i}{LFT_i} = \frac{t_c + d_i}{LST_i + d_i} \quad i \in \mathbf{Q}(t_c)$$

$S_i(t_c)$, as a function of time, is always a positive real number. The value of its magnitude is interpreted as follows:

- when $S_i(t_c) < 1$, the activity i still has some slack remaining and it may be safely delayed;
- when $S_i(t_c) = 1$, the activity i has depleted all of its resource unconstrained slack and any further delay to it will delay its completion as initially computed by conventional unconstrained *CPM*;
- when $S_i(t_c) > 1$, the activity i has exceeded its slack and its completion will be delayed beyond its unconstrained *CPM* duration.

Graphical illustration of *amount of depleted slack* is shown in Figure 4.

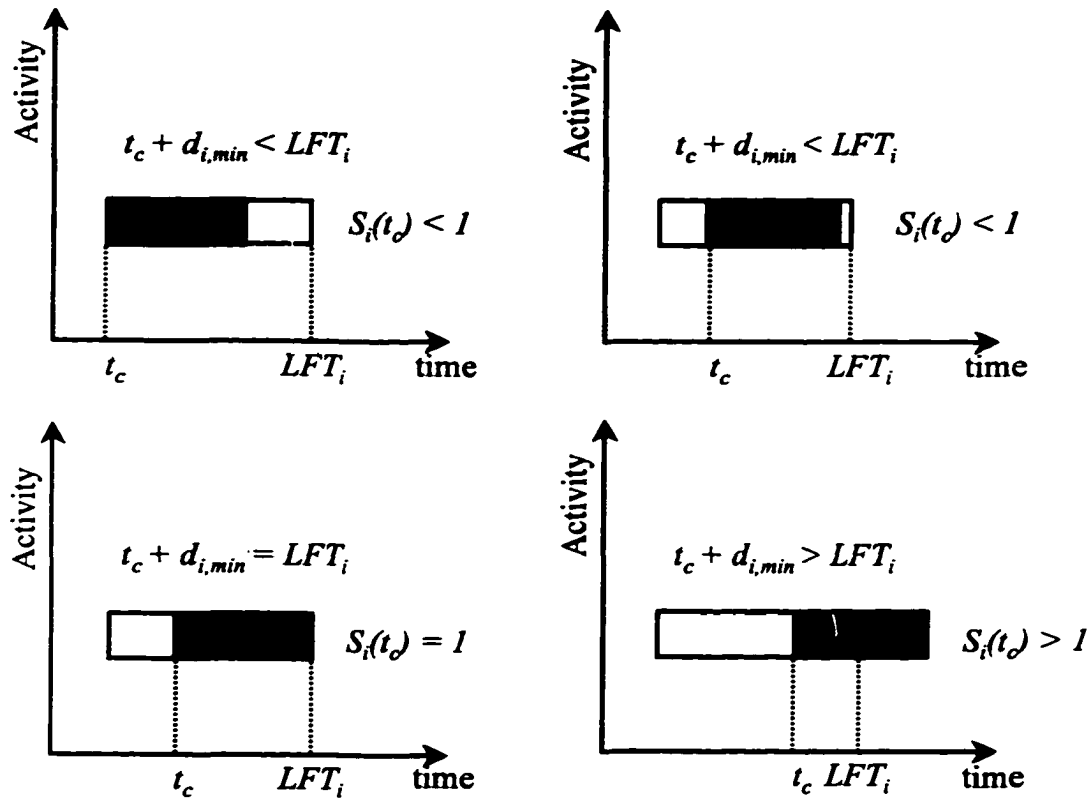


Figure 4. Graphical illustration of the Amount of Depleted Slack Measure.

Once calculated at each t_c , the current amount of depleted, $S_i(t_c)$, is then used in combination with the other two activity attributes for assessing activity priority for scheduling. (These additional attributes are *the number of activity successors*, as well as its *initially estimated duration* d_i). The number of successors is an important determinant in prioritizing, because if an activity with many successors is delayed, chances are that any of its successors will also be delayed, thus eventually prolonging the entire project

itself. Therefore, the prioritizing weight, w_p' , pertaining to basic activity attributes is computed as follows:

$$w_i^p = S_i(t_c) \cdot \left(\frac{\varsigma_i}{\max(\varsigma_i)} \right) \cdot \left(\frac{d_i}{\max(d_i)} \right)$$

where:

$w_i^p \equiv$ activity prioritizing weight that pertains to basic activity attributes.

$\varsigma_i \equiv$ number of successors activities of current candidate activity i .

$\max(\varsigma_i) \equiv$ maximum number of activity successors in project network.

$\max(d_i) \equiv$ maximum of the most optimistic activity durations in a project network.

Notice that, as a project scheduling time progresses, w_i^p becomes largely dominated by the value of $S_i(t_c)$. In the early stages of a project, most activities are expected to have plenty of slack left from their resource unconstrained schedule, forcing $S_i(t_c)$ to remain less than unity (notice again that as long as $S_i(t_c) < 1$, an activity i may be safely postponed). However, as the scheduling time elapses, more activities deplete their unconstrained slack, which increases the value of $S_i(t_c)$ for some of them far beyond unity. Since the issue of timely project completion traditionally becomes increasingly more important with time, $S_i(t_c)$ was left unscaled in the equation for w_i^p .

The secondary objective that may influence activity prioritizing is a manager's desire for a somewhat centralized (i.e., balanced) resource loading graph for one or more resource groups or types. This is generally desirable in cases when a manager does not wish to commit all of the available project funds at the very beginning of the project (Dreger, 1992), or to avoid frequent hiring and firing of project resources (Badiru and Pulat, 1995), which may greatly affect overall project budget. Resource loading graphs are generally illustrated by staircase type of plots with time units on their x -axis, and number of currently engaged project units on y -axis. In this research, we attempt to balance (centralize) loading of pre-specified resources by scheduling those activities whose resource requirements will minimize the increase in the staircase size in the early project stages, and then minimize the decrease in the step size in the later stages. A completely balanced resource loading graph contains no depression regions as defined by Konstantinidis (1998), i.e., it is a nondecreasing graph up to a certain point at which it becomes non-increasing. This should provide for a smooth loading graph, however with a possibility of extended project duration. Generally, different resources are of different importance to a manager, and he or she may not wish to attempt to balance the loading of all resource types. Figure 5 shows a Gantt chart and resource loading graphs of sample project with 7 activities and two resource types. Neither of the two resource type loadings are obviously balanced. The same project has been re-run using the above reasoning, and shown in Figure 6. Notice that the loading of resource type two is now fully balanced. The loading of resource type one still contains depression regions, but to a considerably lesser extent than in the previous figure.

The activity prioritizing weight that pertains to attempting to centralize resource loading is computed in this research as follows:

$$w_i^r = \sum_{j=1}^J \frac{\rho_i^j}{R_j}$$

where:

$w_i^r \equiv$ prioritizing weight that incorporates activity resource requirements.

$\rho_i^j \equiv$ number of resource type j units required by activity i .

$R_j \equiv$ total number of resource type j units required for the project.

Notice that w_i^p and w_i^r are weights of possibly conflicting objectives in prioritization of candidate activities for scheduling.

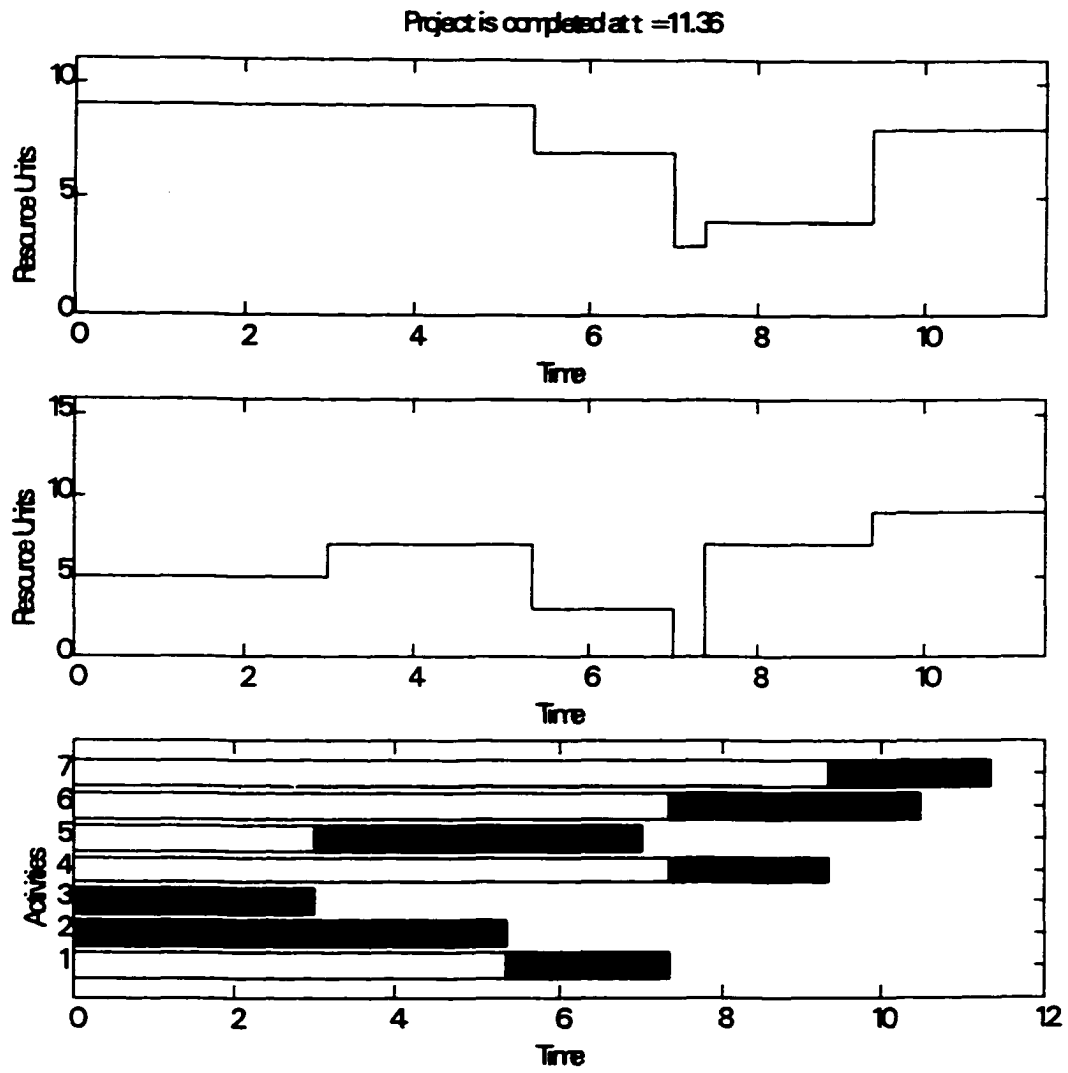


Figure 5. Example of Unbalanced Resource Loading Graphs.

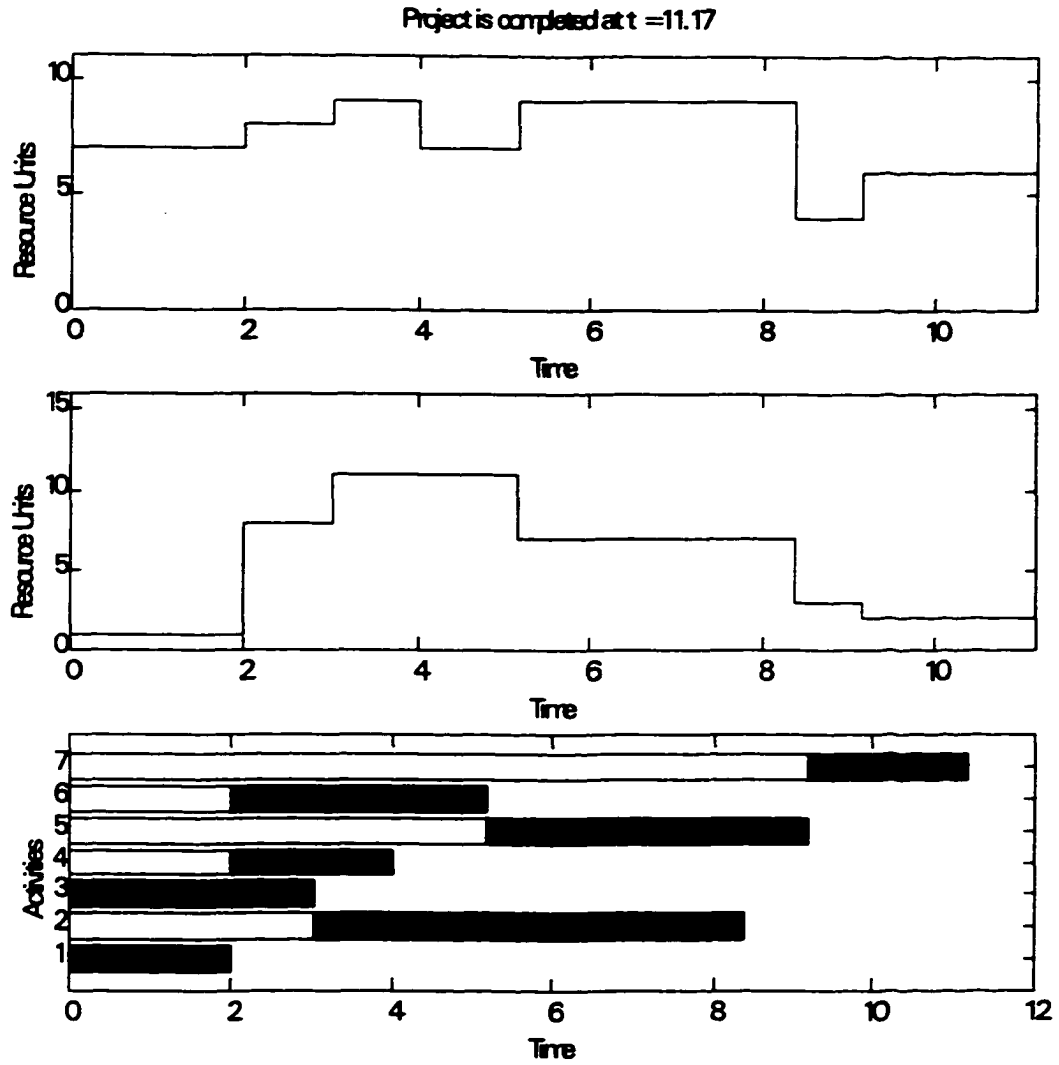


Figure 6. Example of a Project Schedule with the Loading Graph of Resource Type Two Fully Balanced.

To further limit the range of w_i^r between zero and one, we scale it as follows:

$$w_i^r = \frac{w_i^r}{\max(w_i^r)}$$

Notice that with w_i^r being scaled as above, its contribution to activity prioritization may be significant in comparison to w_i^p only in early project stages. As discussed previously, the reasoning for such a scenario is that timely completion of a project (which is dictated by $S_i(t_c)$) traditionally becomes increasingly more important as the scheduling of a project progresses. Thus, in cases when w_i^p and w_i^r are compiled into a single additive objective function for activity prioritization, w_i^r may prevail over w_i^p only at the beginning of a scheduling process. Once computed, w_i^p and w_i^r are combined to form the coefficients in the objective function based on which some or all (depending on resource availability) of candidate activities at decision instance t_c will be scheduled. Modeling of this objective function, and constraints is discussed in the following section.

4.2.3 Formulating the Objective Function for Activity Scheduling and Resource Balancing

With the two weights w_i^p and w_i^r defined and computed, we further use them as the coefficients of activity scheduling objective function:

$$\max \left(\sum_{i \in Q(t_c)} w_i^p \cdot x_i \right) + W \left(\sum_{i \in Q(t_c)} (1 - w_i^r \cdot x_i) \right)$$

where:

$x_i \equiv$ binary variable whose value becomes one if a candidate activity $i \in Q(t_c)$ is scheduled at t_c , and zero if the activity i is not scheduled at t_c .

$W \equiv$ Decision Maker's supplied weight that conveys the importance of resource centralization (balancing) in project schedule.

Notice that W is a parameter that allows a manager to further control the influence of w_i^p . Large values of W will place greater emphasis on the importance of resource balancing. However, to again localize the effect of W to the early stages of a project, we dynamically decrease its value at each subsequent decision instance, t_c according to the following formula:

$$W_{new} = W_{old} \left(\frac{\sum_{i=1}^I d_i - \sum_{i \in H(t_c)} d_i}{\sum_{i=1}^I d_i} \right)$$

where:

$\sum_{i=1}^I d_i \equiv$ The sum of all the most optimistic activity durations (as determined by conventional resource unconstrained CPM computations) for all activities in project network.

$H(t_c) \equiv$ set of activities that have been so far scheduled by the time t_c .

In the previous section, it was proposed that one way of balancing resource loading was to keep minimizing the increase in the staircase size of the loading graph in the early project stages, and then minimize the decrease in the step size in the later stages. The problem with such a reasoning is that a continuous increase in the loading graph in early stages may eventually lead to infeasibility due to limiting constraints in resource availability. Therefore, an intelligent mechanism is needed that will detect the point when resource constraints become binding and force the scheduling to proceed in a way that will start the decrease in resource loading, as previously depicted in Figure 6. In other words, we need to formulate a linear programming model whose constraints will drive the increase in resource staircase shaped loading function up to a point when resource availability is reached. As soon as such a point is reached, the model must adjust the objective function and modify (relax) the constraints to start minimizing the staircase decrease of resource loading.

The constraints to implement this procedure are modified from the traditional knapsack problem. In conjunction with the above objective function, the constraints are formulated to ensure that at each decision instance t_C , maximal number of candidate activities are scheduled, while satisfying activity precedence relations, preventing the excess of resource limitations, and most importantly, flag off the moment when resource limitations are about to be violated. To facilitate a computer implementation and prevent the strategy from crashing, we introduce an auxiliary zero-one variable, x , in this study

referred to as the *peak flag*. The value of λ in the decision vector is zero as long as current constraints are capable of producing a feasible solution. Once that is impossible, all variables in the decision vector must be forced to zero, except λ , which will then take a value of one and indicate that the peak of resource loading is reached. At that moment, the constraints that force the increase in resource loading are relaxed (eliminated).

The *peak flag* is appended to the previous objective function as follows:

$$\max \left(\sum_{i \in Q(t_c)} w_i^p \cdot x_i \right) + W \left(\sum_{i \in Q(t_c)} (1 - w_i^r \cdot x_i) \right) - b \lambda$$

where:

$b \equiv$ arbitrary large positive number (in computer implementation of this study, b was

taken as $b = \sum_{i=1}^I d_i$).

Thus, λ is in effect, a dummy variable whose sole purpose is to prevent a computer implementation of the above methodology from crashing. There are two types of constraints associated with the above objective of scheduling project activities. The first type simply serves to prevent scheduling of activities which would overuse available resource units:

$$\sum_{i \in Q(t_c)} \rho_i^j \cdot x_i + \left(R_j - \sum_{i \in G(t_c)} \rho_i^j \right) x \leq \left(R_j - \sum_{i \in G(t_c)} \rho_i^j \right), \quad j = 1, \dots, J$$

where:

$x_i \equiv$ candidate activity qualified to be scheduled at t_c

$G(t_c) \equiv$ set of activities that are in progress at time t_c .

$\left(R_j - \sum_{i \in G(t_c)} \rho_i^j \right) \equiv$ difference between the total available units of resource type j (denoted as R_j) and the number of units of the same resource type being currently consumed by the activities in progress during the scheduling instant t_c .

Notice that $\left(R_j - \sum_{i \in G(t_c)} \rho_i^j \right)$ appears on both sides of the constraint. On the right hand side (*RHS*) of the inequality, it serves to simply prevent the infeasibility, that is, overuse of available resources and force x_i 's to zero in such a case. Its purpose on the left hand side (*LHS*) is to hold x to zero for as long as the original problem is feasible. Notice that the number of the above constraints for each problem is equal to the number of project resource types, J .

The second type of constraints serves to force the gradual increase in the stairstep resource loading graphs. In other words, at each scheduling instant t_c , these constraints will attempt to force the model to schedule those candidate activities whose total resource

requirements are greater or than equal the total requirements of the activities that have just finished at t_c . The constraints are formulated as follows:

$$\sum_{i \in Q(t_c)} \rho_i^j x_i + \left(\sum_{i \in F(t_c)} \rho_i^j \right) x \geq \left(\sum_{i \in F(t_c)} \rho_i^j \right), \quad j \in \mathcal{D}$$

where:

$F(t_c) \equiv$ Set of activities that have been just completed at t_c ,

$\mathcal{D} \equiv$ set of manager's pre-selected resource types whose loading graphs are to be centralized (i.e., balanced).

$\left(\sum_{i \in F(t_c)} \rho_i^j \right) \equiv$ total resource type j requirements by all activities that have been completed at the decision instance t_c .

Similarly to the previous type of constraints, the term $\left(\sum_{i \in F(t_c)} \rho_i^j \right)$, appears on both sides of inequality. On the *RHS* of the inequality, it forces the increase in the number of engaged units of type j at each subsequent t_c . On the *LHS*, it serves to set x to unity in cases when further increase in the number of engaged type j units would exceed their total availability for a project. In other words, when no candidate activities can be scheduled

at t_c , such that the number of engaged resource units of type j at t_c^+ is greater than the number of engaged units at t_c^- , λ becomes unity, thus indicating infeasibility.

The two types of constraints above form a mutual exclusivity for x_i 's and λ , such that the first type of constraints keep x_i 's to zero when a problem is infeasible and λ to zero when a problem is feasible. The second type of constraints sets λ to unity in cases of infeasibility. This mechanism provides a convenient facility to computer implementation of the methodology by detecting a moment of infeasibility and preventing a program from ever crashing. Notice that the set D is pre-selected by a project manager and may have as many as J members, such that the total number of both types of constraints equals $J + D$, but may be up to $2 \times J$.

Finally, to ensure an integer zero-one solution, we impose the last type of constraints as follows:

$$x_i = 0 \text{ or } 1, \quad \text{for } i \in Q(t_c)$$

As previously discussed, once λ becomes unity, we adjust the objective function and modify the constraints that will, from that point on, allow a decrease in resource loading graph(s). Objective function for activity scheduling is modified such that the product $w_i' \cdot x_i$ is not being subtracted from one any more, while the second type of constraints is eliminated completely:

$$\min \left(- \sum_{i \in Q(t_c)} w_i^f \cdot x_i \right) - W \left(\sum_{i \in Q(t_c)} w_i^r \cdot x_i \right)$$

subject to:

$$\sum_{i \in Q(t_c)} \rho_i^j \cdot x_i \leq \left(R_j - \sum_{i \in G(t_c)} \rho_i^j \right), \quad j = 1, \dots, J$$

$$x_i = 0 \text{ or } 1$$

Since the second type of constraints is eliminated, resource loading function is now allowed to decrease. The first type of constraints still remains in place to prevent any overuse of available resources.

An algorithmic summary of the entire methodology, including both activity scheduling and resource mapping to newly scheduled activities is listed in Appendix A. The assessment of performance of the algorithm presented in this chapter and its implementation are fully discussed in Appendix C.

V. SUMMARY

5.1 Conclusions

The model developed in this research represents an initial step towards a more comprehensive resource-activity integration in project scheduling and management. It provides for both effective activity scheduling based on dynamically updated activity attributes, as well as intelligent iterative mapping of resources to each activity based on resource characteristics and pre-selected shape of project manager's objectives. The model consists of two complementary procedures: an *activity scheduler* and *resource mapper*. The procedures are alternatively being executed throughout the scheduling process at each newly detected decision instance, such that the final output is capable of providing decision support and recommendations with respect to both, scheduling project activities and resource assignments. This approach allows human, social, as well as technical resources to interact and be utilized in value creating ways, while facilitating effective resource tracking and job distribution control.

5.2 Major Research Contributions

The principal contribution of this research work is the development of a project scheduling model that:

- ◇ preserves principal resource values by providing more suitable job assignments and task distributions.
- ◇ allows incorporation of interactive dependencies among resources relative to any of their characteristics.
- ◇ facilitates effective resource tracking, resource utilization relative to the total project duration, and relative resource cost comparisons.
- ◇ allows for dynamic, yet intelligent resource assignment guidance by enabling a project manager to express his or her tacit knowledge or discretionary input by pre-specifying objective functions.
- ◇ the scheduling and mapping output provides complimenting decision support with respect to both activities and resources, and it provides detailed recommendations of which resource units should be assigned to each project activity.
- ◇ the model is relevant for managerial practice while within the rigor of academic standards and assumptions. It has been implemented with an idea to be an open model, customizable, and applicable across various operational settings.

5.3 Future Research

Many feasible directions remain open for the future research. One should certainly include modeling that would incorporate learning and forgetting effects into resource-activity mapping. Learning generally implies improvement in efficiency by repeating an activity (Badiru, 1995). Considering traditional learning curve analysis would require information from past projects. However, the present model is already capable of considering “local” learning/forgetting effects which only require manager’s estimate of how much a resource unit’s performance on the current project may improve or worsen by delaying an assignment for a later time. This can easily be modeled by applying previously discussed window functions which are capable of filtering out learning/forgetting information that is not associated with the current scheduling (decision) instance.

Future research should also facilitate for pre-emptive scheduling. The current model does not support or allow any splitting or prolongation of project activities.

A very relevant problem in knowledge intensive environments and critically skilled settings is *reassignment* of people with a particular skill to accommodate the needs of a new program or project (Cooprider, 1999). In other words, an effective strategy is needed for reallocation of those resources that have already been previously assigned to activities and distributed.

Final stage would be the development of a strategy capable of resource-activity mapping across multiple projects. In such a scenario, all previously discussed resource characteristics could also vary across projects. Other factors such as location and transportation would here also be of interest in problem modeling.

VI. REFERENCES

- Ahuja, H. N. (1976). *Construction Performance Control by Networks*, John Wiley and Sons, Inc., New York.
- Atabakhsh, H. (1991). "A Survey of Constrained Based Scheduling Systems Using an Artificial Intelligence Approach", *Artificial Intelligence in Engineering*, Vol. 6, No. 2, p. 58 - 73.
- Badiru, Adedeji B. (1993). "Activity Resource Assignments Using Critical Resource Diagramming", *Project Management Journal*, Vol. 14, No. 3, p. 15 - 21.
- Badiru, Adedeji B. (1995). "Incorporating Learning Curve Effects into Critical Resource Diagramming", *Project Management Journal*, Vol. 26, No. 2, p. 38 - 45.
- Badiru, Adedeji B. and P. Simin Pulat (1995). *Comprehensive Project Management: Integrating Optimization Models, Management Principles, and Computers*, Prentice Hall, New Jersey, p. 162 - 209.
- Badri, Masood A. (1996). "A Two Stage Multi Criteria Model for Scheduling Faculty-Course-Time Assignment", *European Journal of Operational Research*, Vol. 96, p. 16 - 28.
- Badri, Masood A., Donald L. Davis, Donna F. Davis, and John Hollingsworth (1998). "A Multi-Objective Course Scheduling Model: Combining Faculty Preferences for Courses and Times", *Computers in Operations Research*, Vol. 25, No. 4, p. 303 - 316.
- Bandelloni, M, M. Tucci, and R. Rinaldi (1994). "Optimal Resource Leveling using Non-serial Dynamic Programming", *European Journal of Operational Research*, Vol. 78, p. 162-177.
- Bein, W. W., J. Kamburowski, and M. F. M. Stallmann (1992). "Optimal Reduction of Two Terminal Directed Acyclic Graphs", *SIAM Journal on Computing*, Vol. 21, p. 1112 - 1129.
- Belhe, Upendra and Andrew Kusiak (1997). "Dynamic Scheduling of Design Activities with Resource Constraints", *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, Vol. 27, No. 1, p. 105 - 111.
- Boctor, Fayez F. (1996). "A New and Efficient Heuristic for Scheduling Projects with Resource Constraints and Multiple Execution Modes", *European Journal of Operations Research*, Vol. 90, p. 349 - 361.

- Bracewell, Ronald N. (1978). *The Fourier Transform and its Applications*. McGraw-Hill, Inc., New York, p. 97.
- Brucker, Peter, Andreas Drexl, Rolf Mohring, Klaus Neumann, and Erwin Pesch (1999). "Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods", *European Journal of Operations Research*, Vol. 112, p. 3 - 41.
- Brucker, Peter, Sigrid Knust, Arno Schoo, Olaf Thiele (1998). "A Branch and Bound Algorithm for the Resource-Constrained Project Scheduling Problem", *European Journal of Operational Research*, Vol. 107, p. 272 - 288.
- Burgess, A. R. and James B. Killebrew (1962). "Variation in Activity Level on a Cyclical Arrow Diagram", *The Journal of Industrial Engineering*, Vol. 13, No. 2, p. 76 - 83.
- Campbell, Gerard M (1999). "Cross-Utilization of Workers Whose Capabilities Differ". *Management Science*, Vol. 45, No. 5, p. 722 - 732.
- Carraway, Robert L. and Robert L. Schmidt (1991). "An Improved Discrete Dynamic Programming Algorithm for Allocating Resources among Interdependent Projects", *Management Science*, Vol. 37, No. 9, p. 1195 - 1200.
- Chang, T. C., and K. C. Crandall (1990). "An Algorithm for Solving Expected Possibility and its Application in Construction Resource Allocation", *Fuzzy Sets and Systems*, Vol. 34, p. 157-171.
- Christofides, N., R. Alvarez-Valdes, J. M. Tamarit (1987). "Project Scheduling with Resource Constraints: A Branch and Bound Approach", *European Journal of Operational Research*, Vol. 29, p. 262 - 273.
- Coopridge, Curt (1999). "Solving a Skill Allocation Problem", *Production and Inventory Management Journal*, Third Quarter, p. 1 - 6.
- Davis, K. Roscoe, Antonie Stam, and Ronald A. Grzybowski (1992). "Resource Constrained Project Scheduling with Multiple Objectives: A Decision Support Approach", *Computers in Operations Research*, Vol. 19, No. 7, p. 657 - 669.
- Demeulemeester, Erik (1995). "Minimizing Resource Availability Costs in Time-Limited Project Networks", *Management Science*, Vol. 41, No. 10, p. 1590 - 1598.
- Demeulemeester, Erik L., Willy S. Herroelen, and Salah E. Elmaghraby (1996). "Optimal Procedures for the Discrete Time/Cost Trade-Off Problem in Project Networks", *European Journal of Operational Research*, Vol. 88, p. 50 - 68.

- De Reyck, Bert and Willy Herroelen (1998). "A Branch and Bound Procedure for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations", *European Journal of Operational Research*, Vol. 111, p. 152 - 174.
- De Reyck, Bert, and Willy Herroelen (1998a). "An Optimal Procedure for the Resource-Constrained Project Scheduling Problem with Discounted Cash Flows and Generalized Precedence Relations", *Computers in Operations Research*, Vol. 25, No. 1, p. 1 - 17.
- Doucette, Martin (1998). *Microsoft® Project*. IDG Books Worldwide.
- Drexl, A. (1991). "Scheduling of Project Networks by Job Assignment", *Management Science*, Vol. 37, p. 1590 - 1602.
- Drexl, Andreas (1991). "Scheduling of Project Networks by Job Assignment", *Management Science*, Vol. 37, No. 12, p. 1590 - 1602.
- Easa, S. M. (1989). "Resource Leveling in Construction Optimization", *Journal of Construction Engineering and Management*, Vol. 115, No. 2, p. 302 - 316.
- Ecker, Klaus H. (1999). "Scheduling of Resource Tasks", *European Journal of Operations Research*, Vol. 115, p. 314 - 327.
- Elmaghraby, Salah E. (1993). "Resource Allocation via Dynamic Programming in Activity Networks", *European Journal of Operational Research*, Vol. 64, p. 199 - 215.
- Faaland, Bruce, and Tim Schmitt (1993). "Cost-Based Scheduling of Workers and Equipment in a Fabrication and Assembly Shop", *Operations Research*, Vol. 41, No. 2, p. 253 - 268.
- Franz, Lori S. and Janis L. Miller (1993). "Scheduling Medical Residents to Rotations: Solving the Large Scale Multiperiod Staff Assignment Problem", *Operations Research*, Vol. 41, No. 2, p. 269 - 279.
- Fulkerson, D. R. (1961). "A Network Flow Computation for Project Cost Curves", *Management Science*, Vol. 7, p. 167 - 178.
- Gray, Jennifer J., Don McIntire, and Herbert J. Doller (1993). "Preferences for Specific Work Schedulers: Foundation for an Expert-System Scheduling Program", *Computers in Nursing*, Vol. 11, No. 3, p. 115-121.
- Hapke, M., A. Jaskiewicz, and R. Slowinski (1994). "Fuzzy Project Scheduling System for Software Development", *Fuzzy Sets and Systems*, Vol. 67, p. 101-117.

- Herroelen, Willy, Bert De Reyck, and Erik Demeulemeester (1998). "Resource-Constrained Project Scheduling: A Survey of Recent Developments", *Computers in Operations Research*, Vol. 25, No. 4, p. 279 - 302.
- Hori, M., Y. Nakamura, H. Satoh, K. Maruyama, T. Hama, S. Honda, T. Takenaka, and F. Sekine (1995). "Knowledge-Level Analysis for Eliciting Composable Scheduling Knowledge", *Artificial Intelligence in Engineering*, Vol. 9, p. 253-264.
- <http://www.ima.mdh.se/tom> (Website for TOMLAB v1.0 optimization environment package)
- Hussein, M. L. and M. A. Abo-Sinna (1995). "A Fuzzy Dynamic Approach to the Multicriterion Resource Allocation Problem", *Fuzzy Sets and Systems*, Vol. 69, p. 115-124.
- Icmeli-Tukel, Oya and Walter O. Rom (1997). "Ensuring Quality in Resource Constrained Project Scheduling", *European Journal of Operations Research*, Vol. 103, p. 483 - 496.
- Keeney, Ralph L. and Howard Raiffa (1993). *Decisions with multiple objectives: preferences and value tradeoffs*, Cambridge University Press, Cambridge, New York.
- Kelly, J. E. (1961). "Critical Path Planning and Scheduling: Mathematical Basis", *Operations Research*, Vol. 9, p. 296 - 320.
- Khattab, Mostafa M. and F. Choobineh (1991). "A New Approach for Project Scheduling with a Limited Resource", *International Journal of Production Research*, Vol. 29, No. 1, p. 185 - 198.
- Konstantinidis, P. D. (1998). "A Model to Optimize Project Resource Allocation by Construction of a Balanced Histogram", *European Journal of Operational Research*, Vol. 104, p. 559-571.
- Kostreva, M. M., and P. Geneviev (1989). "Nurses Preferences vs. Circadian Rhythms in Scheduling", *Nursing Management*, Vol. 20, No. 7, p. 50 - 62.
- Leachman, Robert C. and Sooyoung Kim (1993). "A Revised Critical Path Method for Networks Including Both Overlap Relationships and Variable-Duration Activities", *European Journal of Operational Research*, Vol. 64, p. 229 - 248.
- Lee, J. K. , K. J. Lee, J. S. Hong, W. Kim, E. Y. Kim, S. Y. Choi, H. D. Kim, O. R. Yang, H. R. Choi (1995). "DAS: Intelligent Scheduling Systems for Shipbuilding", *AI Magazine*, Winter 1995, p. 78-94

- Lee, S. J. and C. H. Wu (1995). "CLXPert: A Rule-Based Scheduling System", *Expert Systems with Applications*, Vol. 9, No. 2, p. 153-164.
- Li, K. Y. and R. J. Willis (1992). "An Iterative Scheduling Technique for Resource-Constrained Project Scheduling", *European Journal of Operational Research*, Vol. 56, p. 370 - 379.
- Liebowitz, J. and W. E. Potter (1995). "Scheduling Objectives, Requirements, Resources, Constraints, and Processes: Implications for a Generic Expert Scheduling System Architecture and Toolkit", *Expert Systems with Applications*, Vol. 9, No. 3, p. 423-432.
- Liou, Ay-Hwa Andy, and Ming-Tser Wu (1996). "Mapping Knowledge to Rules for Scheduling Expert Systems", *Expert Systems with Applications*, Vol. 10, No. 3, p. 341 - 350.
- Mattila, K. G. and D. M. Abraham (1998). "Resource Leveling of Linear Schedules Using Integer Linear Programming", *Journal of Construction Engineering and Management*, Vol. 124, No. 3, p. 232 - 244.
- Minciardi, R., M. Paolucci, and P. P. Puliafito (1994). "Development of a Heuristic Project Scheduler under Resource Constraints", *European Journal of Operations Research*, Vol. 79, p. 176 - 182.
- Mingozzi, A., V. Maniezzo, S. Ricciardelli, and L. Bianco (1998). "An Exact Algorithm for the Resource-Constrained Project Scheduling Based on a New Mathematical Formulation", *Management Science*, Vol. 44, p. 714 - 729.
- Morse, Lucy C., John O. McIntosh, and Gary E. Whitehouse (1996). "Using Combinations of Heuristics to Schedule Activities of Constrained Multiple Resource Projects", *Project Management Journal*, March 1996, p. 34 - 40.
- Mueller, C. W. and J. C. McCloskey (1990). "Nurses' Job Satisfaction: A Proposed Measure", *Nursing Research*, Vol. 39, No. 2, p. 113 - 117.
- Nasution, S. H. (1994). "Critical Path Method", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24, No. 1, p. 48-57.
- Nazareth, Terence, Sanjay Verma, Subir Bhattacharya, Amitava Bagchi (1999). "The Multiple Resource Constrained Project Scheduling Problem: A Breadth-First Approach", *European Journal of Operations Research*, Vol. 112, p. 347 - 366.
- Nowicki, E. and C. Smutnicki (1994). "A Decision Support System for the Resource Constrained Project Scheduling Problem", *European Journal of Operations Research*, Vol. 79, p. 183 - 195.

- Ntuen, C. A., and E. H. Park (1995). "An Experiment in Scheduling and Planning of Non-Structured Jobs: Lessons Learned from Artificial Intelligence and Operational Research Toolbox", *European Journal of Operational Research*, Vol. 84, p. 96-115.
- Park, S. J., J. W. Kim, and H. W. Kang (1996). "Heuristic Knowledge Representation of Production Scheduling: An Integrated Modeling Approach", *Expert Systems with Applications*, Vol. 10, No. 3/4, p. 325-339.
- Patterson, J. H., R. Slowinski, F. B. Talbot, and J. Weglarz (1989). "An Algorithm for General Class of Precedence and Resource Constrained Scheduling Problems", *Advances in Project Scheduling* by R. Slowinski and J. Weglarz (Editors), Elsevier, Amsterdam, p. 3 - 28.
- Rao, S. S. (1979). *Optimization: Theory and Applications*, John Wiley and Sons, New York, p. 533 - 551.
- Roberts, Stephen M. (1992). "Human Skills - Keys to Effectiveness", *Cost Engineering*, Vol. 34, No. 2, p. 17-19.
- Sauer, Jurgen and Ralph Burns (1997). "Knowledge-Based Scheduling Systems in Industry and Medicine", *IEEE Expert*, January-February 1997, p. 24-31.
- Seibert, J. E., and G. W. Evans (1991). "Time-Constrained Resource Leveling", *Journal of Construction Engineering and Management*, Vol. 117, No. 3, p. 503-520.
- Sipos, Andrew (1992). "Duration Analysis", *Cost Engineering*, Vol. 34, No. 2, p. 9 - 14.
- Slowinski, R., B. Soniewicki, and J. Weglarz (1994). "DSS for Multiobjective Project Scheduling subject to Multiple-Category Resource Constraints", *European Journal of Operational Research*, Vol. 79, p. 220 - 229.
- Sprecher, A., and Drexl (1998). "Solving Multi-Mode Resource-Constrained Project Scheduling Problems by a Simple, General and Powerful Sequencing Algorithm", *European Journal of Operational Research*, Vol. 107, 431 - 450.
- Sprecher, A., S. Hartmann, and A. Drexl (1997). "An Exact Algorithm for Project Scheduling with Multiple Nodes", *OR Spektrum*, Vol. 19, 195 - 203.
- Stinson, J. P., E. W. Davis, and B. M. Khumawala (1978). "Multiple Resource-Constrained Scheduling Using Branch and Bound", *AIIE Transactions*, Vol. 10, p. 252 - 259.

- Sung, C. S., and S. K. Lim (1994). "A Project Activity Scheduling Problem with Net Present Value Measure", *International Journal of Production Economics*, Vol. 37, p. 177 - 187.
- Tsang, E. P. K. (1995). "Scheduling Techniques - A Comparative Study", *BT Technology Journal*, Vol. 13, January 1995, p. 16-28.
- Ulusoy, Gunduz and Linet Ozdamar (1989). "Heuristic Performance and Network/Resource Characteristics in Resource-Constrained Project Scheduling", *Journal of Operational Research Society*, Vol. 40, No. 12, p. 1145 - 1152.
- Ulusoy, G. and L. Ozdamar (1994). "A Constraint-Based Perspective in Resource Constrained Project Scheduling", *International Journal of Production Research*, Vol. 32, No. 3, p. 693 - 705.
- Wiers, V.C.S (1997). "A Review of the Applicability of OR and AI Scheduling Techniques in Practice", *Omega*, Vol. 25, No. 2, 0. 145-153.
- Wu, R. W. K., and F. C. Hadipriono (1994). "Fuzzy Modus Ponens Deduction Technique for Construction Scheduling", *Journal of Construction Engineering and Management*, Vol. 120, No. 1, p. 162-179.
- Wu, Y. and C. Li (1994). "Minimal Cost Project Networks: The Cut Set Parallel Difference Method", *Omega, International Journal of Management Science*, Vol. 22, No. 4, p. 401 - 407.
- Yang, Kum-Khiong (1996). "Effects of Erroneous Estimation of Activity Durations on Scheduling and Dispatching a Single Project", *Decision Sciences*, Vol. 27, No. 2, p. 255 - 290.
- Yura, Kenji (1994). "Production Scheduling to Satisfy Worker's Preferences for Days Off and Overtime Under Due-Date Constraints", *International Journal of Production Economics*, Vol. 33, p. 265 - 270.

APPENDIX A

Complete Heuristic for Dynamic Mapping Resource Units to Project Activities

A complete procedure that combines all the previously defined inputs and objectives to perform dynamic mapping of project resources is described below:

Initialize $y_i^{j,k}$ (variable that indicates which resource unit is mapped to which activity)

If activities' duration is resource dependent

 compute d_i for each activity

Perform the unconstrained CPM to obtain *EST* and *LST* times

Initialize set of *scheduled* activities to zero

Initialize set of *finished* activities to zero (set of activities completed at each t_c)

Initialize set of *newly added* activities to zero (set of activities just scheduled at each t_c)

Initialize set of *in progress* activities to zero (set of activities that are in progress at each t_c)

Initialize *time* to zero

If resource centralizing is selected

 Set the centralizing weight \mathcal{W} to user specified value

 Set the *centralizing direction* to *up* (indicating the attempt to keep increasing...

 ...the resource loading until the peak is reached)

For each project activity, calculate the number of immediate successors (*numsucc*)...

... and scale it by a maximum number of immediate successors in the network

Until all the project activities are scheduled, DO

 If *scheduled* is nonempty

 Update *time* to the next instant corresponding to the smallest...

 ...activity duration from the *scheduled* set added to the current *time*

 Update the set *finished* to include all activities that are completed...

 ...by the newly updated *time*

 Reset the set *newly added* to zero

 Update the set *in progress* to filter out the finished activities

 Update the precedence relationships to exclude the finished activities...

 ...from *in progress* and allow the successors to be scheduled

 At new *time* compose the *candidate* set of candidate activities...

 ...whose predecessors have just finished at *time*

If *direction* is set to *up*

Schedule_up activities from the *cand* set

 Update the multiplier \mathcal{W} (to a smaller value as previously discussed)

 If *peak flag*, \hat{x} becomes one

 Modify the optimizing constraints that force the non-decrease...

 ...in resource loading

 Reset the weight to its original user selected value

 Reset the *direction* to *down*

 else

Schedule_down the activities without additional constraints forcing the...

 ...non-decrease in resource utilization

Reset the *newly_added* set and fill it with activities scheduled at *time*

For each of the resource types starting from type one:

 Map the resource units of the current type optimizing the user...

 ...selected or formulated objective

 Update $y_i^{j,k}$ and set it to one if unit k is mapped to activity i .

If activities' duration is resource dependent

 Update duration of each activity to the longest time any of the...

 ...mapped resource units would take to complete its task on that activity

Update the *scheduled* set

Update the *in_progress* activity set to include the newly added activities

Update the precedence relationship to account for newly added activities

End DO

APPENDIX B

OVERVIEW OF *PROMAP* (PROJECT RESOURCE MAPPER) SOFTWARE

To run *PROMAP*, type *promap* at the Matlab prompt. A menu window will appear as shown in Figure B1, with three main menu titles: *Project*, *Run*, and *Graph*.



Figure B1. *PROMAP*'s Main Menu.

The *Project* menu has the following menu items: *New Project*, *Open Project*, *Save Project*, and *Close*.

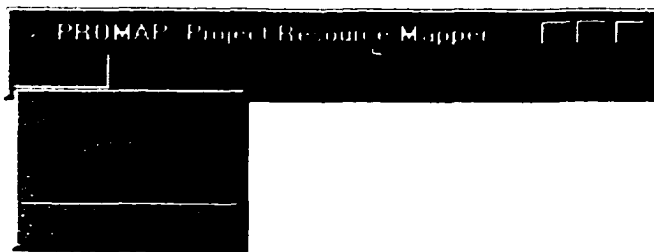


Figure B2. *Project* Menu Items.

Just like in conventional software, *New Project* will prompt the user to enter the data for a newly created project. *Open Project* will open a file that contains a previously stored

project data. *Save Project* will save the basic data of the currently opened or created project.

By selecting *New Project*, a new window will appear prompting the user to enter the basic project data: number of activities, number of resource types, number of units of each resource type available for the project, activity requirements for the number of units of each type, and activity precedence relations. The window is shown in Figure B3.

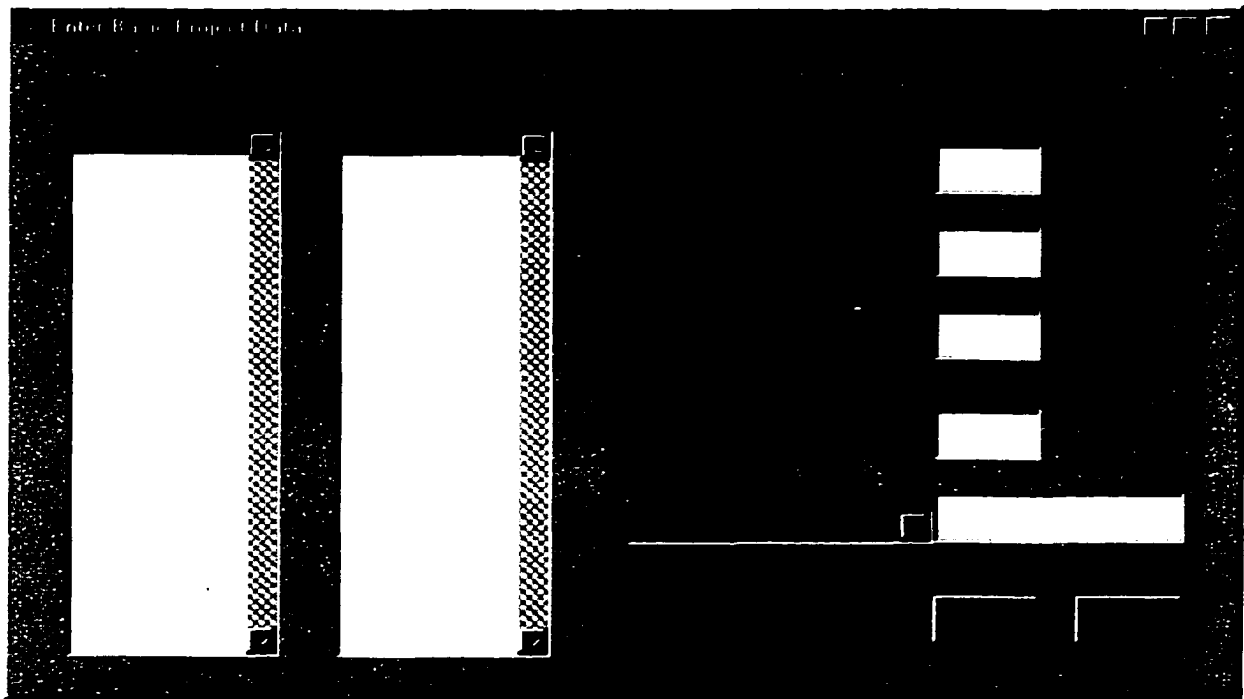


Figure B3. Window for Entering the Basic Project Data.

Once the user enters the number of activities and resource types, they appear in the list boxes on the left hand side of the window as shown in Figure B4.

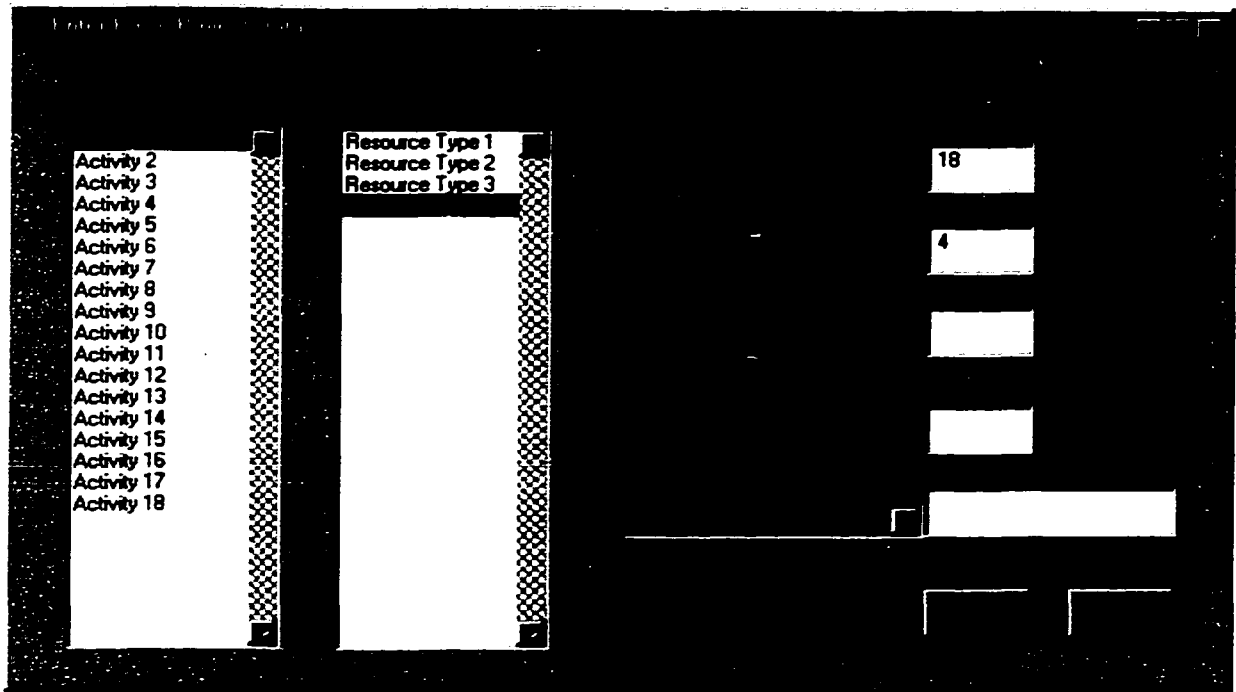


Figure B4. Use of List Boxes to Display Project Activities and Resource Types.

Each time the user enters the number of units available of each type, the number on the left of the edit text box increases by one, as shown in Figure B5.



Figure B5. Text Box for Entering Availability of Resource Types.

Activity requirements and precedences are entered in the same fashion, except that there is a pull-down menu provided to target specific activities when entering the precedences as shown in Figure B6.

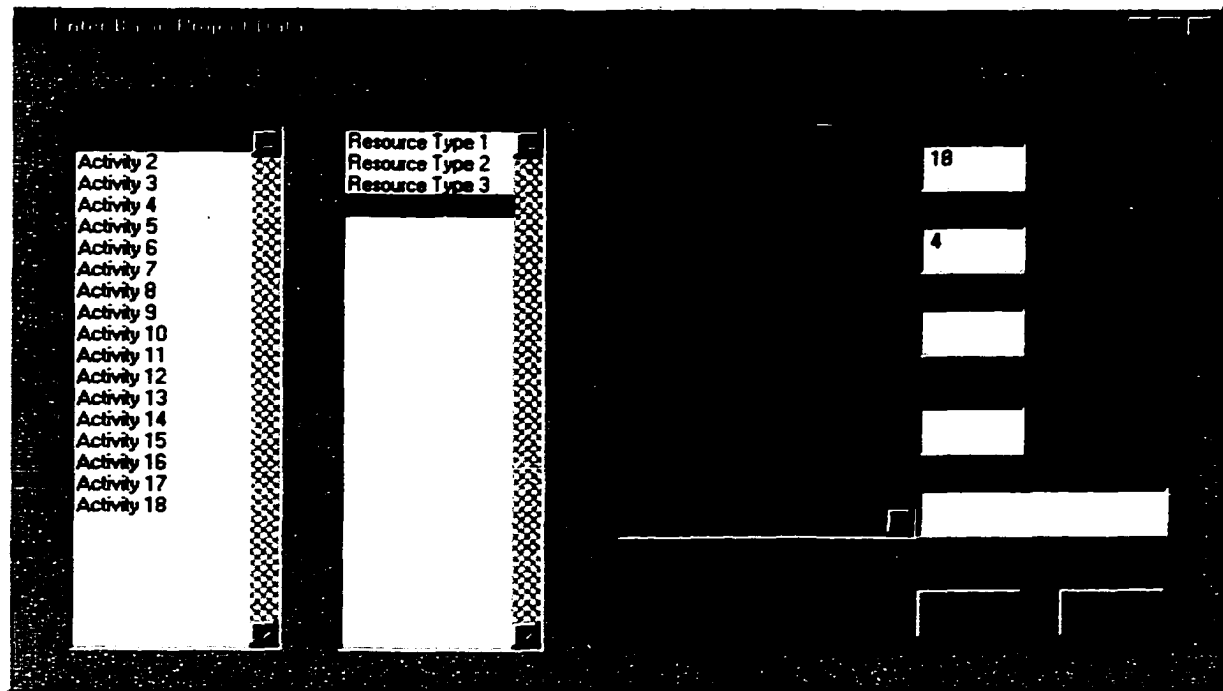


Figure B6. Pull-Down Menu that Facilitates the Entering of Activity Precedence Relations.

Once all the data has been entered the user should press *Accept All* and *Exit* the window.

As soon as the window in Figures B3 through B6 closes, a new window appears as shown in Figure B7.

Figure B7. Window for Entering Functional Dependencies among Resources.

The list boxes on top of the window display activities, resource types, and the number of resource units associated with each highlighted resource type. Below the list boxes are the text edit boxes which will either display the activities and resources that the user has highlighted or enable the user to manually enter the inputs. *Reference Activity* is simply the activity i with respect to which dependencies are inputted. A more than one, and up to the size of I activities may be entered. The user then must select *dependent* and *driver* resources in the middle third of the window. If no *driver* resources are specified, the

program assumes that there either is no interaction, or that the currently entered resource is a driver itself (and thus cannot depend on any other resources).

The lower third of the window is where the final project data are entered. The first pull-down menu displays four items: *Varying Resource Time Requirements*, *Desired Resource Start Time*, *Resource Interval Availability* (which refers to the Resource Time Window), and *Fixed Activity Duration*. The choices are shown in Figure B8.

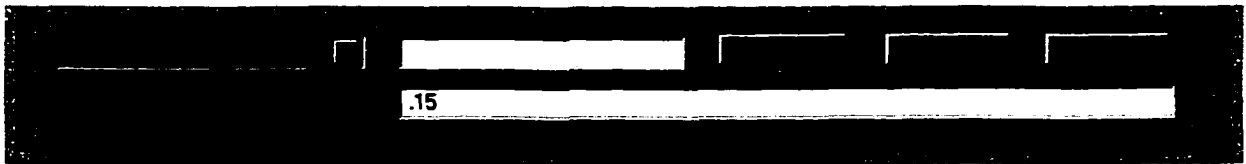


Figure B8. Pull-Down Menu Items for Entering Specific Resource Data.

Varying Resource Time Requirements refers to $t_i^{j,k}$'s. In cases when all the dependencies are nonexistent or implicit, the user will resort to this option. *Desired Resource Start Time* refers to $\tau_i^{j,k}$. When a single number is entered for a *Desired Start Time*, it is then assumed that $a=1$ and $b=2$. Otherwise, the project manager may enter the parameters as $[a, b, \tau_i^{j,k}]$ as a vector. *Resource Interval Availability* requires the user to enter a two dimensional vector in the form of " $[t_{LO}, t_{HI}]$ ", where t_{LO} and t_{HI} refer to the strict time box constraints. *Fixed Activity Duration* is a feature that pertains only to activities and assumes that activity duration is independent of which resource units are mapped to it. This feature is added to facilitate traditional project scheduling where the manager

estimates activity durations prior to resource assignment. At any time the user may enter any number as a fuzzy number by typing $fuz(a,b,c,d)$ where a , b , c , and d , are the edges of a trapezoidal fuzzy number. The subroutine is smart enough to recognize a triangular fuzzy number in cases when the user enters $fuz(a,b,c)$. The number is defuzzified using a formula proposed by Lee and Li (1988):

$$\bar{X} = \frac{(-a^2 - b^2 + c^2 + d^2 - ab + cd)}{3(-a - b + c + d)}$$

Figure B9 displays the items under the lower pull-down menu. They actually enable the user to enter the *Time Dependencies*, *Preferences*, and *Costs* as defined in the methodology. User may enter any number of the dependencies, preferences or costs and they will be evaluated and compiled to determine the coefficients of the objective function.

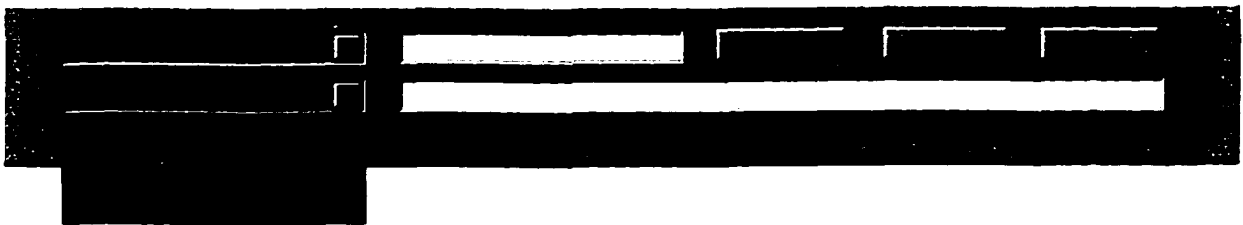


Figure B9. Pull-Down Menu Items.

Once all the project data has been entered, it may be saved before scheduling or for later use. The data saved by *Save Project* or retrieved by *Open Project* include the number of

activities and resource types, project availability for each of the resource types, precedence relations among activities, activity requirements for the units of each resource type, and basic dependencies, costs, preferences, and time constraints as previously user-defined.

Close Item under the *Project* Menu will terminate the program.

The menu *Run* has seven items: *Schedule*, *Optimizing Objectives*, *Set Centralizing Importance Level*, *Map and Centralize*, *Centralize Only*, and *Map Only*. *Schedule* simply schedules the activities, and depending on the user choice also centralizes and or maps the resources to activities. The menu items under the *Run* are shown in Figure B10.

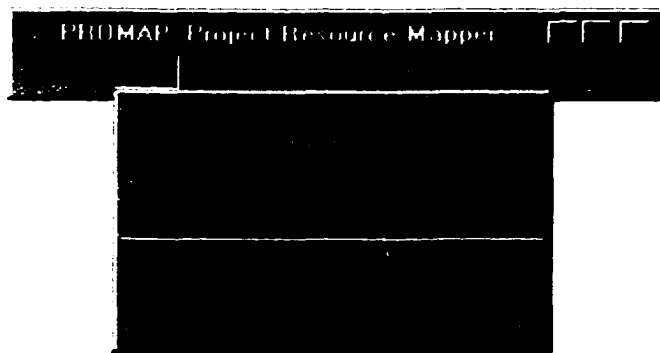


Figure B10. Run Menu Items.

When selected, *Optimizing Objectives* invokes a list dialog box which enables the user to select one of the four objectives shown in Figure B11 and as defined in the methodology: *Time Effectiveness*, *Preferences*, *Costs*, and *Resource Availability*. Alternatively, the user

may also select a *Composite Utility Function*, which will open another input dialog box and prompt the user to enter the formula for the utility.

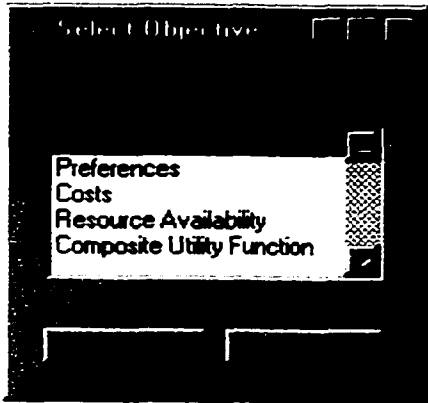


Figure B11. Choices of User Selected Objectives according to which Resources are to be Mapped.

The utility function input dialog box is shown in Figure B12. The user is cautioned that the variable pertaining to functional time dependencies must be called *timedep*, the variable for preferences must be entered as *pref*, and the other two variables are *cost* and *starttime*.

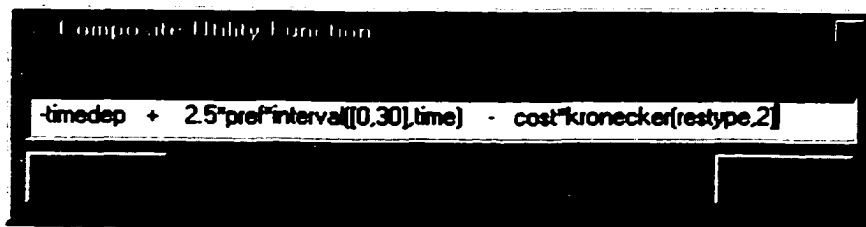


Figure B12. Dialog Box for Entering Optimizing Utility Functions.

An example utility function is shown in Figure B12. The formula shows that *the time effectiveness* will be the optimizing objective for all resource types, except for the type two which, in addition, also requires the optimization of costs. *kroncker* is a subroutine, named after Kronecker's Delta Function (Bracewell, 1978) and used in this research to compare the current resource type to the input, and if they are equal, the subroutine returns the value of one, otherwise it becomes zero. Thus, the third part of the utility is nonzero only during the mapping of resource type two. To facilitate for resource preferences, a project manager may want to consider incorporating them into the utility, but only for the first 30 time units when timing is not of exclusive importance. Thus, the function *interval([from, to], time)* is used to filter out those additive components of the utility that are not associated with the current *time*, that is, current decision instance.

The next menu item under *Run* is *Set Centralizing Importance Level*. This, when selected prompts the user to enter the weight, \mathcal{W} for balancing the objective function when scheduling activities. \mathcal{W} was also discussed in methodology, and the dialog box is shown in Figure B13. If omitted, the default value that \mathcal{W} takes is zero.

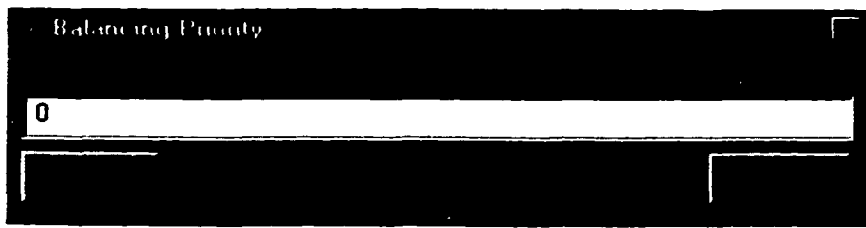


Figure B13. Dialog Box for Entering Resource Centralizing Level.

By selecting the *Resource Types to Centralize*, the program invokes another list dialog box that lists all the resource types and asks the user to select those whose loading the program should attempt to centralize. In other words, the user is asked to define the elements of the set \mathcal{S} , which was discussed in the methodology. The list box is shown in Figure B14.

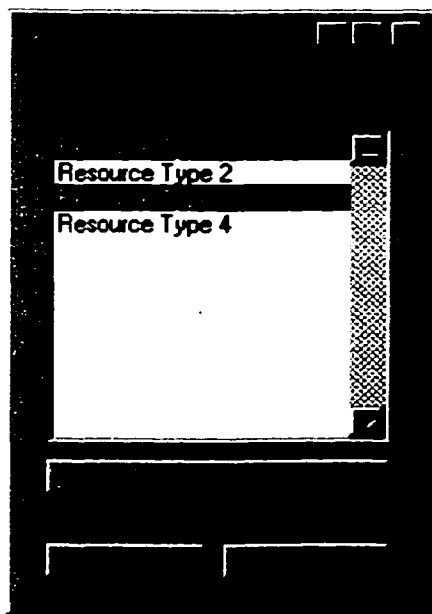


Figure B14. List Box for Selection of Resource Types to be Centralized.

The final three items under the menu *Run* are *Map and Centralize*, *Centralize Only*, and *Map Only*. The first option will dynamically attempt to first, at each decision instance t_c , schedule activities such that the resource loading is centralized, and then map the resources units to each of the newly scheduled activities. The second option only attempts to centralize the resource utilization and allocate enough resource units, but it does not perform the discrimination and mapping of distinct units to scheduled activities. This speeds up the scheduling significantly, and is useful in cases when all the units are generic, indistinguishable, and without specific costs, preferences, or dependencies. The last option only maps the resources to activities but skips the attempt to centralize their loading by suppressing w to zero.

Finally, the menu title *Graph* offers the graphical solutions of the scheduled project. The items under *Graph* are displayed in Figure B15.

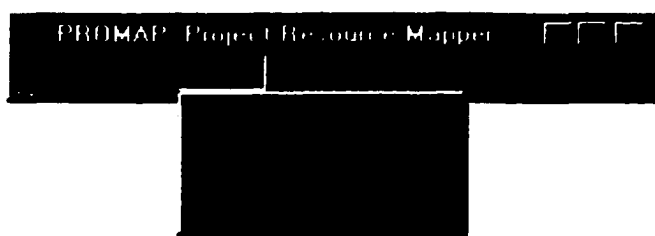


Figure B15. Items under Graph Menu.

Gantt draws a traditional Gantt chart of a scheduled project as shown in Figure B16.

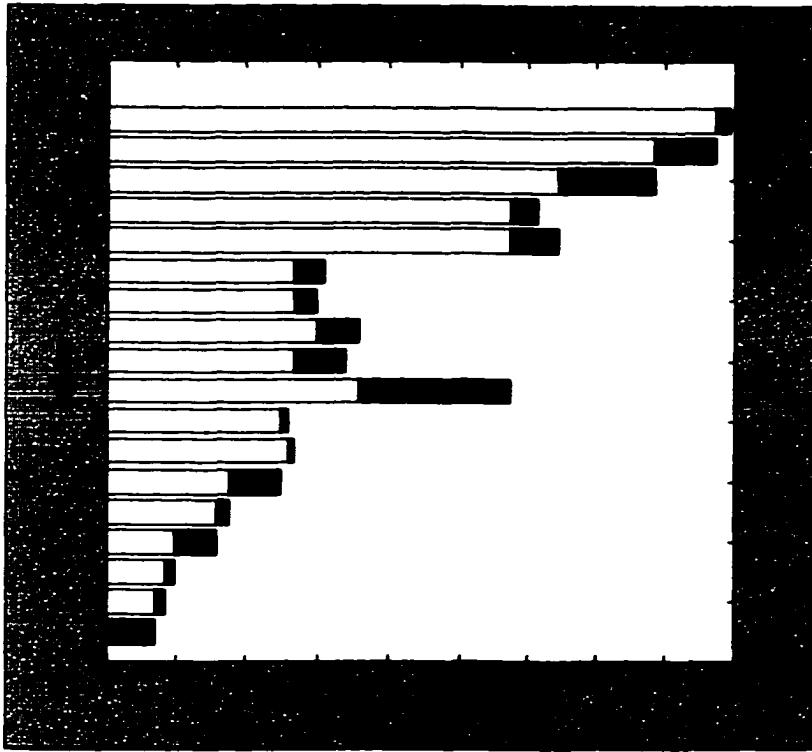


Figure B16. Example Gantt Chart by PROMAP.

Resource Loading draws the loading graph of each of the resource types. An example of a somewhat centralized resource loading graph is shown in Figure B17.

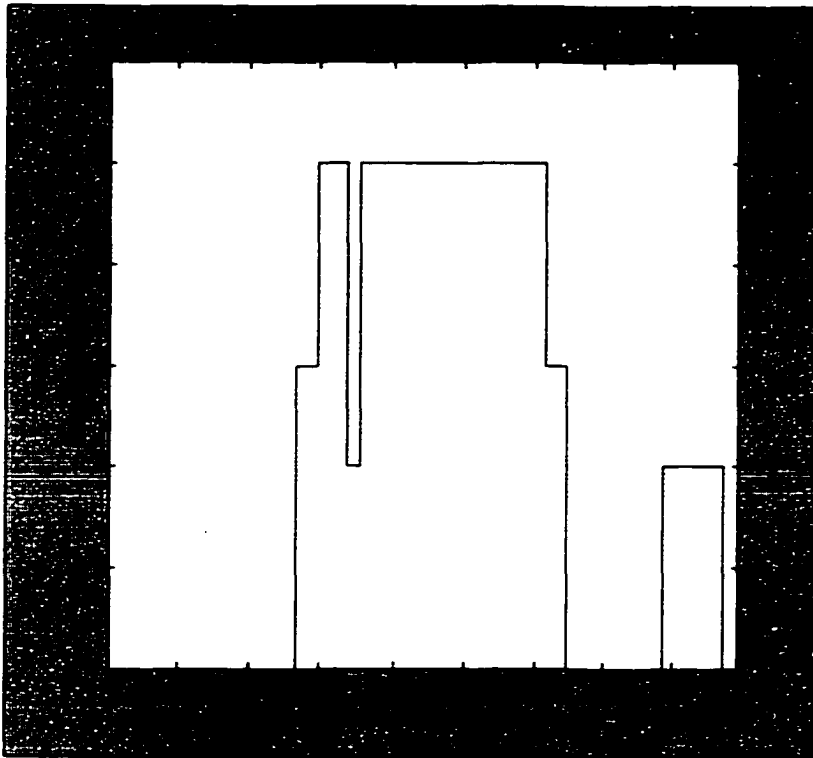


Figure B17. Example of a Resource Loading Graph.

Resource-Activity Grid displays a grid chart for each of the resource types, showing PROMAP's recommendations on which resource unit of each type should be assigned to which project activity. An example of *resource-activity grid* graph is shown in Figure B18. It should be noted that graph in Figure B18 conveys the same type of information as the network presented previously in Figure 3. For example, Figure B18 shows that the project activity 11 has three units of type 2 assigned to it. Those resource units are: unit 1, unit 2, and unit 6. Activity 13, for example, has resource units 3 and 5 assigned to it.

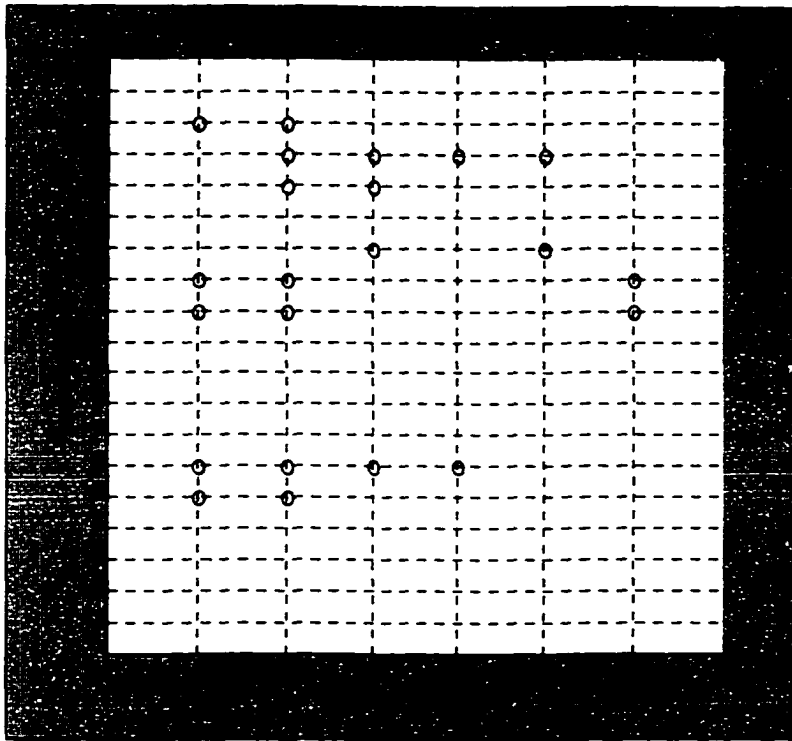


Figure B18. Example of Resource-Activity Mapping Grid.

Unit Utilization shows the expected time each resource unit is expected to be employed as a percentage of the total project duration. An example bar plot of unit utilization for a particular resource type is shown in Figure B19.

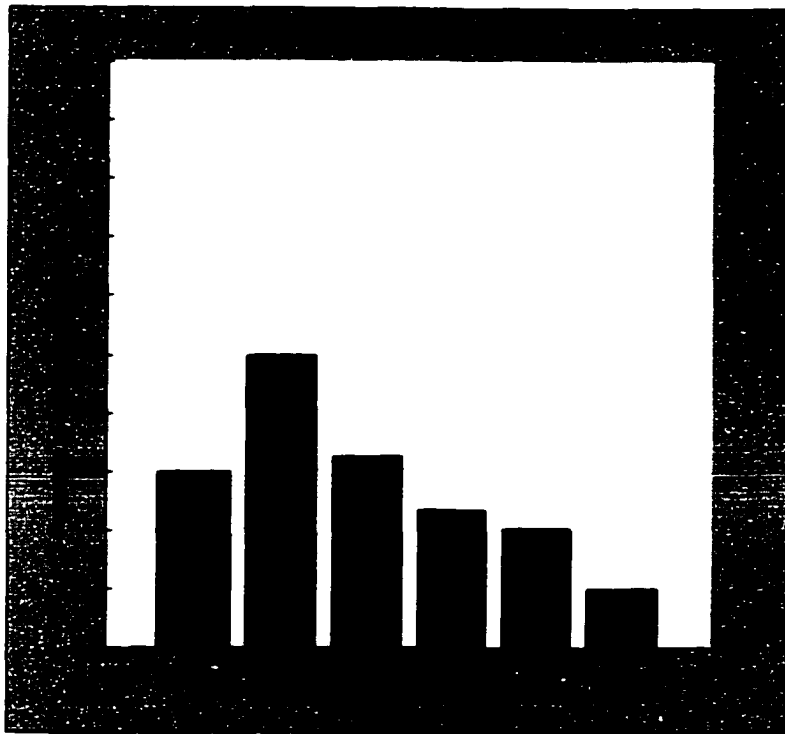


Figure B19. Resource Units Utilization Bar Chart.

The bottom bars indicate the total time it takes each unit to complete all of its own project tasks. The upper bars indicate the total additional time a unit may be locked in or engaged in an activity by waiting for other units to finish their tasks. In other words, the upper bars indicate the total resource idle time during which it cannot be reassigned to other activities because it is blocked waiting for other units to finish their own portions of work. This information is very useful in non-preemptive scheduling as assumed in this study, as well as in contract employment of resources.

Finally, we may easily monitor the cost of each resource unit as a result of its mapping to various activities. Figure B20 shows the total project cost for each unit of resource type

two. It should be noted that PROMAP displays similar plots for units of all resource types involved in the project.

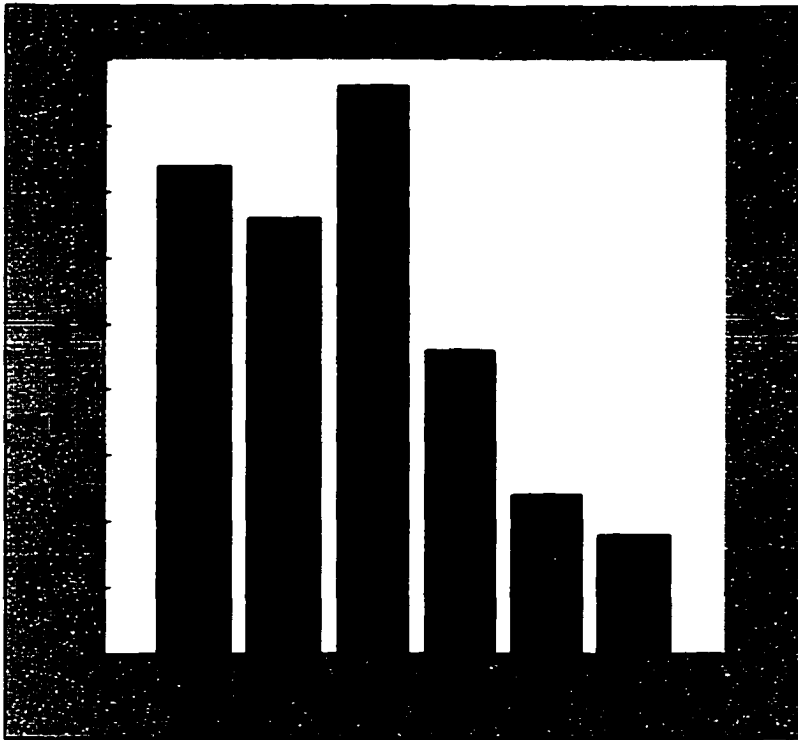


Figure B20. Example Cost Chart for Units of a Project Resource Type.

APPENDIX C

EXAMPLES OF *PROMAP* PROJECT INPUT AND OUTPUT

The two example projects in this section illustrate the power and capabilities of *PROMAP*. Both examples provide the full format and structure of the data input, as well as several output scenarios, each reflecting a result of a different scheduling-mapping objective. Both examples are heavily modified and extended from Doucette (1998). The first project consists of 18 activities, four resource types, and considers time-effective capabilities, costs, and preferences as resource characteristics used for mapping decisions. The second project has 22 activities, three resource types, and considers time-effective capabilities, preferences, and resource availability as potential mapping objective components. The input for both projects are classified and tabulated according to the above characteristics, and ordered with respect to resource interdependencies (that is, lower indexed resource data is displayed first). The first table for each project represents basic project data, such as activity names, activity precedence relations, resource types, availability of each resource type, and activity needs with respect to each resource type or group. Each subsequent table represents specific characteristics of each resource group.

The output of each project includes a *resource-activity mapping grid* which provides a recommendation of which resource units should be assigned to which activities. After each run, the *PROMAP* provides as many of such grid plots as there are resource groups or types in a project. The output also includes a *Gantt chart* showing the actual schedule

of all project activities. Further, the program's output provides traditional *resource loading graphs* which are dynamic indicators of resource usage for each resource group. An overall *resource utilization bar chart* for each resource unit as a percentage of the overall project duration is also tracked and available. Finally, the output also displays relative *total cost bar charts* of each resource unit based on its utilization.

Tables C1 and C14 display basic project data for the two projects. Tables C1-C13 illustrate resource characteristics and their interdependencies for the first project. Tables C15-C23 display resource characteristics and their interdependencies for the second fictitious project.

EXAMPLE PROJECT #1: INPUT DATA

Table C1. Basic Project #1 Data (Partially adopted from Doucette, 1998)

Act. #	Act. Name	Predecessors	Type 1	Type 2	Type 3	Type 4
			Utility Workers	Contractor Workers	Carpenters	Office Staff (including counsels)
			Max: 4	Max: 6	Max: 5	Max: 4
1	Customer selections	-				1(sales)
2	Write specifications	1				3(specs)
3	Write contract	2				2(one counsel)
4	Detail Plans	2				3(drafting)
5	Excavation	5		2(excavation)		2
6	Footing/foundation	5		4(excavation)		1
7	Water service	6	3(water)			1
8	Electrical service	6	2(electric)			1
9	Wood framing	7,8			5(framing)	1
10	Roofing	7,8			3(framing)	1
11	Plumbing lines	7,8	2(water)	3(mechanical)	2	1
12	Furnace and A/C	7,8		3(mechanical)		1
13	Electrical wiring	7,8	2(electric)	2(electric)		1
14	Wallboard	9,10, 11, 12, 13, 14			3(wallboard)	1
15	Stairway	9,10, 11, 12, 13, 14		2	2(finish)	1
16	Painting	14,15		4(painting)		1
17	Trim and Final Corrections	16		2	2	2
18	Contract and Admin. Closure	17				2

Table C2. Time-Effective Capabilities For Resource Type 1.

		Utility Worker	Utility Worker	Utility Worker	Utility Worker
Act. No.	Act. Name	1(water)	2(water, electric)	3(electric, water)	4(electric, water)
1	Cust. select.				
2	Write specs				
3	Write contract				
4	Detail Plans				
5	Excavation				
6	Footing/found.				
7	Water service	1	0.6	0.6	1
8	Elect. service	-	1	1	1
9	Wood framing				
10	Roofing				
11	Plumb. lines	2.5	3	3	3
12	Furnace & A/C				
13	Electric. wiring	-	2	2	2
14	Wallboard				
15	Stairway				
16	Painting				
17	Trim and Final Corrections				
18	Contract and Admin. Closure				

Table C3. Time-Effective Capabilities For Resource Type 2

Act. No.	Act. Name	Contractor 1(excav., mech.)	Contractor 2(excav., mech.)	Contractor 3(excav., paint., mech.)	Contractor 4(excav., electr., paint.)	Contractor 5(electr., excav.)	Contractor 6(paint.,mech , electr.)
1	Cust. select.						
2	Write specs						
3	Write contract						
4	Detail Plans						
5	Excavation	2	2	2	2	2.1	3
6	Footing/found.	7	5	7	7	7.2	7.4
7	Water service						
8	Elect. service						
9	Wood framing						
10	Roofing						
11	Plumb. lines	$(T)_1=(t_{11}^{1,1}$ $+1) \cdot y_{11}^{1,1}$ $(T)_2=(t_{11}^{1,2}$ $+1) \cdot y_{11}^{1,2}$ $(T)_3=(t_{11}^{1,3}$ $+3) \cdot y_{11}^{1,3}$ $(T)_4=(t_{11}^{1,4}$ $+2) \cdot y_{11}^{1,4}$	$(T)_1=(t_{11}^{1,1}$ $+1) \cdot y_{11}^{1,1}$ $(T)_2=(t_{11}^{1,2}$ $+1) \cdot y_{11}^{1,2}$ $(T)_3=(t_{11}^{1,3}$ $+3) \cdot y_{11}^{1,3}$ $(T)_4=(t_{11}^{1,4}$ $+2) \cdot y_{11}^{1,4}$	$(T)_1=(t_{11}^{1,1}$ $+1) \cdot y_{11}^{1,1}$ $(T)_2=(t_{11}^{1,2}$ $+1) \cdot y_{11}^{1,2}$ $(T)_3=(t_{11}^{1,3}$ $+3) \cdot y_{11}^{1,3}$ $(T)_4=(t_{11}^{1,4}$ $+2) \cdot y_{11}^{1,4}$	$(T)_1=(t_{11}^{1,1}$ $+1.5) \cdot y_{11}^{1,1}$ $(T)_2=(t_{11}^{1,2}$ $+1.5) \cdot y_{11}^{1,2}$ $(T)_3=(t_{11}^{1,3}$ $+3.8) \cdot y_{11}^{1,3}$ $(T)_4=(t_{11}^{1,4}$ $+2.6) \cdot y_{11}^{1,4}$	$(T)_1=(t_{11}^{1,1}$ $+1.5) \cdot y_{11}^{1,1}$ $(T)_2=(t_{11}^{1,2}$ $+1.5) \cdot y_{11}^{1,2}$ $(T)_3=(t_{11}^{1,3}$ $+3.8) \cdot y_{11}^{1,3}$ $(T)_4=(t_{11}^{1,4}$ $+2.6) \cdot y_{11}^{1,4}$	$(T)_1=(t_{11}^{1,1}$ $+1) \cdot y_{11}^{1,1}$ $(T)_2=(t_{11}^{1,2}$ $+1) \cdot y_{11}^{1,2}$ $(T)_3=(t_{11}^{1,3}$ $+3) \cdot y_{11}^{1,3}$ $(T)_4=(t_{11}^{1,4}$ $+2) \cdot y_{11}^{1,4}$
12	Furnace & A/C	3	2.5	3	3.5	3.5	3.1

Time-Effect. Capab. For Resource Type 2		Contractor	Contractor	Contractor	Contractor	Contractor	Contractor
Act. No.	Act. Name	1(excav., mech.)	2(excav., mech.)	3(excav., paint., mech.)	4(excav., electr., paint.)	5(electr., excav.)	6(paint.,mec h., electr.)
13	Electric. wiring	$(T)_5=(t_{13}^{1,1}$ $+2)\cdot y_{13}^{1,1}$ $(T)_6=(t_{13}^{1,2}$ $+2)\cdot y_{13}^{1,2}$ $(T)_7=(t_{13}^{1,3}$ $+0.5)\cdot y_{13}^{1,3}$ $(T)_8=(t_{13}^{1,4}$ $+0.5)\cdot y_{13}^{1,4}$	$(T)_5=(t_{13}^{1,1}$ $+2.9)\cdot y_{13}^{1,1}$ $(T)_6=(t_{13}^{1,2}$ $+2.9)\cdot y_{13}^{1,2}$ $(T)_7=(t_{13}^{1,3}$ $+0.5)\cdot y_{13}^{1,3}$ $(T)_8=(t_{13}^{1,4}$ $+0.5)\cdot y_{13}^{1,4}$	$(T)_5=(t_{13}^{1,1}$ $+2)\cdot y_{13}^{1,1}$ $(T)_6=(t_{13}^{1,2}$ $+2)\cdot y_{13}^{1,2}$ $(T)_7=(t_{13}^{1,3}$ $+0.5)\cdot y_{13}^{1,3}$ $(T)_8=(t_{13}^{1,4}$ $+0.5)\cdot y_{13}^{1,4}$	$(T)_5=(t_{13}^{1,1}$ $+2.9)\cdot y_{13}^{1,1}$ $(T)_6=(t_{13}^{1,2}$ $+2.9)\cdot y_{13}^{1,2}$ $(T)_7=(t_{13}^{1,3}$ $+0.4)\cdot y_{13}^{1,3}$ $(T)_8=(t_{13}^{1,4}$ $+0.4)\cdot y_{13}^{1,4}$	$(T)_5=(t_{13}^{1,1}$ $+2)\cdot y_{13}^{1,1}$ $(T)_6=(t_{13}^{1,2}$ $+2)\cdot y_{13}^{1,2}$ $(T)_7=(t_{13}^{1,3}$ $+0.5)\cdot y_{13}^{1,3}$ $(T)_8=(t_{13}^{1,4}$ $+0.5)\cdot y_{13}^{1,4}$	$(T)_5=(t_{13}^{1,1}$ $+2.2)\cdot y_{13}^{1,1}$ $(T)_6=(t_{13}^{1,2}$ $+2.2)\cdot y_{13}^{1,2}$ $(T)_7=(t_{13}^{1,3}$ $+0.2)\cdot y_{13}^{1,3}$ $(T)_8=(t_{13}^{1,4}$ $+0.2)\cdot y_{13}^{1,4}$
14	Wallboard						
15	Stairway	3	3	3	3	3	3
16	Painting	17	14	13	13	13	17
17	Trim and Final Corrections	7	7	7	7	7	7
18	Contract and Admin. Closure						

Table C4. Time-Effective Capabilities For Resource Type 3

Time-Effect. Capab. For Resource Type 3		Carpenter	Carpenter	Carpenter	Carpenter	Carpenter
Act. No.	Act. Name	1(frame, wall)	2 (frame, wall)	3(finish, frame)	4(finish, frame)	5(wall, frame)
1	Cust. Select.					
2	Write specs					
3	Write contract					
4	Detail Plans					
5	Excavation					
6	Footing/found.					
7	Water service					
8	Elect. service					
9	Wood framing	17	20	22	21	21
10	Roofing	7	7	6	6	7

Time-Effect. Capab. For Resource Type 3		Carpenter	Carpenter	Carpenter	Carpenter	Carpenter
Act. No.	Act. Name	1(frame, wall)	2 (frame, wall)	3(finish, frame)	4(finish, frame)	5(wall, frame)
11	Plumb. lines	$(T)_1 = t_{11}^{2,1} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,1}$ $(T)_2 = t_{11}^{2,2} \cdot (1 + 0.$ $15) \cdot y_{11}^{2,2}$ $(T)_3 = t_{11}^{2,3} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,3}$ $(T)_4 = t_{11}^{2,4} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,4}$ $(T)_5 = t_{11}^{2,5} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,5}$ $(T)_6 = t_{11}^{2,6} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,6}$	$(T)_1 = t_{11}^{2,1} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,1}$ $(T)_2 = t_{11}^{2,2} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,2}$ $(T)_3 = t_{11}^{2,3} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,3}$ $(T)_4 = t_{11}^{2,4} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,4}$ $(T)_5 = t_{11}^{2,5} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,5}$ $(T)_6 = t_{11}^{2,6} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,6}$	$(T)_1 = t_{11}^{2,1} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,1}$ $(T)_2 = t_{11}^{2,2} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,2}$ $(T)_3 = t_{11}^{2,3} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,3}$ $(T)_4 = t_{11}^{2,4} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,4}$ $(T)_5 = t_{11}^{2,5} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,5}$ $(T)_6 = t_{11}^{2,6} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,6}$	$(T)_1 = t_{11}^{2,1} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,1}$ $(T)_2 = t_{11}^{2,2} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,2}$ $(T)_3 = t_{11}^{2,3} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,3}$ $(T)_4 = t_{11}^{2,4} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,4}$ $(T)_5 = t_{11}^{2,5} \cdot (1$ $+ 0.15) \cdot y_{11}^{2,5}$ $(T)_6 = t_{11}^{2,6} \cdot (1 + 0.1$ $5) \cdot y_{11}^{2,6}$	
12	Furn. & A/C					
13	Electric. wiring					
14	Wallboard	6	7	7	7	7
15	Stairway	5	5	4	4	6
16	Painting					

Time-Effect. Capab. For Resource Type 3		Carpenter	Carpenter	Carpenter	Carpenter	Carpenter
Act. No.	Act. Name	1(frame, wall)	2 (frame, wall)	3(finish, frame)	4(finish, frame)	5(wall, frame)
17	Trim and Final Corrections	$(T)_7=(t_{17}^{2,1}$ $+1) \cdot y_{17}^{2,1}$	$(T)_7=(t_{17}^{2,1}$ $+1) \cdot y_{17}^{2,1}$	$(T)_7=(t_{17}^{2,1}$ $+1) \cdot y_{17}^{2,1}$	$(T)_7=(t_{17}^{2,1}$ $+1) \cdot y_{17}^{2,1}$	$(T)_7=(t_{17}^{2,1}$ $+1) \cdot y_{17}^{2,1}$
		$(T)_8=(t_{17}^{2,2}$ $+1) \cdot y_{17}^{2,2}$	$(T)_8=(t_{17}^{2,2}$ $+1) \cdot y_{17}^{2,2}$	$(T)_8=(t_{17}^{2,2}$ $+1) \cdot y_{17}^{2,2}$	$(T)_8=(t_{17}^{2,2}$ $+1) \cdot y_{17}^{2,2}$	$(T)_8=(t_{17}^{2,2}$ $+1) \cdot y_{17}^{2,2}$
		$(T)_9=(t_{17}^{2,3}$ $+1) \cdot y_{17}^{2,3}$	$(T)_9=(t_{17}^{2,3}$ $+1) \cdot y_{17}^{2,3}$	$(T)_9=(t_{17}^{2,3}$ $+1) \cdot y_{17}^{2,3}$	$(T)_9=(t_{17}^{2,3}$ $+1) \cdot y_{17}^{2,3}$	$(T)_9=(t_{17}^{2,3}$ $+1) \cdot y_{17}^{2,3}$
		$(T)_{10}=(t_{17}^{2,4}$ $+1) \cdot y_{17}^{2,4}$	$(T)_{10}=(t_{17}^{2,4}$ $+1) \cdot y_{17}^{2,4}$	$(T)_{10}=(t_{17}^{2,4}$ $+1) \cdot y_{17}^{2,4}$	$(T)_{10}=(t_{17}^{2,4}$ $+1) \cdot y_{17}^{2,4}$	$(T)_{10}=(t_{17}^{2,4}$ $+1) \cdot y_{17}^{2,4}$
		$(T)_{11}=(t_{17}^{2,5}$ $+0.5) \cdot y_{17}^{2,5}$	$(T)_{11}=(t_{17}^{2,5}$ $+0.5) \cdot y_{17}^{2,5}$	$(T)_{11}=(t_{17}^{2,5}$ $+0.5) \cdot y_{17}^{2,5}$	$(T)_{11}=(t_{17}^{2,5}$ $+0.5) \cdot y_{17}^{2,5}$	$(T)_{11}=(t_{17}^{2,5}$ $+0.5) \cdot y_{17}^{2,5}$
		$(T)_{12}=(t_{17}^{2,6}$ $+0.5) \cdot y_{17}^{2,6}$	$(T)_{12}=(t_{17}^{2,6}$ $+0.5) \cdot y_{17}^{2,6}$	$(T)_{12}=(t_{17}^{2,6}$ $+0.5) \cdot y_{17}^{2,6}$	$(T)_{12}=(t_{17}^{2,6}$ $+0.5) \cdot y_{17}^{2,6}$	$(T)_{12}=(t_{17}^{2,6}$ $+0.5) \cdot y_{17}^{2,6}$
18	Contract and Admin. Closure					

Table C5. Time-Effective Capabilities For Resource Type 4

Time-Effect. Capab. For Resource Type 4		Office Staff	Office Staff	Office Staff	Office Staff
Act. No.	Act. Name	1(spec., draft)	2(spec., draft)	3(spec., draft)	4(couns., sales)
1	Cust. Select.	7	7	7	7
2	Write specs	1	1.5	1	4
3	Write contract				0.8
4	Detail Plans	6	4	5	11
5	Excavation				
6	Footing/ Foundation	$(T)_1 = (t_6^{2,1} + 0.5) \cdot y_6^{2,1}$ $(T)_2 = (t_6^{2,2} + 1.3) \cdot y_6^{2,2}$ $(T)_3 = (t_6^{2,4} + 0.4) \cdot y_6^{2,4}$	$(T)_1 = t_6^{2,1} \cdot (1 + 0.15) \cdot y_6^{2,1}$ $(T)_2 = (t_6^{2,2} + 1.5) \cdot y_6^{2,2}$ $(T)_3 = (t_6^{2,4} + 1.4) \cdot y_6^{2,4}$ $(T)_4 = (t_6^{2,6} + 0.4) \cdot y_6^{2,6}$	$(T)_1 = (t_6^{2,1} + 0.5) \cdot y_6^{2,1}$ $(T)_2 = (t_6^{2,2} + 0.5) \cdot y_6^{2,2}$ $(T)_3 = t_6^{2,5} \cdot (1 + 0.1) \cdot y_6^{2,5}$ $(T)_4 = t_6^{2,6} \cdot (1 + 0.05) \cdot y_6^{2,6}$	$(T)_1 = (t_6^{2,1} + 1.5) \cdot y_6^{2,1}$ $(T)_2 = (t_6^{2,2} + 2.5) \cdot y_6^{2,2}$ $(T)_3 = (t_6^{2,3} + 1.4) \cdot y_6^{2,3}$ $(T)_4 = (t_6^{2,4} + 1.4) \cdot y_6^{2,4}$ $(T)_5 = (t_6^{2,5} + 1.7) \cdot y_6^{2,5}$ $(T)_6 = (t_6^{2,6} + 1.7) \cdot y_6^{2,6}$
7	Water service				
8	Electric. service				
9	Wood framing				
10	Roofing				
11	Plumb. lines				
12	Furn. & A/C				
13	Electric. wiring				

Time-Effect. Capab. For Resource Type 4		Office Staff	Office Staff	Office Staff	Office Staff
Act. No.	Act. Name	1(spec., draft)	2(spec., draft)	3(spec., draft)	4(couns., sales)
14	Wallboard				
15	Stairway				
16	Painting				
17	Trim and Final Corrections	$(\Phi)_4 = t_{17}^{2,1} (1 + 0.25) \cdot y_{17}^{2,1}$ $(T)_5 = (t_{17}^{2,2} + 0.2) \cdot y_{17}^{2,2}$ $(T)_6 = (t_{17}^{2,4} + 0.4) \cdot y_{17}^{2,4}$ $(T)_7 = (t_{17}^{2,6} + 0.4) \cdot y_{17}^{2,6}$ $(T)_8 = (t_{17}^{3,2} + 0.3) \cdot y_{17}^{3,2}$ $(T)_9 = (t_{17}^{3,3} + 0.2) \cdot y_{17}^{3,3}$ $(T)_{10} = (t_{17}^{3,5} + 0.35) \cdot y_{17}^{3,5}$	$(T)_5 = (t_{17}^{2,2} + 0.2) \cdot y_{17}^{2,2}$ $(T)_6 = (t_{17}^{2,3} + 0.3) \cdot y_{17}^{2,3}$ $(T)_7 = (t_{17}^{2,5} + 0.4) \cdot y_{17}^{2,5}$ $(T)_8 = (t_{17}^{3,1} + 1.26) \cdot y_{17}^{3,1}$ $(T)_9 = (t_{17}^{3,2} + 1.2) \cdot y_{17}^{3,2}$ $(T)_{10} = (t_{17}^{3,4} + 1.05) \cdot y_{17}^{3,4}$ $(T)_{11} = (t_{17}^{3,5} + 1.10) \cdot y_{17}^{3,5}$	$(T)_5 = (t_{17}^{2,2} + 0.1) \cdot y_{17}^{2,2}$ $(T)_6 = (t_{17}^{2,4} + 0.25) \cdot y_{17}^{2,4}$ $(T)_7 = (t_{17}^{2,5} + 0.4) \cdot y_{17}^{2,5}$ $(T)_8 = (t_{17}^{2,6} + 0.4) \cdot y_{17}^{2,6}$ $(T)_9 = (t_{17}^{3,3} + 1.2) \cdot y_{17}^{3,3}$ $(T)_{10} = (t_{17}^{3,4} + 1.05) \cdot y_{17}^{3,4}$ $(T)_{11} = (t_{17}^{3,5} + 1.10) \cdot y_{17}^{3,5}$	$(T)_7 = (t_{17}^{2,1} + 2) \cdot y_{17}^{2,1}$ $(T)_8 = (t_{17}^{2,2} + 2) \cdot y_{17}^{2,2}$ $(T)_9 = (t_{17}^{2,3} + 2) \cdot y_{17}^{2,3}$ $(T)_{10} = (t_{17}^{2,4} + 2) \cdot y_{17}^{2,4}$ $(T)_{11} = (t_{17}^{2,5} + 2) \cdot y_{17}^{2,5}$ $(T)_{12} = (t_{17}^{2,6} + 2) \cdot y_{17}^{2,6}$ $(T)_{13} = (t_{17}^{3,1} + 1.5) \cdot y_{17}^{3,1}$ $(T)_{14} = (t_{17}^{3,2} + 1.1) \cdot y_{17}^{3,2}$ $(T)_{15} = (t_{17}^{3,5} + 1.1) \cdot y_{17}^{3,5}$
18	Contract and Admin. Closure	2	3	3	2

PREFERENCES

Table C6. Preferences for Resource Type 1.

		Utility Worker	Utility Worker	Utility Worker	Utility Worker
Act. No.	Act. Name	1(water)	2(water, electric)	3(electric, water)	4(electric, water)
1	Cust. Select.				
2	Write specs				
3	Write contract				
4	Detail Plans				
5	Excavation				
6	Footing/ Foundation				
7	Water service	8	6	7	8
8	Electric. service	5	6	7	6
9	Wood framing				
10	Roofing				
11	Plumb. lines	8	8	7	7
12	Furn. & A/C	5	7	6	7
13	Electr. wiring				
14	Wallboard				
15	Stairway				
16	Painting				
17	Trim and Final Corrections				
18	Contract & Admin. Clos.				

Table C7. Preferences for Resource Type 2.

Act. No.	Act. Name	Contractor 1(excav., mech.)	Contractor 2(excav., mech.)	Contractor 3(excav., paint., mech.)	Contractor 4(excav., electr., paint.)	Contractor 5(electr., excav.)	Contractor 6(paint.,mec h, electr.)
1	Cust. Select.						
2	Write specs						
3	Write contract						
4	Detail Plans						
5	Excavation	8	8	6	6	7	3
6	Footing/ Foundation	8	7	6	6	6	3
7	Water service						
8	Electric. service						
9	Wood framing						
10	Roofing						
11	Plumb. lines	$(\mathcal{P}_1 = 7 \cdot y_{11}^{1,1}$ $(\mathcal{P}_2 = 6 \cdot y_{11}^{1,2}$ $(\mathcal{P}_3 = 5 \cdot y_{11}^{1,3}$ $(\mathcal{P}_4 = 6 \cdot y_{11}^{1,4}$	7	6	$(\mathcal{P}_1 = 7 \cdot y_{11}^{1,1}$ $(\mathcal{P}_2 = 6 \cdot y_{11}^{1,2}$ $(\mathcal{P}_3 = 5 \cdot y_{11}^{1,3}$ $(\mathcal{P}_4 = 6 \cdot y_{11}^{1,4}$	5	
12	Furn. & A/C	6	6	4	4	5	6

Preferences for Resource Type 2		Contractor	Contractor	Contractor	Contractor	Contractor	Contractor
Act. No.	Act. Name	1(excav., mech.)	2(excav., mech.)	3(excav., paint., mech.)	4(excav., electr., paint.)	5(electr., excav.)	6(paint.,mec h, electr.)
13	Electr. wiring	3	4	3	$(\mathcal{P}_5 = 2 \cdot y_{13}^{1,1}$ $(\mathcal{P}_6 = 7 \cdot y_{13}^{1,2}$ $(\mathcal{P}_7 = 8 \cdot y_{13}^{1,3}$ $(\mathcal{P}_8 = 8 \cdot y_{13}^{1,4}$	$(\mathcal{P}_1 = 3 \cdot y_{13}^{1,1}$ $(\mathcal{P}_2 = 8 \cdot y_{13}^{1,2}$ $(\mathcal{P}_3 = 7 \cdot y_{13}^{1,3}$ $(\mathcal{P}_4 = 8 \cdot y_{13}^{1,4}$	$(\mathcal{P}_1 = 4 \cdot y_{13}^{1,1}$ $(\mathcal{P}_2 = 6 \cdot y_{13}^{1,2}$ $(\mathcal{P}_3 = 8 \cdot y_{13}^{1,3}$ $(\mathcal{P}_4 = 7 \cdot y_{13}^{1,4}$
14	Wallboard						
15	Stairway	7	6	3	4	2	8
16	Painting	2	2	9	6	5	8
17	Trim and Final Corrections	8	4	4	1	8	9
18	Contract & Admin. Clos.						

Table C8. Preferences for Resource Type 3.

Act. No.	Act. Name	Carpenter 1(frame, wall)	Carpenter 2 (frame, wall)	Carpenter 3(finish, frame)	Carpenter 4(finish, frame)	Carpenter 5(wall, frame)
1	Cust. Select.					
2	Write specs					
3	Write contract					
4	Detail Plans					
5	Excavation					
6	Footing/ Foundation					
7	Water service					
8	Electric. service					
9	Wood framing	8	8	6	6	7
10	Roofing	7	7	8	8	5
11	Plumb. lines	$(\mathcal{P}_1 = 3 \cdot y_{11}^{2,1})$ $(\mathcal{P}_2 = 2 \cdot y_{11}^{2,2})$ $(\mathcal{P}_3 = 8 \cdot y_{11}^{2,3})$ $(\mathcal{P}_4 = 6 \cdot y_{11}^{2,4})$ $(\mathcal{P}_5 = 4 \cdot y_{11}^{2,5})$ $(\mathcal{P}_6 = 6 \cdot y_{11}^{2,6})$ $(\mathcal{P}_7 = 4 \cdot y_{11}^{1,1})$ $(\mathcal{P}_8 = 5 \cdot y_{11}^{1,2})$ $(\mathcal{P}_9 = 4 \cdot y_{11}^{1,3})$ $(\mathcal{P}_{10} = 4 \cdot y_{11}^{1,4})$	$(\mathcal{P}_1 = 2 \cdot y_{11}^{2,1})$ $(\mathcal{P}_2 = 1 \cdot y_{11}^{2,2})$ $(\mathcal{P}_3 = 9 \cdot y_{11}^{2,3})$ $(\mathcal{P}_4 = 1 \cdot y_{11}^{2,4})$ $(\mathcal{P}_5 = 7 \cdot y_{11}^{2,5})$ $(\mathcal{P}_6 = 4 \cdot y_{11}^{2,6})$ $(\mathcal{P}_7 = 4 \cdot y_{11}^{1,1})$ $(\mathcal{P}_8 = 5 \cdot y_{11}^{1,2})$ $(\mathcal{P}_9 = 4 \cdot y_{11}^{1,3})$ $(\mathcal{P}_{10} = 4 \cdot y_{11}^{1,4})$	$(\mathcal{P}_1 = 3 \cdot y_{11}^{2,1})$ $(\mathcal{P}_2 = 2 \cdot y_{11}^{2,2})$ $(\mathcal{P}_3 = 8 \cdot y_{11}^{2,3})$ $(\mathcal{P}_4 = 6 \cdot y_{11}^{2,4})$ $(\mathcal{P}_5 = 4 \cdot y_{11}^{2,5})$ $(\mathcal{P}_6 = 6 \cdot y_{11}^{2,6})$ $(\mathcal{P}_7 = 4 \cdot y_{11}^{1,1})$ $(\mathcal{P}_8 = 5 \cdot y_{11}^{1,2})$ $(\mathcal{P}_9 = 4 \cdot y_{11}^{1,3})$ $(\mathcal{P}_{10} = 4 \cdot y_{11}^{1,4})$	$(\mathcal{P}_1 = 4 \cdot y_{11}^{2,1})$ $(\mathcal{P}_2 = 4 \cdot y_{11}^{2,2})$ $(\mathcal{P}_3 = 7 \cdot y_{11}^{2,3})$ $(\mathcal{P}_4 = 2 \cdot y_{11}^{2,4})$ $(\mathcal{P}_5 = 4 \cdot y_{11}^{2,5})$ $(\mathcal{P}_6 = 3 \cdot y_{11}^{2,6})$ $(\mathcal{P}_7 = 4 \cdot y_{11}^{1,1})$ $(\mathcal{P}_8 = 5 \cdot y_{11}^{1,2})$ $(\mathcal{P}_9 = 4 \cdot y_{11}^{1,3})$ $(\mathcal{P}_{10} = 4 \cdot y_{11}^{1,4})$	$(\mathcal{P}_1 = 7 \cdot y_{11}^{2,1})$ $(\mathcal{P}_2 = 7 \cdot y_{11}^{2,2})$ $(\mathcal{P}_3 = 7 \cdot y_{11}^{2,3})$ $(\mathcal{P}_4 = 7 \cdot y_{11}^{2,4})$ $(\mathcal{P}_5 = 7 \cdot y_{11}^{2,5})$ $(\mathcal{P}_6 = 7 \cdot y_{11}^{2,6})$ $(\mathcal{P}_7 = 4 \cdot y_{11}^{1,1})$ $(\mathcal{P}_8 = 5 \cdot y_{11}^{1,2})$ $(\mathcal{P}_9 = 4 \cdot y_{11}^{1,3})$ $(\mathcal{P}_{10} = 4 \cdot y_{11}^{1,4})$

Preferences for Resource Type 3		Carpenter	Carpenter	Carpenter	Carpenter	Carpenter
Act. No.	Act. Name	1(frame, wall)	2 (frame, wall)	3(finish, frame)	4(finish, frame)	5(wall, frame)
12	Furn. & A/C					
12	Furn. & A/C					
14	Wallboard	5	5	5	6	8
15	Stairway	$\mathcal{J}_{11} = 6 \cdot y_{15}^{2,1}$ $(\mathcal{J}_{12} = 8 \cdot y_{15}^{2,2})$ $(\mathcal{J}_{13} = 2 \cdot y_{15}^{2,3})$ $(\mathcal{J}_{14} = 2 \cdot y_{15}^{2,4})$ $(\mathcal{J}_{15} = 4 \cdot y_{15}^{2,5})$ $(\mathcal{J}_{16} = 6 \cdot y_{15}^{2,6})$	$\mathcal{J}_{11} = 6 \cdot y_{15}^{2,1}$ $(\mathcal{J}_{12} = 8 \cdot y_{15}^{2,2})$ $(\mathcal{J}_{13} = 2 \cdot y_{15}^{2,3})$ $(\mathcal{J}_{14} = 2 \cdot y_{15}^{2,4})$ $(\mathcal{J}_{15} = 4 \cdot y_{15}^{2,5})$ $(\mathcal{J}_{16} = 6 \cdot y_{15}^{2,6})$	$\mathcal{J}_{11} = 1 \cdot y_{15}^{2,1}$ $(\mathcal{J}_{12} = 1 \cdot y_{15}^{2,2})$ $(\mathcal{J}_{13} = 8 \cdot y_{15}^{2,3})$ $(\mathcal{J}_{14} = 10 \cdot y_{15}^{2,4})$ $(\mathcal{J}_{15} = 4 \cdot y_{15}^{2,5})$ $(\mathcal{J}_{16} = 6 \cdot y_{15}^{2,6})$	$\mathcal{J}_8 = 12 \cdot y_{15}^{2,1}$ $(\mathcal{J}_9 = 8 \cdot y_{15}^{2,2})$ $(\mathcal{J}_{10} = 2 \cdot y_{15}^{2,3})$ $(\mathcal{J}_{11} = 2 \cdot y_{15}^{2,4})$ $(\mathcal{J}_{12} = 2 \cdot y_{15}^{2,5})$ $(\mathcal{J}_{13} = 1 \cdot y_{15}^{2,6})$	$\mathcal{J}_8 = 6 \cdot y_{15}^{2,1}$ $(\mathcal{J}_9 = 8 \cdot y_{15}^{2,2})$ $(\mathcal{J}_{10} = 2 \cdot y_{15}^{2,3})$ $(\mathcal{J}_{11} = 2 \cdot y_{15}^{2,4})$ $(\mathcal{J}_{12} = 4 \cdot y_{15}^{2,5})$ $(\mathcal{J}_{13} = 6 \cdot y_{15}^{2,6})$
16	Painting					
17	Trim and Final Corrections	$(\mathcal{J}_{17} = 6 \cdot y_{17}^{2,3})$ $(\mathcal{J}_{18} = 8 \cdot y_{17}^{2,4})$ $(\mathcal{J}_{19} = 6 \cdot y_{17}^{2,6})$	$(\mathcal{J}_{17} = 6 \cdot y_{17}^{2,3})$ $(\mathcal{J}_{18} = 8 \cdot y_{17}^{2,4})$ $(\mathcal{J}_{19} = 6 \cdot y_{17}^{2,6})$	$(\mathcal{J}_{17} = 6 \cdot y_{17}^{2,3})$ $(\mathcal{J}_{18} = 8 \cdot y_{17}^{2,4})$ $(\mathcal{J}_{19} = 6 \cdot y_{17}^{2,6})$	$(\mathcal{J}_{17} = 6 \cdot y_{17}^{2,3})$ $(\mathcal{J}_{18} = 8 \cdot y_{17}^{2,4})$ $(\mathcal{J}_{19} = 6 \cdot y_{17}^{2,6})$	$(\mathcal{J}_{17} = 6 \cdot y_{17}^{2,3})$ $(\mathcal{J}_{18} = 8 \cdot y_{17}^{2,4})$ $(\mathcal{J}_{19} = 6 \cdot y_{17}^{2,6})$
18	Contract and Admin. Closure					

Table C9. Preferences for Resource Type 4.

Act. No.	Act. Name	Office Staff 1(spec.,draft)	Office Staff 2(spec.,draft)	Office Staff 3(spec.,draft)	Office Staff 4(couns.,sales)
1	Cust. Select.	4	4	4	8
2	Write specs	7	7	6	3
3	Write contract	5	4	4	6
4	Detail Plans	7	6	7	3
5	Excavation	5	4	4	4
6	Footing/ Foundation	$(\mathcal{P}_1 = 5 \cdot y_6^{2,1})$ $(\mathcal{P}_2 = 4 \cdot y_6^{2,2})$ $(\mathcal{P}_3 = 2 \cdot y_6^{2,4})$ $(\mathcal{P}_4 = 4 \cdot y_6^{2,6})$	$(\mathcal{P}_1 = 7 \cdot y_6^{2,1})$ $(\mathcal{P}_2 = 3 \cdot y_6^{2,2})$ $(\mathcal{P}_3 = 5 \cdot y_6^{2,4})$	$(\mathcal{P}_1 = 4 \cdot y_6^{2,1})$ $(\mathcal{P}_2 = 4 \cdot y_6^{2,2})$ $(\mathcal{P}_3 = 4 \cdot y_6^{2,4})$ $(\mathcal{P}_4 = 4 \cdot y_6^{2,6})$	$(\mathcal{P}_1 = 4 \cdot y_6^{2,1})$ $(\mathcal{P}_2 = 2 \cdot y_6^{2,2})$ $(\mathcal{P}_3 = 2 \cdot y_6^{2,4})$ $(\mathcal{P}_4 = 2 \cdot y_6^{2,6})$
7	Water service	5	4	5	6
8	Electric. service	5	4	5	6
9	Wood framing	8	7	7	8
10	Roofing	8	7	7	8
11	Plumb. lines				
12	Furn. & A/C				
13	Electric. wiring				
14	Wallboard				
15	Stairway				
16	Painting				

Preferences for Resource Type 4		Office Staff	Office Staff	Office Staff	Office Staff
Act. No.	Act. Name	1(spec.,draft)	2(spec.,draft)	3(spec.,draft)	4(couns.,sales)
17	Trim and Final Corrections	$(\mathcal{P})_5 = 5 \cdot y_{18}^{2,1}$ $(\mathcal{P})_6 = 4 \cdot y_{18}^{2,2}$ $(\mathcal{P})_7 = 2 \cdot y_{18}^{2,4}$ $(\mathcal{P})_8 = 4 \cdot y_{18}^{2,6}$ $(\mathcal{P})_9 = 8 \cdot y_{18}^{3,3}$ $(\mathcal{P})_{10} = 6 \cdot y_{18}^{3,4}$	$(\mathcal{P})_4 = 5 \cdot y_{18}^{2,1}$ $(\mathcal{P})_5 = 2 \cdot y_{18}^{2,2}$ $(\mathcal{P})_6 = 3 \cdot y_{18}^{2,4}$ $(\mathcal{P})_7 = 9 \cdot y_{18}^{3,3}$ $(\mathcal{P})_8 = 6 \cdot y_{18}^{3,4}$	$(\mathcal{P})_5 = 8 \cdot y_{18}^{2,1}$ $(\mathcal{P})_6 = 4 \cdot y_{18}^{2,2}$ $(\mathcal{P})_7 = 6 \cdot y_{18}^{2,4}$ $(\mathcal{P})_8 = 4 \cdot y_{18}^{2,6}$ $(\mathcal{P})_9 = 6 \cdot y_{18}^{3,3}$ $(\mathcal{P})_{10} = 6 \cdot y_{18}^{3,4}$	$(\mathcal{P})_5 = 2 \cdot y_{18}^{2,1}$ $(\mathcal{P})_6 = 1 \cdot y_{18}^{2,2}$ $(\mathcal{P})_7 = 2 \cdot y_{18}^{2,4}$ $(\mathcal{P})_8 = 2 \cdot y_{18}^{2,6}$ $(\mathcal{P})_9 = 5 \cdot y_{18}^{3,3}$ $(\mathcal{P})_{10} = 5 \cdot y_{18}^{3,4}$
18	Contract and Admin. Closure	5	7	7	9

Table C10. Costs for Resource Type 1.

Costs for Resource Type 1		Utility Worker	Utility Worker	Utility Worker	Utility Worker
Act. No.	Act. Name	1(water)	2(water, electric)	3(electric, water)	4(electric, water)
1	Cust..Select.				
2	Write specs				
3	Write contract				
4	Detail Plans				
5	Excavation				
6	Footing/ Foundation				
7	Water service	4	5	5	5
8	Electric. service	6	5	6	5
9	Wood framing				
10	Roofing				
11	Plumb. lines	4	5	5	5
12	Furn. & A/C				
13	Electr. wiring	6	5	6	5
14	Wallboard				
15	Stairway				
16	Painting				
17	Trim and Final Corrections				
18	Contract & Admin. Clos.				

Table C11. Costs for Resource Type 2.

		Contractor	Contractor	Contractor	Contractor	Contractor	Contract or
Act. No.	Act. Name	1(excav., mech.)	2(excav., mech.)	3(excav., paint., mech.)	4(excav., electr., paint.)	5(electr., excav.)	6(paint., mech, electr.)
1	Cust. Select.						
2	Write specs						
3	Write contract						
4	Detail Plans						
5	Excavation	4	4	5	5	6	8
6	Footing/ Foundation	4	4	6	5	6	8
7	Water service						
8	Electric. service						
9	Wood framing						
10	Roofing						
11	Plumb. lines	$(\theta_1 = 11 \cdot y_{11}^{1,1})$ $(\theta_2 = 10 \cdot y_{11}^{1,2})$ $(\theta_3 = 12 \cdot y_{11}^{1,3})$ $(\theta_4 = 14 \cdot y_{11}^{1,4})$	$(\theta_1 = 9 \cdot y_{11}^{1,1})$ $(\theta_2 = 10 \cdot y_{11}^{1,2})$ $(\theta_3 = 12 \cdot y_{11}^{1,3})$ $(\theta_4 = 14 \cdot y_{11}^{1,4})$	$(\theta_1 = 11 \cdot y_{11}^{1,1})$ $(\theta_2 = 12 \cdot y_{11}^{1,2})$ $(\theta_3 = 14 \cdot y_{11}^{1,3})$ $(\theta_4 = 16 \cdot y_{11}^{1,4})$	$(\theta_1 = 14 \cdot y_{11}^{1,1})$ $(\theta_2 = 14 \cdot y_{11}^{1,2})$ $(\theta_3 = 16 \cdot y_{11}^{1,3})$ $(\theta_4 = 18 \cdot y_{11}^{1,4})$	$(\theta_1 = 13 \cdot y_{11}^{1,1})$ $(\theta_2 = 14 \cdot y_{11}^{1,2})$ $(\theta_3 = 16 \cdot y_{11}^{1,3})$ $(\theta_4 = 18 \cdot y_{11}^{1,4})$	$(\theta_1 = 10 \cdot y_{11}^{1,1})$ $(\theta_2 = 9 \cdot y_{11}^{1,2})$ $(\theta_3 = 11 \cdot y_{11}^{1,3})$ $(\theta_4 = 13 \cdot y_{11}^{1,4})$

Costs for Resource Type 2		Contractor	Contractor	Contractor	Contractor	Contractor	Contract or
Act. No.	Act. Name	1(excav., mech.)	2(excav., mech.)	3(excav., paint., mech.)	4(excav., electr., paint.)	5(electr., excav.)	6(paint., mech, electr.)
12	Furn. & A/C	6	4	4	7	8	8
13	Electr. wiring						
14	Wallboard	7	6	7	4	5	4
15	Stairway	7	6	6	4	5	6
16	Painting	9	7	4	6	7	5
17	Trim and Final Corrections	5	5	4	4	4	4
18	Contract & Admin. Clos.						

Table C12. Costs for Resource Type 3.

		Carpenter	Carpenter	Carpenter	Carpenter	Carpenter
Act. No.	Act. Name	1(frame, wall)	2 (frame, wall)	3(finish, frame)	4(finish, frame)	5(wall, frame)
1	Cust. Select.					
2	Write specs					
3	Write contract					
4	Detail Plans					
5	Excavation					
6	Footing/ Foundation					
7	Water service					
8	Electric. service					
9	Wood framing	7	3	4	6	5
10	Roofing	4	8	6	5	4
11	Plumb. lines	$(Q_1 = 10 \cdot y_{11}^{2,1})$ $(Q_2 = 9 \cdot y_{11}^{2,2})$ $(Q_3 = 11 \cdot y_{11}^{2,3})$ $(Q_4 = 13 \cdot y_{11}^{2,4})$ $(Q_4 = 13 \cdot y_{11}^{2,5})$ $(Q_4 = 13 \cdot y_{11}^{2,6})$	$(Q_1 = 8 \cdot y_{11}^{2,1})$ $(Q_2 = 12 \cdot y_{11}^{2,2})$ $(Q_3 = 7 \cdot y_{11}^{2,3})$ $(Q_4 = 14 \cdot y_{11}^{2,4})$ $(Q_4 = 14 \cdot y_{11}^{2,5})$ $(Q_4 = 14 \cdot y_{11}^{2,6})$	$(Q_1 = 13 \cdot y_{11}^{2,1})$ $(Q_2 = 12 \cdot y_{11}^{2,2})$ $(Q_3 = 14 \cdot y_{11}^{2,3})$ $(Q_4 = 16 \cdot y_{11}^{2,4})$ $(Q_4 = 16 \cdot y_{11}^{2,5})$ $(Q_4 = 16 \cdot y_{11}^{2,6})$	$(Q_1 = 15 \cdot y_{11}^{2,1})$ $(Q_2 = 14 \cdot y_{11}^{2,2})$ $(Q_3 = 16 \cdot y_{11}^{2,3})$ $(Q_4 = 18 \cdot y_{11}^{2,4})$ $(Q_4 = 18 \cdot y_{11}^{2,5})$ $(Q_4 = 18 \cdot y_{11}^{2,6})$	$(Q_1 = 14 \cdot y_{11}^{2,1})$ $(Q_2 = 13 \cdot y_{11}^{2,2})$ $(Q_3 = 15 \cdot y_{11}^{2,3})$ $(Q_4 = 17 \cdot y_{11}^{2,4})$ $(Q_4 = 17 \cdot y_{11}^{2,5})$ $(Q_4 = 17 \cdot y_{11}^{2,6})$

Costs for Resource Type 3		Carpenter	Carpenter	Carpenter	Carpenter	Carpenter
Act. No.	Act. Name	1(frame, wall)	2 (frame, wall)	3(finish, frame)	4(finish, frame)	5(wall, frame)
12	Furn. & A/C					
13	Electr. wiring					
14	Wallboard	5	8	4	4	4
15	Stairway	7	7	7	4	6
16	Painting					
17	Trim and Final Corrections	5	5	5	5	5
18	Contract & Admin. Clos.					

Table C13. Costs for Resource Type 4.

Costs for Resource Type 4		Office Staff	Office Staff	Office Staff	Office Staff
Act. No.	Act. Name	1(spec., draft)	2(spec., draft)	3(spec., draft)	4(couns., sales)
1	Cust. Select.	8	7	8	6
2	Write specs	5	5	5	5
3	Write contract	7	6	6	6
4	Detail Plans	4	6	4	7
5	Excavation	6	6	7	7
6	Footing/Foundation	4	3	3	5
7	Water service	4	3	4	6
8	Electric. service	4	3	4	6
9	Wood framing	4	3	4	6
10	Roofing	4	3	4	6
11	Plumb. lines	4	3	4	6
12	Furn. & A/C	4	3	4	6
13	Electr. wiring	4	3	4	6
14	Wallboard	4	3	4	6
15	Stairway	4	3	4	6
16	Painting	4	3	4	6
17	Trim and Final Corrections	6	7	5	6
18	Contract & Admin. Clos.	6	6	6	6

EXAMPLE PROJECT #1: OUTPUT

As previously described, PROMAP's output is displayed through five different plots: *resource-activity mapping grid*, *total resource utilization bar charts*, *total relative resource cost charts*, and more traditional *activity Gantt chart*, and *resource loading graphs*.

For the same project, the outputs may vary depending on project managers pre-specified input parameters, such as his/her composite objective or utility function, and or intention to only map resources, only centralize their resource loading graphs, or perform both mapping and centralization simultaneously.

For example, consider a scenario where a project manager would be interested in mapping resource units to project activities, but attempting to centralize the loading graph of only resource type one. The mapping strategy would be to assign all resource units to the most adequate activities based on resource time-effective capabilities. In addition to that, the project manager might also want to put emphasis on satisfying project personnel's preferences, but only for the first 30 time units of the project (since the timely project completion becomes crucial at any later time). Finally, since resource type or group one was selected to have its resource loading graph centralized (or balanced), it is likely that this resource type will be of the greatest budgetary consideration. Thus, the manager's mapping strategy might also include cost considerations.

The actual input reflecting the above strategy is shown in Figure C1.

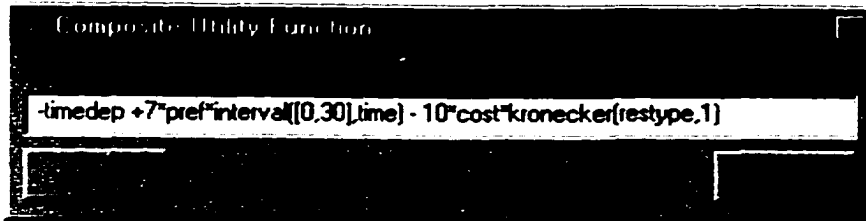


Figure C1. Example of a Project Manager's Mapping Strategy Input.

As illustrated in Figure C1, the additive objective function may also include subjective weighting coefficients for some of its components. In the above example, the preferences component was multiplied by seven, while the cost component was multiplied by a factor of 10.

The *Centralizing Importance Level*, that is the weight \mathcal{W} , was arbitrarily set to 10,000.

The five types of output charts are displayed in Figure C2 through B18. The first displayed are *resource-activity mapping grids*:

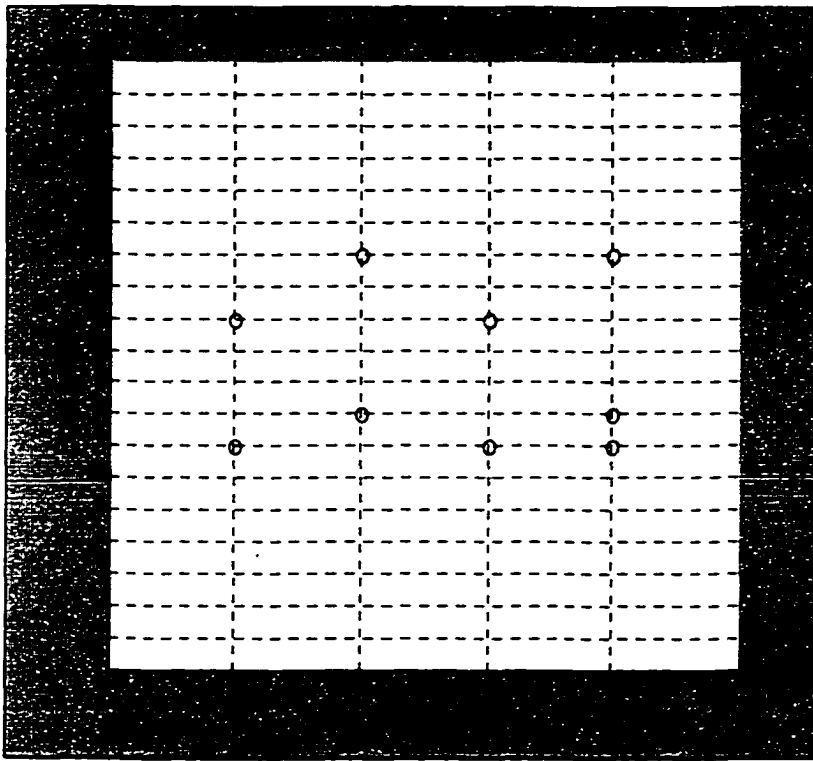


Figure C2. Resource-Activity Mapping Grid for the Units of Resource Type 1.

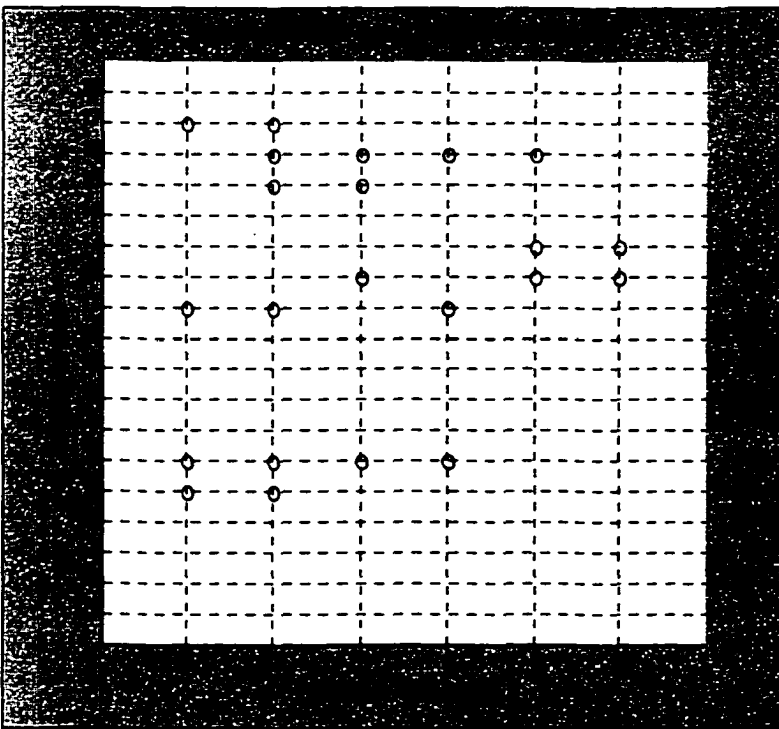


Figure C3. Resource-Activity Mapping Grid for the Units of Resource Type 2.

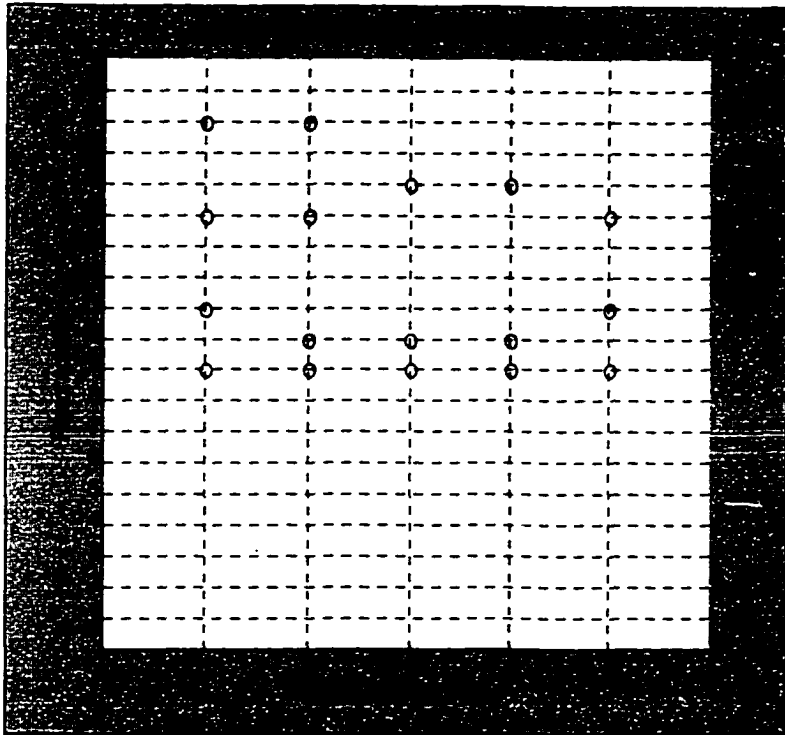


Figure C4. Resource-Activity Mapping Grid for the Units of Resource Type 3.

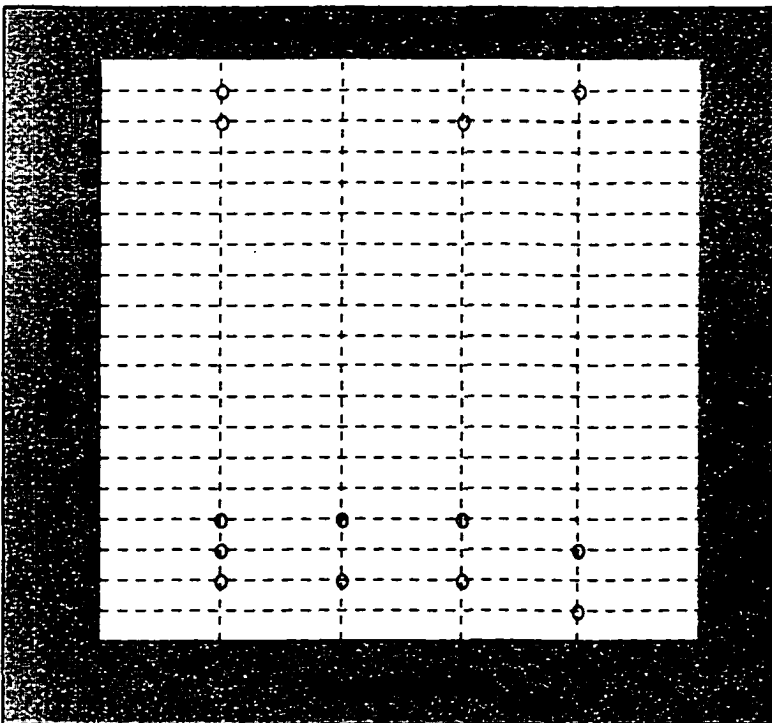


Figure C5. Resource-Activity Mapping Grid for the Units of Resource Type 4.

Next, the project manager may be interested in the total time utilization of each project resource unit as shown in Figure C6-C9. Each bar indicates the total utilization of a specific resource unit as a percentage of the total project duration. The blue colored portion of a bar on the bottom (the darker one, if viewed in black and white mode) is the percentage of time the unit will spend working on its own tasks. The red colored portion of a bar (the lighter area) on top indicates any additional project time that the particular resource unit is engaged in activities by waiting on other units to finish their portions of tasks.

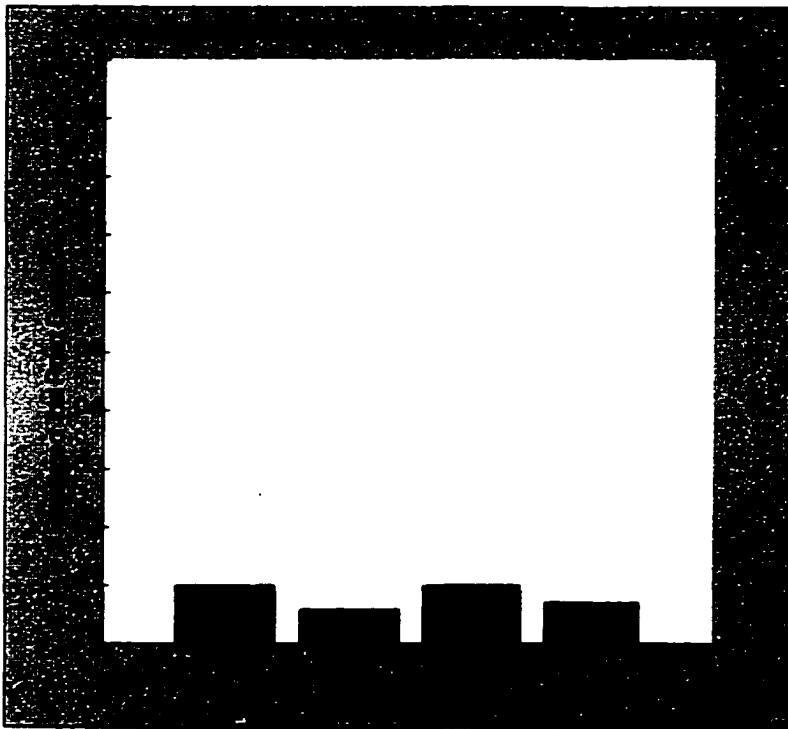


Figure C6. Percentage of Resource Units Utilization for Type 1.

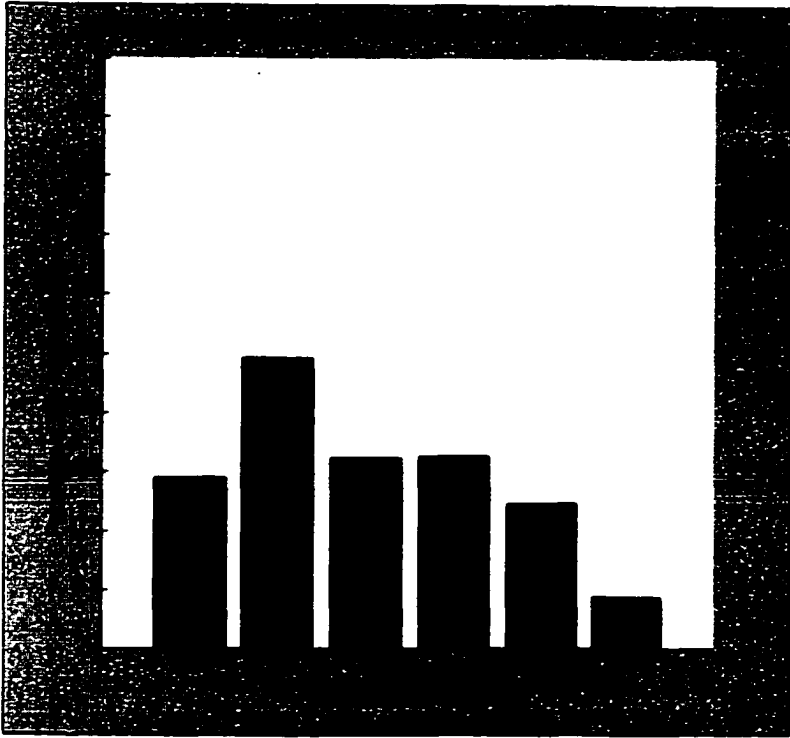


Figure C7. Percentage of Resource Units Utilization for Type 2.

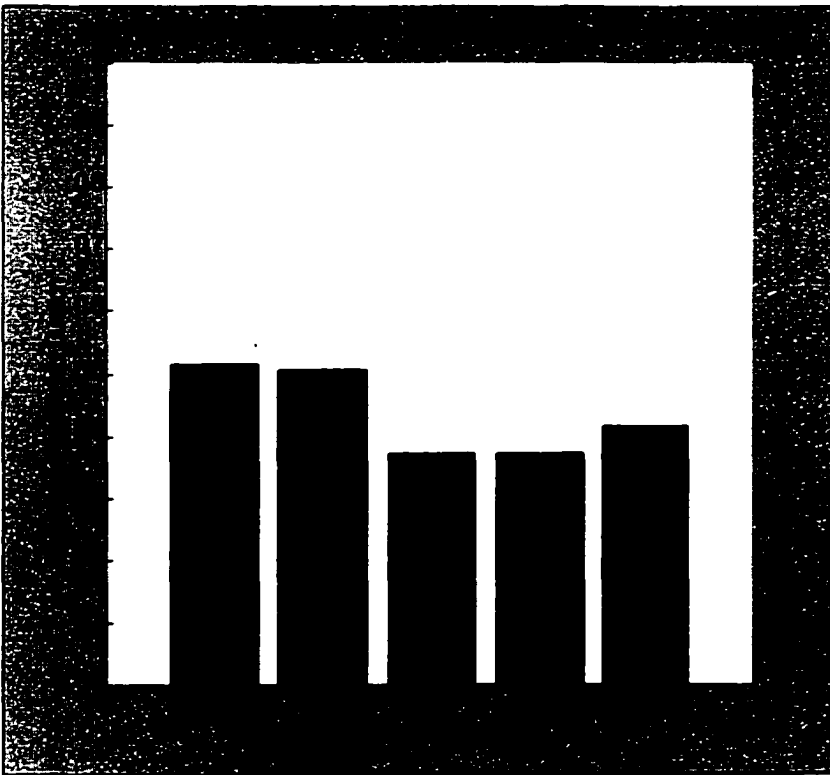


Figure C8. Percentage of Resource Units Utilization for Type 3.

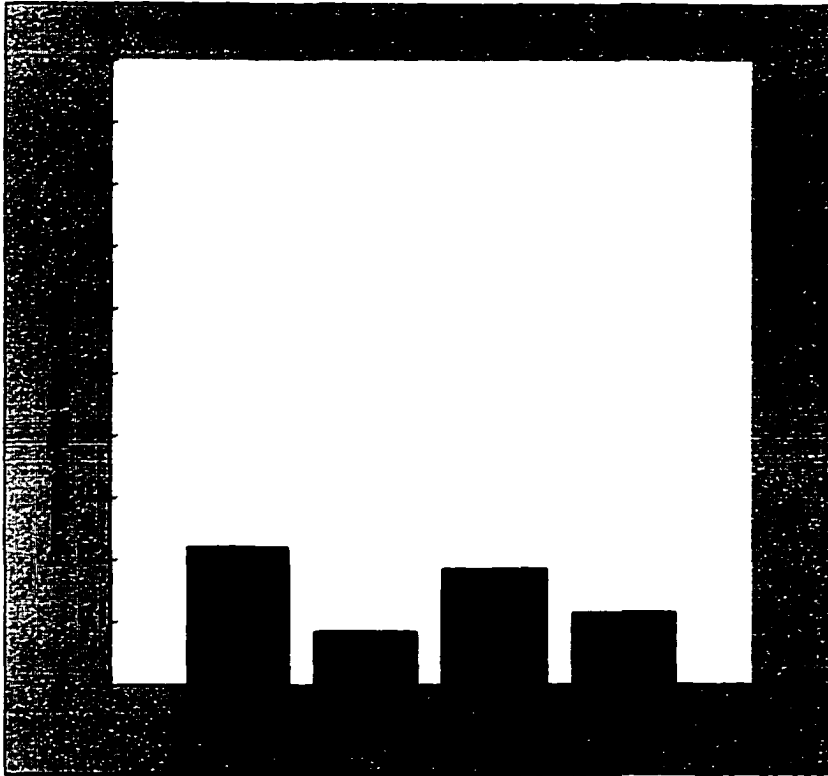


Figure C9. Percentage of Resource Units Utilization for Type 4.

The next set of output charts shown in Figures C10-C13 are the *relative resource costs*.

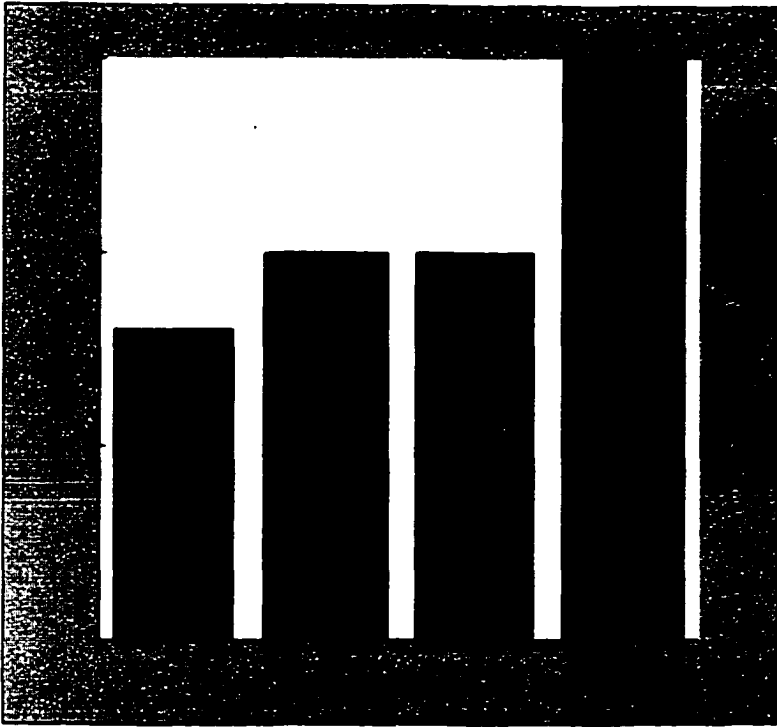


Figure C10. Total Relative Resource Units Costs for Resource Type 1.

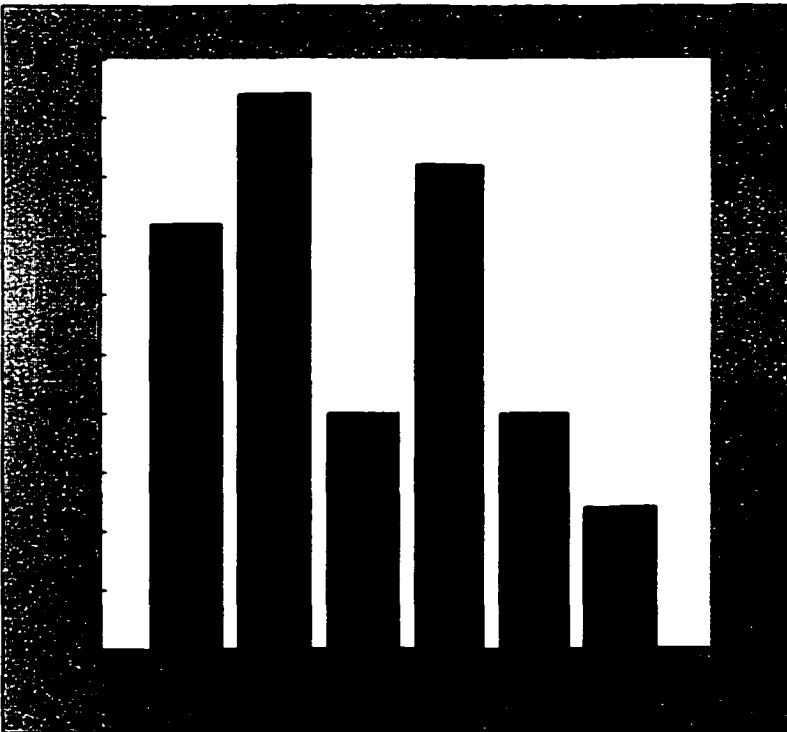


Figure C11. Total Relative Resource Units Costs for Resource Type 2.

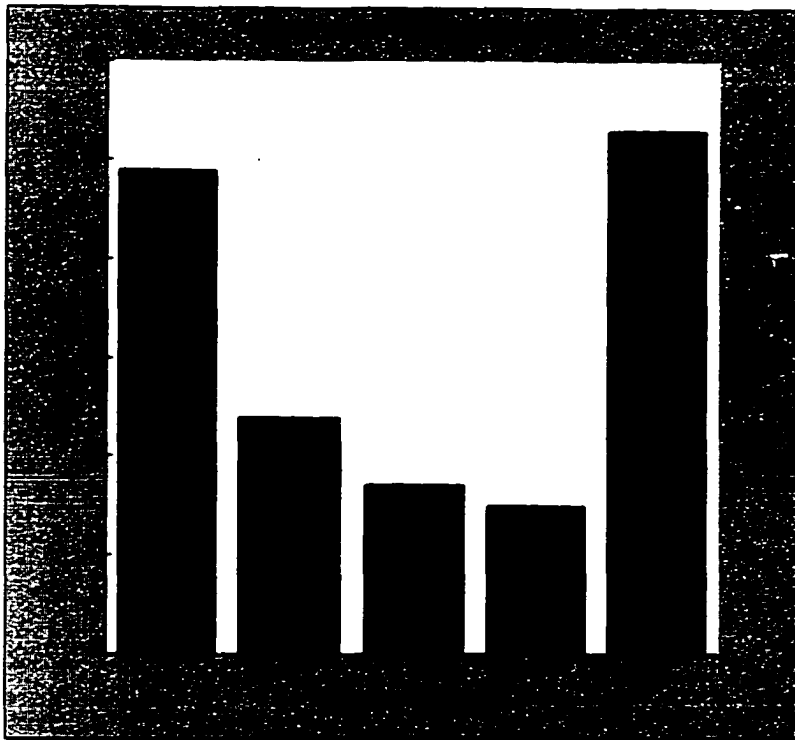


Figure C12. Total Relative Resource Units Costs for Resource Type 3.

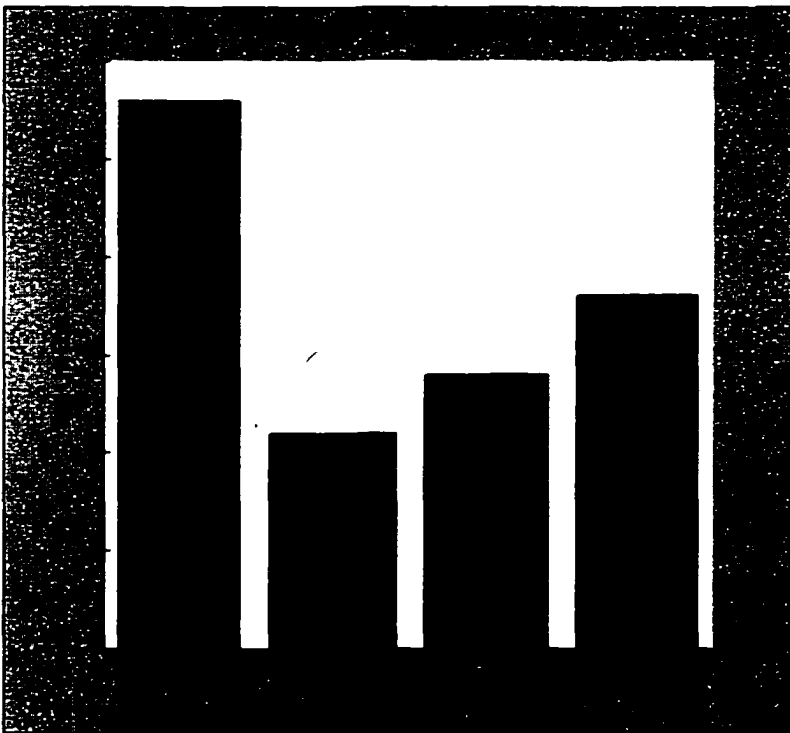


Figure C13. Total Relative Resource Units Costs for Resource Type 4.

The last two types of graphs are traditional ones in project schedules: *resource loading graphs and activity Gantt chart*.

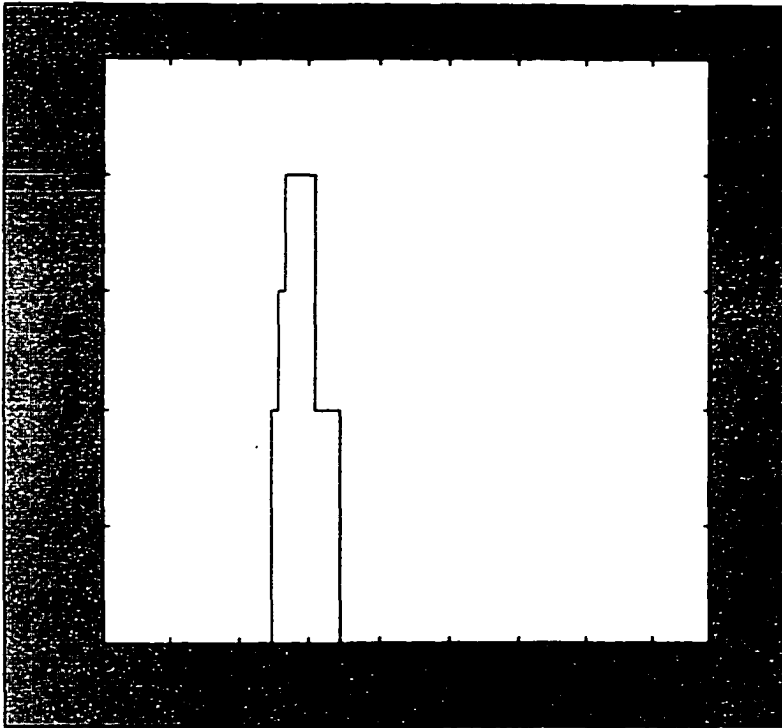


Figure C14. Resource Loading Graph for Resource Type 1.

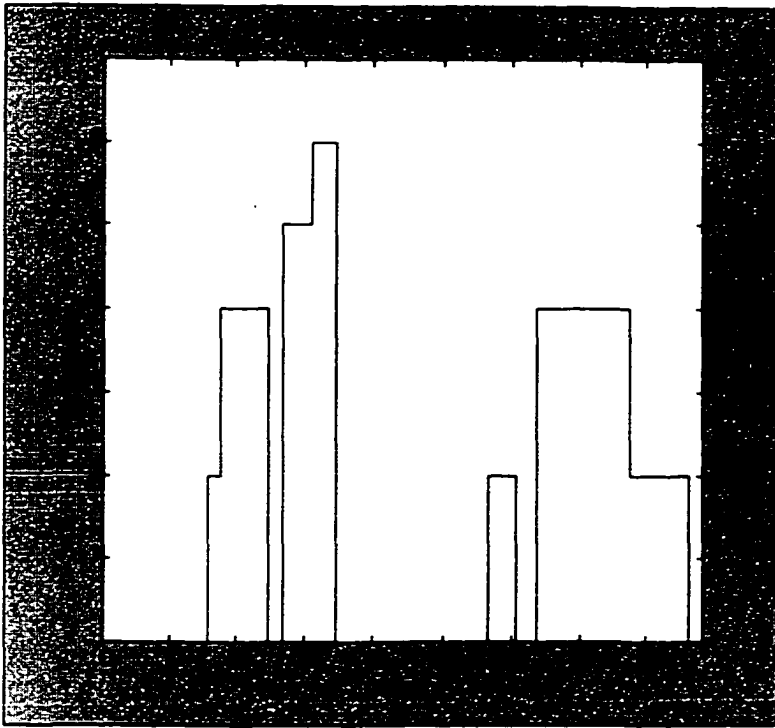


Figure C15. Resource Loading Graph for Resource Type 2.

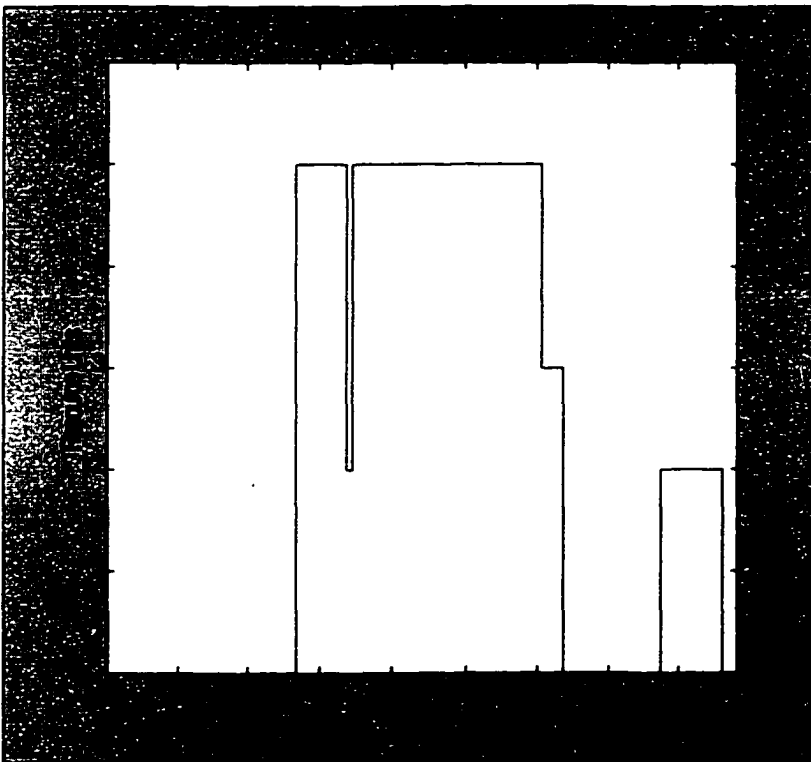


Figure C16. Resource Loading Graph for Resource Type 3.

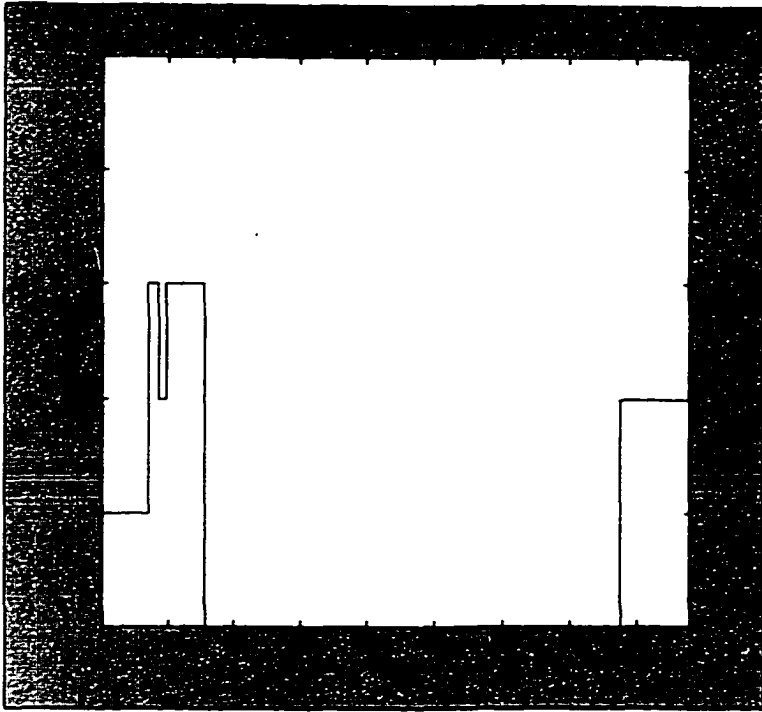


Figure C17. Resource Loading Graph for Resource Type 4.

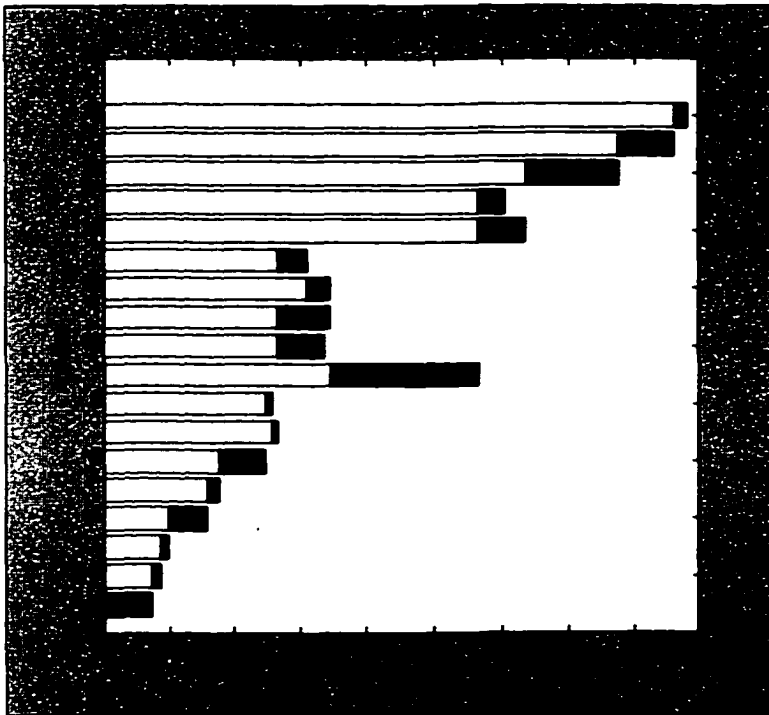


Figure C18. Project Activity Gantt Chart.

Resource centralization and attempting to satisfy resource preferences may enhance personnel's morale and motivation, but could also affect project's duration. Assume that the previous project is to be scheduled and resources mapped, but with a much simplified strategy: without any centralization and considering resource time capabilities only. The resulting output Gantt chart in Figure C19 indicates that, as a result of this relaxation, the project will finish two time units early.

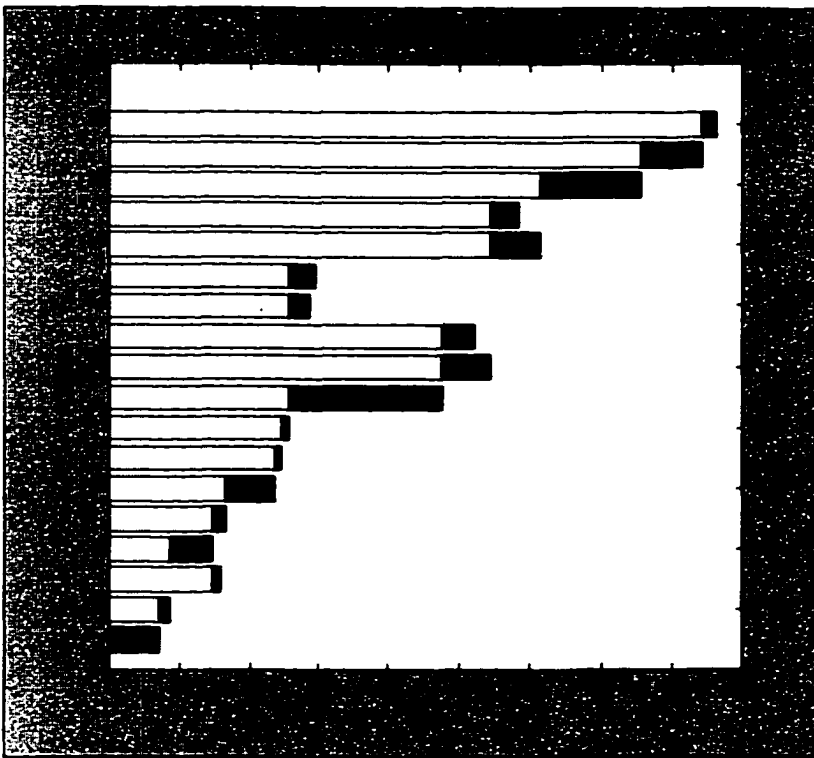


Figure C19. Project Gantt Chart After Simplifying the Scheduling and Mapping Strategies.

Since we have “turned off” the centralization feature, the *resource loading graph* of type 1, now may, and as Figure C20 indicates, will have depression regions.

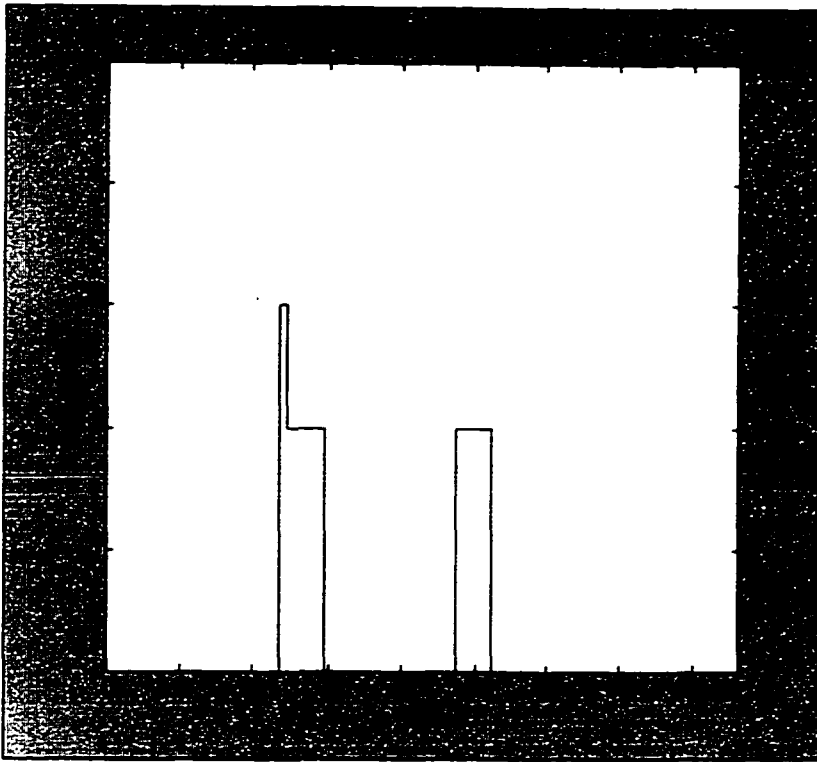


Figure C20. Resource Type 1 loading Graph After Simplifying the Scheduling and Mapping Strategies.

In a more extreme case, where a project manager wishes to satisfy resource preferences with a much greater bias than their capabilities, the project duration and resource-activity mapping may produce significantly different outputs. Consider, for example, the following mapping strategy as shown in Figure C21. The preferences are now 200 times more valued than resource capabilities, and are being considered throughout the entire project schedule (not for just first 30 time units as in Figure C1).

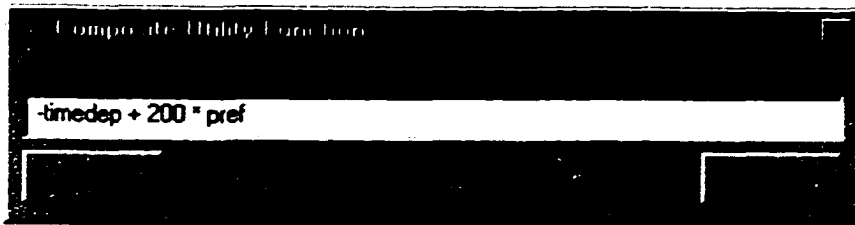


Figure C21. Modified Mapping Strategy.

This strategy of heavily considering preferences will, as indicated in Figure C22, substantially prolong the project schedule.

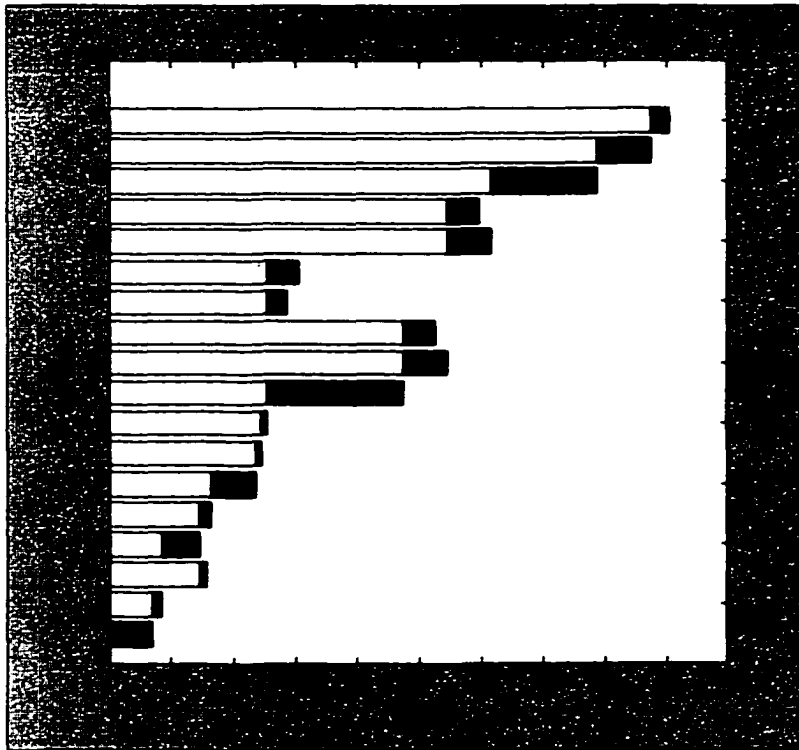


Figure C22. Project Gantt Chart when Resource Preferences Prevail over Resource Capabilities

Notice that the project duration now exceeds 90 time units. Besides the Gantt chart, it should also be expected that resource assignments are also affected and changed by

placing more emphasis on preferences. As shown in Figure C23, the *resource-activity mapping grid* for resource type 1 show different assignments than the ones in Figure C2.

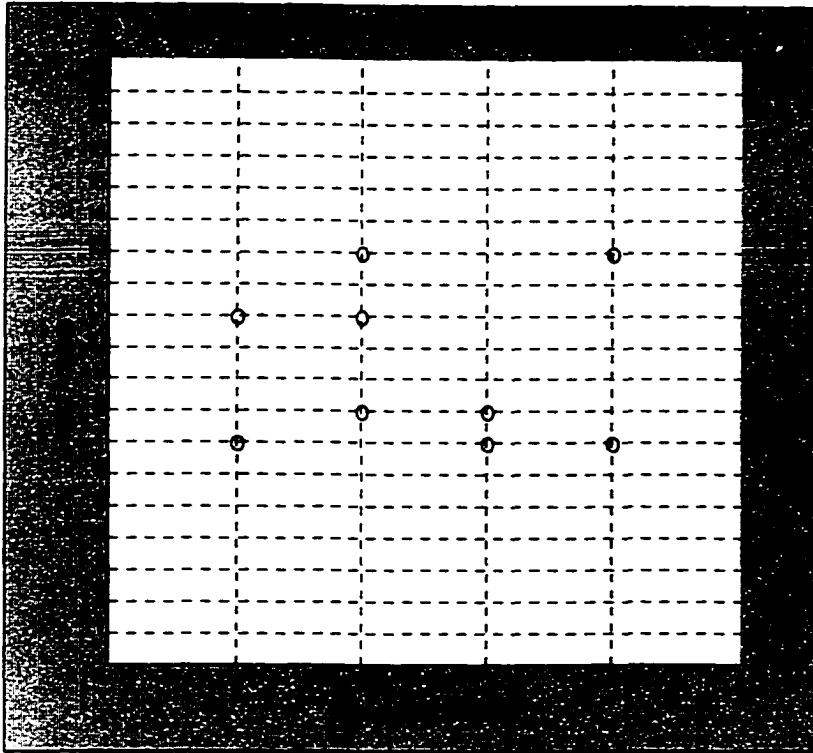


Figure C23. Resource-Activity Mapping Grid for Type 1 when Resource Preferences Prevail over Resource Capabilities.

Finally, an important observation must be made. Table B1 shows that, for example, activity eight requires two units of resource type one. In both Figures B2 and B23, the activity eight is assigned two resource units. However, in Figure C2, those two units are unit two and unit four, while in Figure C23 those units are unit two and three. *In other words, by changing mapping strategies, PROMAP may map different resources to the same activity, however, the number of resource units of each type required by an*

activity must remain unchanged. Similar observations may be made by comparing the mapping grids in Figure C24 and Figure C3.

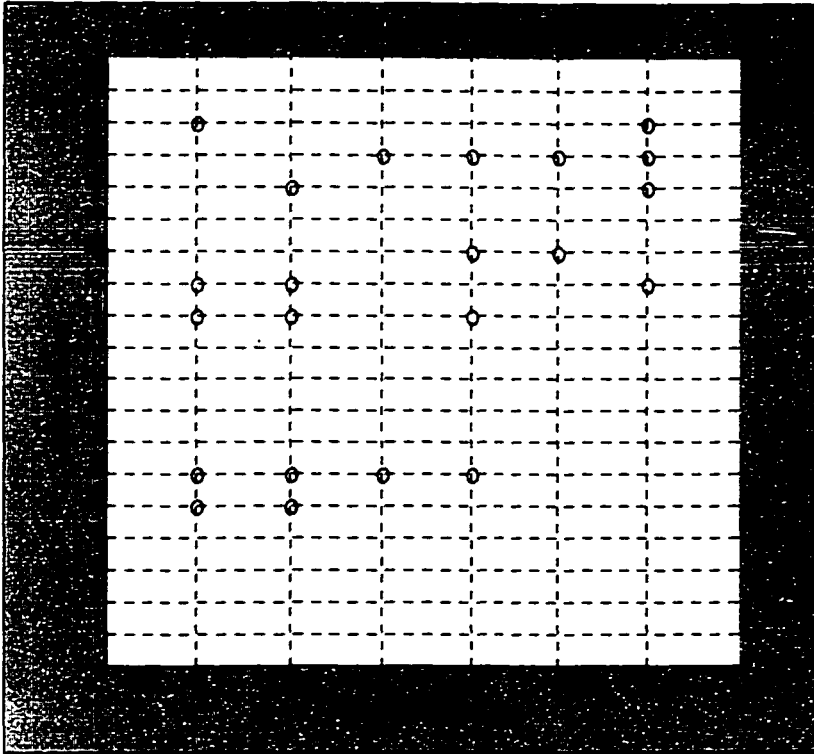


Figure C24. Resource-Activity Mapping Grid for Type 2 when Resource Preferences Prevail over Resource Capabilities.

EXAMPLE PROJECT #2: INPUT DATA

Table C14. Basic Project #2 Data (Partially adopted from Doucette, 1998)

Activity	Activity Name	Predecessors	Res. Group 1	Res. Group 2	Res. Group 3
			<i>Free Lancers</i> <i>NuView Productions</i>	<i>Free Lancers</i> <i>MultiEye Media</i>	<i>Staff</i>
			<i>Max Units: 4</i>	<i>Max Units: 4</i>	<i>Max Units: 4</i>
1	1st meeting w/ customer				2
2	Preliminary outline	1			1
Proposal Stage					
3	Develop proposal	2	2	2	1
4	Presentation to customer	3	1	2	1
5	Develop contract	4			1
6	Create detailed program outline	5		2	1
7	Write scripts	6			1
Development Stage					
8	Create multimedia engine	7		3	
9	Create dummy graphics	7			1
10	Develop dummy interface	8,9		2	1
11	Create preliminary tests	10			1
Production Stage					
12	Develop graphics	11			1
13	Develop multimedia pgrm. W/dummies	11		3	
14	Shoot video	12,13	3		2
15	Capture narration	12,13	1		
16	Offline edit	14,15	1		
17	Final graphics	16			1
Post-Production					
18	Online edit	17	2		1
19	Final assemble	18		2	1
20	Burn gold CD-ROM's	19		2	1
21	Beta test	20			2
Completion					
22	Final revisions	21	2	2	1

Table C15. Time-Effective Capabilities for Resource Group 1.

Activity	Activity Name	NuView Productions			
		Employee 1	Employee 2	Employee 3	Employee 4
1	1st meeting w/ customer				
2	Preliminary outline				
	Proposal Stage				
3	Develop proposal	2	2.5	1.8	2
4	Presentation to customer	1	1	1	1
5	Develop contract				
6	Create detailed program outline				
7	Write scripts				
	Development Stage				
8	Create multimedia engine				
9	Create dummy graphics				
10	Develop dummy interface				
11	Create preliminary tests				
	Production Stage				
12	Develop graphics				
13	Develop multimedia program				
14	Shoot video	2	2	2	2
15	Capture narration	0.8	1	0.7	1.6
16	Offline edit	1	0.9	0.9	1.1
17	Final graphics				
	Post-Production				
18	Online edit	2	2.5	2	2
19	Final assemble				
20	Burn gold CD-ROM's				
21	Beta test				
	Completion				
22	Final revisions	5	5	4	4

Table C16. Time-Effective Capabilities for Resource Group 2.

Activity	Activity Name	Res. Group 2: Multeye Media			
		Employee 1	Employee 2	Employee 3	Employee 4
1	1st meeting w/ customer				
2	Preliminary outline				
	Proposal Stage				
3	Develop proposal	$(T)_1 = (t_3^{1,1} + 0.5) \cdot y_3^{1,1}$ $(T)_2 = (t_3^{1,2} + 0.7) \cdot y_3^{1,2}$ $(T)_3 = (t_3^{1,3} + 0.6) \cdot y_3^{1,3}$ $(T)_4 = (t_3^{1,4} + 0.4) \cdot y_3^{1,4}$	$(T)_1 = (t_3^{1,1} + 0.5) \cdot y_3^{1,1}$ $(T)_2 = (t_3^{1,2} + 0.4) \cdot y_3^{1,2}$ $(T)_3 = (t_3^{1,3} + 0.4) \cdot y_3^{1,3}$ $(T)_4 = (t_3^{1,4} + 0.4) \cdot y_3^{1,4}$	$(T)_1 = (t_3^{1,1} + 0.6) \cdot y_3^{1,1}$ $(T)_2 = (t_3^{1,2} + 0.7) \cdot y_3^{1,2}$ $(T)_3 = (t_3^{1,3} + 0.7) \cdot y_3^{1,3}$ $(T)_4 = (t_3^{1,4} + 0.4) \cdot y_3^{1,4}$	$(T)_1 = (t_3^{1,1} + 0.5) \cdot y_3^{1,1}$ $(T)_2 = (t_3^{1,2} + 0.7) \cdot y_3^{1,2}$ $(T)_3 = (t_3^{1,3} + 0.6) \cdot y_3^{1,3}$ $(T)_4 = (t_3^{1,4} + 0.4) \cdot y_3^{1,4}$
4	Presentation to customer	1	1	1	1
5	Develop contract				
6	Create detailed program outline	6	7	7	5
7	Write scripts				
	Development Stage				
8	Create multimedia engine	7	8	5	8
9	Create dummy graphics				
10	Develop dummy interface	3	5	3	3
11	Create preliminary tests				

		Time-Effective Capabilities			
		Res. Group 2:	Multeye Media		
Activity	Activity Name	Employee1	Employee 2	Employee 3	Employee 4
	Production Stage				
12	Develop graphics				
13	Develop multimedia program	12	15	14	14
14	Shoot video				
15	Capture narration				
16	Offline edit				
17	Final graphics				
	Post-Production				
18	Online edit				
19	Final assemble	5	7	5	8
20	Burn gold CD-ROM's	1	1	1	1
21	Beta test				
	Completion				
22	Final revisions	$(T)_5 = t_{22}^{1,1} \cdot (1.15) \cdot y_{22}^{1,1}$ $(T)_6 = t_{22}^{1,2} \cdot (1.10) \cdot y_{22}^{1,2}$ $(T)_7 = t_{22}^{1,3} \cdot (1.15) \cdot y_{22}^{1,3}$ $(T)_8 = t_{22}^{1,4} \cdot (1.20) \cdot y_{22}^{1,4}$	$(T)_5 = t_{22}^{1,1} \cdot (1.15) \cdot y_{22}^{1,1}$ $(T)_6 = t_{22}^{1,2} \cdot (1.15) \cdot y_{22}^{1,2}$ $(T)_7 = t_{22}^{1,3} \cdot (1.15) \cdot y_{22}^{1,3}$ $(T)_8 = t_{22}^{1,4} \cdot (1.25) \cdot y_{22}^{1,4}$	4 $(T)_5 = t_{22}^{1,1} \cdot (1.15) \cdot y_{22}^{1,1}$ $(T)_6 = t_{22}^{1,2} \cdot (1.15) \cdot y_{22}^{1,2}$	5 $(T)_5 = t_{22}^{1,1} \cdot (1.15) \cdot y_{22}^{1,1}$ $(T)_6 = t_{22}^{1,2} \cdot (1.15) \cdot y_{22}^{1,2}$

Table C17. Time-Effective Capabilities for Resource Group 3.

Activity	Activity Name	Staff			
		Larry	Gloria	Bud	Susan
1	1st meeting w/ customer	0.5	0.5	0.5	0.5
2	Preliminary outline	2	4	5	4
	Proposal Stage				
3	Develop proposal	$(T)_1 = (t_3^{1,1} + 1.5) \cdot y_3^{1,1}$ $(T)_2 = (t_3^{2,1} + 1.05) \cdot y_3^{2,1}$ $(T)_3 = (t_3^{2,2} + 1.2) \cdot y_3^{2,2}$ $(T)_4 = (t_3^{2,3} + 1.2) \cdot y_3^{2,3}$ $(T)_5 = (t_3^{2,4} + 0.8) \cdot y_3^{2,4}$	$(T)_1 = (t_3^{1,1} + 1.5) \cdot y_3^{1,1}$ $(T)_2 = (t_3^{2,1} + 1) \cdot y_3^{2,1}$ $(T)_3 = (t_3^{2,2} + 1.2) \cdot y_3^{2,2}$ $(T)_4 = (t_3^{2,3} + 1.2) \cdot y_3^{2,3}$ $(T)_5 = (t_3^{2,4} + 0.8) \cdot y_3^{2,4}$	4	$(T)_1 = (t_3^{2,1} + 1) \cdot y_3^{2,1}$ $(T)_2 = (t_3^{2,2} + 1.2) \cdot y_3^{2,2}$ 4.5
4	Presentation to customer	1	1	1	1
5	Develop contract	0.8	1.5	1	1
6	Create detailed program outline	$(T)_6 = t_6^{2,1} \cdot (1.25) \cdot y_6^{2,1}$ $(T)_7 = t_6^{2,2} \cdot (1.25) \cdot y_6^{2,2}$ $(T)_8 = t_6^{2,3} \cdot (1.25) \cdot y_6^{2,3}$ $(T)_9 = t_6^{2,4} \cdot (1.25) \cdot y_6^{2,4}$	$(T)_6 = t_6^{2,1} \cdot (1.20) \cdot y_6^{2,1}$ $(T)_7 = t_6^{2,2} \cdot (1.20) \cdot y_6^{2,2}$ $(T)_8 = t_6^{2,3} \cdot (1.25) \cdot y_6^{2,3}$ $(T)_9 = t_6^{2,4} \cdot (1.25) \cdot y_6^{2,4}$	$(T)_1 = t_6^{2,1} \cdot (1.25) \cdot y_6^{2,1}$ $(T)_2 = t_6^{2,2} \cdot (1.20) \cdot y_6^{2,2}$ $(T)_3 = t_6^{2,3} \cdot (1.15) \cdot y_6^{2,3}$ $(T)_4 = t_6^{2,4} \cdot (1.25) \cdot y_6^{2,4}$	$(T)_3 = t_6^{2,1} \cdot (1.25) \cdot y_6^{2,1}$ $(T)_4 = t_6^{2,2} \cdot (1.10) \cdot y_6^{2,2}$ $(T)_5 = t_6^{2,3} \cdot (1.10) \cdot y_6^{2,3}$ $(T)_6 = t_6^{2,4} \cdot (1.10) \cdot y_6^{2,4}$
7	Write scripts	8	7	4	5
	Development Stage				
8	Create multimedia engine				
9	Create dummy graphics	3	5	5	5

Activity	Activity Name	Time-Effective Capabilities			
		Res. Group 3:	Staff		
		Larry	Gloria	Bud	Susan
10	Develop dummy interface	$(T)_{10} = (t_3^{2,1} + 2) \cdot y_3^{2,1}$ $(T)_{11} = (t_3^{2,2} + 1.5) \cdot y_3^{2,2}$ $(T)_{12} = (t_3^{2,3} + 1) \cdot y_3^{2,3}$ $(T)_{13} = (t_3^{2,4} + 1.5) \cdot y_3^{2,4}$	$(T)_{10} = (t_3^{2,1} + 2) \cdot y_3^{2,1}$ $(T)_{11} = (t_3^{2,2} + 1.5) \cdot y_3^{2,2}$ $(T)_{12} = (t_3^{2,3} + 1) \cdot y_3^{2,3}$ $(T)_{13} = (t_3^{2,4} + 1.5) \cdot y_3^{2,4}$	$(T)_5 = (t_3^{2,1} + 2) \cdot y_3^{2,1}$ $(T)_6 = (t_3^{2,2} + 1.5) \cdot y_3^{2,2}$ $(T)_7 = (t_3^{2,3} + 1) \cdot y_3^{2,3}$ $(T)_8 = (t_3^{2,4} + 1.5) \cdot y_3^{2,4}$	$(T)_7 = (t_3^{2,1} + 2) \cdot y_3^{2,1}$ $(T)_8 = (t_3^{2,2} + 1.5) \cdot y_3^{2,2}$ $(T)_9 = (t_3^{2,3} + 1) \cdot y_3^{2,3}$ $(T)_{10} = (t_3^{2,4} + 1.5) \cdot y_3^{2,4}$
11	Create preliminary tests	2	2.5	2	2
	Production Stage				
12	Develop graphics	16	13	13	14
13	Develop multimedia program				
14	Shoot video	1	1	1	1
15	Capture narration				
16	Offline edit				
17	Final graphics	7	6	6	6
	Post-Production				
18	Online edit	$(T)_{14} = t_{22}^{1,1} \cdot (1.30) \cdot y_{22}^{1,1}$ $(T)_{15} = t_{22}^{1,2} \cdot (1.20) \cdot y_{22}^{1,2}$ $(T)_{16} = t_{22}^{1,3} \cdot (1.20) \cdot y_{22}^{1,3}$ $(T)_{17} = t_{22}^{1,4} \cdot (1.25) \cdot y_{22}^{1,4}$	$(T)_{14} = t_{22}^{1,1} \cdot (1.30) \cdot y_{22}^{1,1}$ $(T)_{15} = t_{22}^{1,2} \cdot (1.20) \cdot y_{22}^{1,2}$ $(T)_{16} = t_{22}^{1,3} \cdot (1.20) \cdot y_{22}^{1,3}$ $(T)_{17} = t_{22}^{1,4} \cdot (1.25) \cdot y_{22}^{1,4}$	$(T)_9 = t_{22}^{1,1} \cdot (1.30) \cdot y_{22}^{1,1}$ $(T)_{10} = t_{22}^{1,2} \cdot (1.20) \cdot y_{22}^{1,2}$ $(T)_{11} = t_{22}^{1,3} \cdot (1.20) \cdot y_{22}^{1,3}$ $(T)_{12} = t_{22}^{1,4} \cdot (1.25) \cdot y_{22}^{1,4}$	$(T)_{11} = t_{22}^{1,1} \cdot (1.30) \cdot y_{22}^{1,1}$ $(T)_{12} = t_{22}^{1,2} \cdot (1.20) \cdot y_{22}^{1,2}$ $(T)_{13} = t_{22}^{1,3} \cdot (1.20) \cdot y_{22}^{1,3}$ $(T)_{14} = t_{22}^{1,4} \cdot (1.25) \cdot y_{22}^{1,4}$
19	Final assemble	5	4	3	4
20	Burn gold CD-ROM's	1	1	1	1
21	Beta test	12	12	15	15
	Completion				
22	Final revisions	7	5	5	7

Table C18. Preferences for Resource Group 1.

Activity	Activity Name	NuView Productions			
		Employee1	Employee 2	Employee 3	Employee 4
1	1st meeting w/ customer				
2	Preliminary outline				
	Proposal Stage				
3	Develop proposal	6	4	2	9
4	Presentation to customer	6	1	2	8
5	Develop contract				
6	Create detailed program outline				
7	Write scripts				
	Development Stage				
8	Create multimedia engine				
9	Create dummy graphics				
10	Develop dummy interface				
11	Create preliminary tests				
	Production Stage				
12	Develop graphics				
13	Develop multimedia program				
14	Shoot video	6	6	6	5
15	Capture narration	7	8	9	9
16	Offline edit	2	5	5	8
17	Final graphics				
	Post-Production				
18	Online edit	4	6	7	9
19	Final assemble				
20	Burn gold CD-ROM's				
21	Beta test				
	Completion				
22	Final revisions	5	6	7	7

Table C19. Preferences for Resource Group 2.

Activity	Activity Name	Multeye Media			
		Employee1	Employee 2	Employee 3	Employee 4
1	1st meeting w/ customer				
2	Preliminary outline				
	Proposal Stage				
3	Develop proposal	$(P)_1=4 \cdot y_3^{1,1}$ $(P)_2=7 \cdot y_3^{1,2}$ $(P)_3=4 \cdot y_3^{1,3}$ $(P)_4=5 \cdot y_3^{1,4}$	$(P)_1=2 \cdot y_3^{1,1}$ $(P)_2=7 \cdot y_3^{1,2}$ $(P)_3=8 \cdot y_3^{1,3}$ $(P)_4=7 \cdot y_3^{1,4}$	$(P)_1=7 \cdot y_3^{1,1}$ $(P)_2=8 \cdot y_3^{1,2}$ $(P)_3=8 \cdot y_3^{1,3}$ $(P)_4=1 \cdot y_3^{1,4}$	$(P)_1=4 \cdot y_3^{1,1}$ $(P)_2=7 \cdot y_3^{1,2}$ $(P)_3=4 \cdot y_3^{1,3}$ $(P)_4=5 \cdot y_3^{1,4}$
4	Presentation to customer	$(P)_5=3 \cdot y_3^{1,1}$ $(P)_6=2 \cdot y_3^{1,12}$ $(P)_7=8 \cdot y_3^{1,3}$ $(P)_8=9 \cdot y_3^{1,4}$	$(P)_5=2 \cdot y_3^{1,1}$ $(P)_6=7 \cdot y_3^{1,12}$ $(P)_7=8 \cdot y_3^{1,3}$ $(P)_8=7 \cdot y_3^{1,4}$	$(P)_5=7 \cdot y_3^{1,1}$ $(P)_6=8 \cdot y_3^{1,12}$ $(P)_7=8 \cdot y_3^{1,3}$ $(P)_8=1 \cdot y_3^{1,4}$	$(P)_5=4 \cdot y_3^{1,1}$ $(P)_6=7 \cdot y_3^{1,12}$ $(P)_7=4 \cdot y_3^{1,3}$ $(P)_8=5 \cdot y_3^{1,4}$
5	Develop contract				
6	Create detailed program outline				
7	Write scripts				
	Development Stage				
8	Create multimedia engine	5	3	7	8
9	Create dummy graphics				
10	Develop dummy interface	8	5	4	5
11	Create preliminary tests				
	Production Stage				

			Preferences		
		Res. Group 2:	Multeye Media		
Activity	Activity Name	Employee1	Employee 2	Employee 3	Employee 4
12	Develop graphics				
13	Develop multimedia program	6	5	7	6
14	Shoot video				
15	Capture narration				
16	Offline edit				
17	Final graphics				
	Post-Production				
18	Online edit				
19	Final assemble	6	5	5	6
20	Burn gold CD-ROM's	4	4	4	5
21	Beta test				
	Completion				
22	Final revisions	7	6	7	7

Table C20. Preferences for Resource Group 3.

Staff		Staff			
Activity	Activity Name	Larry	Gloria	Bud	Susan
1	1st meeting w/ customer	6	4	7	8
2	Preliminary outline	7	5	4	3
	Proposal Stage				
3	Develop proposal	$(P)_1=5 \cdot y_3^{1,1}$ $(P)_2=7 \cdot y_3^{1,2}$ $(P)_3=8 \cdot y_3^{1,3}$ $(P)_4=3 \cdot y_3^{1,4}$ $(P)_5=3 \cdot y_3^{2,1}$ $(P)_6=2 \cdot y_3^{2,2}$ $(P)_7=8 \cdot y_3^{2,3}$ $(P)_8=9 \cdot y_3^{2,4}$	$(P)_1=2 \cdot y_3^{1,1}$ $(P)_2=5 \cdot y_3^{1,2}$ $(P)_3=8 \cdot y_3^{1,3}$ $(P)_4=7 \cdot y_3^{1,4}$	$(P)_1=3 \cdot y_3^{1,1}$ $(P)_2=8 \cdot y_3^{1,2}$ $(P)_3=2 \cdot y_3^{1,3}$ $(P)_4=1 \cdot y_3^{1,4}$	$(P)_1=3 \cdot y_3^{1,1}$ $(P)_2=8 \cdot y_3^{1,2}$ $(P)_3=4 \cdot y_3^{1,3}$ $(P)_4=5 \cdot y_3^{1,4}$ $(P)_5=3 \cdot y_3^{2,1}$ $(P)_6=6 \cdot y_3^{2,2}$ $(P)_7=8 \cdot y_3^{2,3}$ $(P)_8=3 \cdot y_3^{2,4}$
4	Presentation to customer	6	7	7	9
5	Develop contract	5	3	5	2
6	Create detailed program outline	$(P)_9=7 \cdot y_3^{2,1}$ $(P)_{10}=4 \cdot y_3^{2,2}$ $(P)_{11}=4 \cdot y_3^{2,3}$ $(P)_{12}=9 \cdot y_3^{2,4}$	$(P)_5=6 \cdot y_3^{2,1}$ $(P)_6=6 \cdot y_3^{2,2}$ $(P)_7=8 \cdot y_3^{2,3}$ $(P)_8=9 \cdot y_3^{2,4}$	7	$(P)_9=7 \cdot y_3^{2,1}$ $(P)_{10}=4 \cdot y_3^{2,2}$ $(P)_{11}=4 \cdot y_3^{2,3}$ $(P)_{12}=9 \cdot y_3^{2,4}$
7	Write scripts	6	8	7	6

			Preferences		
	Staff	Res. Group 3:	Staff		
Activity	Activity Name	Larry	Gloria	Bud	Susan
	Development Stage				
8	Create multimedia engine				
9	Create dummy graphics	1	1	8	1
10	Develop dummy interface	1	1	8	3
11	Create preliminary tests	7	6	7	7
	Production Stage				
12	Develop graphics	4	3	8	4
13	Develop multimedia program				
14	Shoot video	3	5	6	4
15	Capture narration				
16	Offline edit				
17	Final graphics	3	3	7	3
	Post-Production				
18	Online edit	4	3	5	3
19	Final assemble	3	3	3	3
20	Burn gold CD-ROM's	3	3	3	3
21	Beta test	7	5	7	9

			Preferences		
	Staff	Res. Group 3:	Staff		
Activity	Activity Name	Larry	Gloria	Bud	Susan
	Completion				
22	Final revisions	$(P)_{13}=5 \cdot y_3^{1,1}$ $(P)_{14}=7 \cdot y_3^{1,2}$ $(P)_{15}=8 \cdot y_3^{1,3}$ $(P)_{16}=3 \cdot y_3^{1,4}$ $(P)_{17}=3 \cdot y_3^{2,1}$ $(P)_{18}=2 \cdot y_3^{2,2}$ $(P)_{19}=8 \cdot y_3^{2,3}$ $(P)_{20}=9 \cdot y_3^{2,4}$	$(P)_9=7 \cdot y_3^{2,1}$ $(P)_{10}=4 \cdot y_3^{2,2}$ $(P)_{11}=4 \cdot y_3^{2,3}$ $(P)_{12}=9 \cdot y_3^{2,4}$	5	7

Table C21. Time Availability for Resource Group 1.

Activity	Activity Name	NuView Productions			
		Employee1	Employee 2	Employee 3	Employee 4
1	1st meeting w/ customer				
2	Preliminary outline				
	Proposal Stage				
3	Develop proposal	desired[5]	desired[2,4,10]	desired[10]	desired[7]
4	Presentation to customer	desired[5]	desired[2,4,10]	desired[10]	desired[8]
5	Develop contract				
6	Create detailed program outline				
7	Write scripts				
	Development Stage				
8	Create multimedia engine				
9	Create dummy graphics				
10	Develop dummy interface				
11	Create preliminary tests				
	Production Stage				
12	Develop graphics				
13	Develop multimedia program				
14	Shoot video	desired[5]	desired[2,4,10]	desired[15]	desired[12]
15	Capture narration	desired[5]	desired[2,4,10]	desired[15]	desired[13]
16	Offline edit	desired[5]	desired[2,4,10]	desired[15]	desired[16]
17	Final graphics				
	Post-Production				
18	Online edit	desired[5]	desired[2,4,10]	desired[15]	desired[18]
19	Final assemble				
20	Burn gold CD-ROM's				
21	Beta test				
	Completion				
22	Final revisions	desired[5]	desired[2,4,10]	desired[15]	desired[20]

Table C22. Time Availability for Resource Group 2.

Activity	Activity Name	Multeye Media			
		Employee1	Employee 2	Employee 3	Employee 4
1	1st meeting w/ customer				
2	Preliminary outline				
	Proposal Stage				
3	Develop proposal	interval[0,15]	interval[0,15]	desired[7]	desired[12]
4	Presentation to customer	interval[0,15]	desired[9]	desired[7]	desired[6]
5	Develop contract				
6	Create detailed program outline	desired[14]	desired[14]	desired[10]	desired[7]
7	Write scripts				
	Development Stage				
8	Create multimedia engine	desired[11]	desired[11]	desired[14]	desired[10]
9	Create dummy graphics				
10	Develop dummy interface	interval[3,40]	desired[25]	desired[3,4,20]	
11	Create preliminary tests				
	Production Stage				
12	Develop graphics				
13	Develop multimedia program	desired[18]	desired[28]	desired[20]	desired[20]
14	Shoot video				
15	Capture narration				
16	Offline edit				
17	Final graphics				
	Post-Production				
18	Online edit				
19	Final assemble				
20	Burn gold CD-ROM's				
21	Beta test				
	Completion				
22	Final revisions	desired[40]	desired[28]	desired[28]	desired[32]

Table C23. Time Availability for Resource Group 3.

Activity	Activity Name	Staff			
		Larry	Gloria	Bud	Susan
1	1st meeting w/ customer	interval[0, 1000]	interval[0, 1000]	interval[0, 1000]	interval[0, 1000]
2	Preliminary outline	interval[0, 1000]	interval[0, 1000]	desired[4]	desired[7]
	Proposal Stage				
3	Develop proposal	desired[5]	desired[7]	desired[5]	interval[0,10]
4	Presentation to customer				
5	Develop contract	desired[8]	desired[2]	desired[10]	
6	Create detailed program outline				
7	Write scripts				
	Development Stage				
8	Create multimedia engine				
9	Create dummy graphics	desired[20]	desired[28]	desired[18]	desired[30]
10	Develop dummy interface	desired[20]	desired[28]	desired[18]	desired[30]
11	Create preliminary tests				
	Production Stage				
12	Develop graphics				
13	Develop multimedia program				
14	Shoot video	desired[28]	desired[30]	desired[30]	desired[30]
15	Capture narration				
16	Offline edit				
17	Final graphics	desired[30]	desired[30]	desired[30]	desired[30]
	Post-Production				
18	Online edit				
19	Final assemble				
20	Burn gold CD-ROM's				
21	Beta test				
	Completion				
22	Final revisions				

EXAMPLE PROJECT #2: OUTPUT

The structure of this project is similar to the previous one. However, the input data differs in the fact that no information is provided on costs. Instead, resource-activity mapping with respect to resource availability may become of interest since that data is provided. With that respect, consider the following fictitious mapping strategy, as shown in Figure C25.

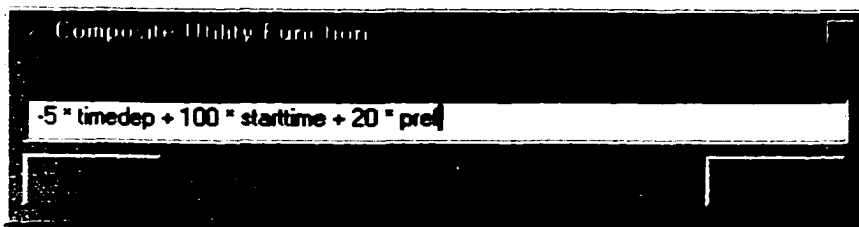


Figure C25. Example Mapping Strategy.

Figure C25 indicates that the primary mapping objective is satisfying resource choices with respect to their availability, while the preferences and especially time capabilities and dependencies are of secondary issues. This strategy will produce a Gantt chart as displayed in Figure C26 and *resource-activity grids* as shown in Figures B27-B29.

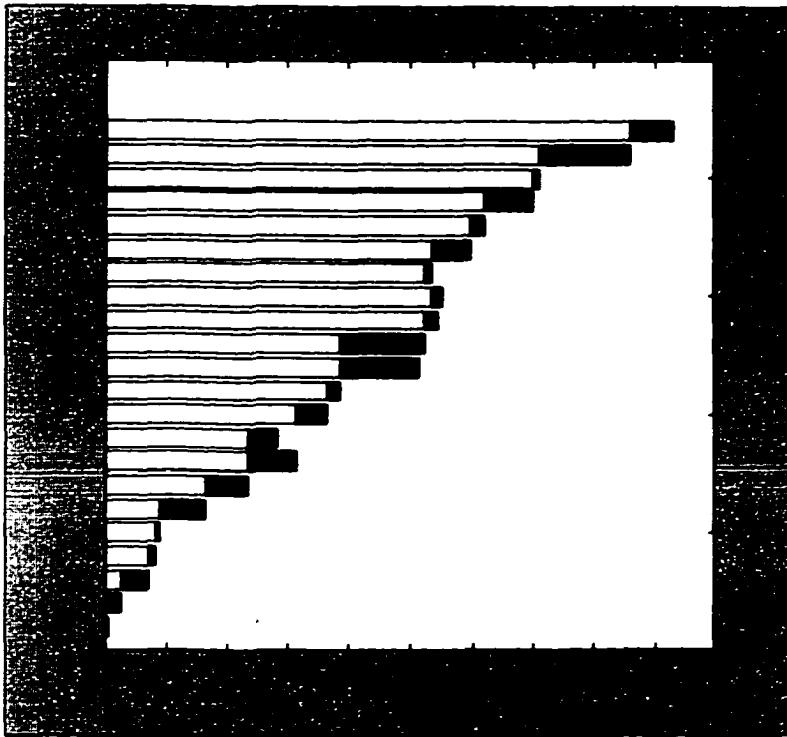


Figure C26. Project Gantt Chart for a Schedule Emphasized on Resource Availability.

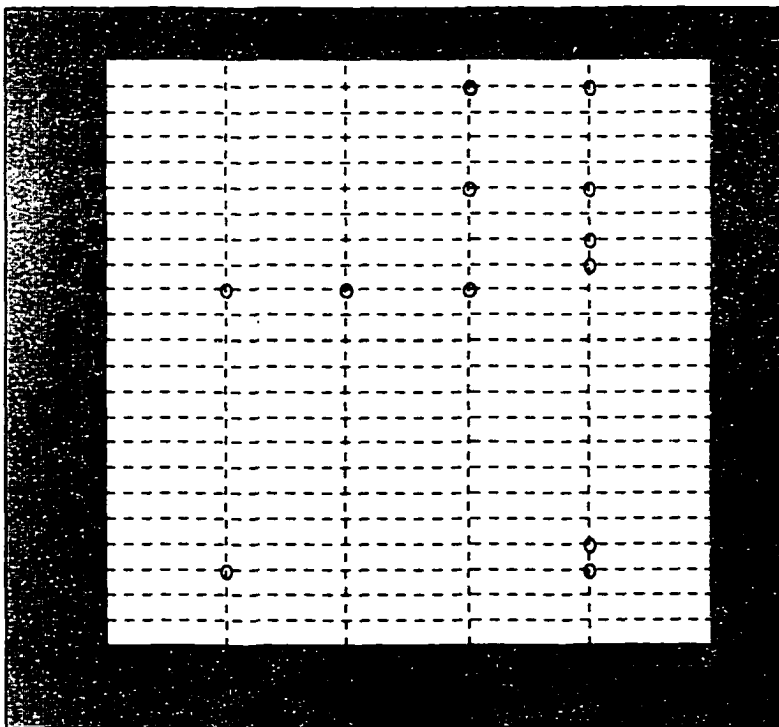


Figure C27. Resource-Activity Grid for Type 1 of Strategy Emphasized on Resource Availability.

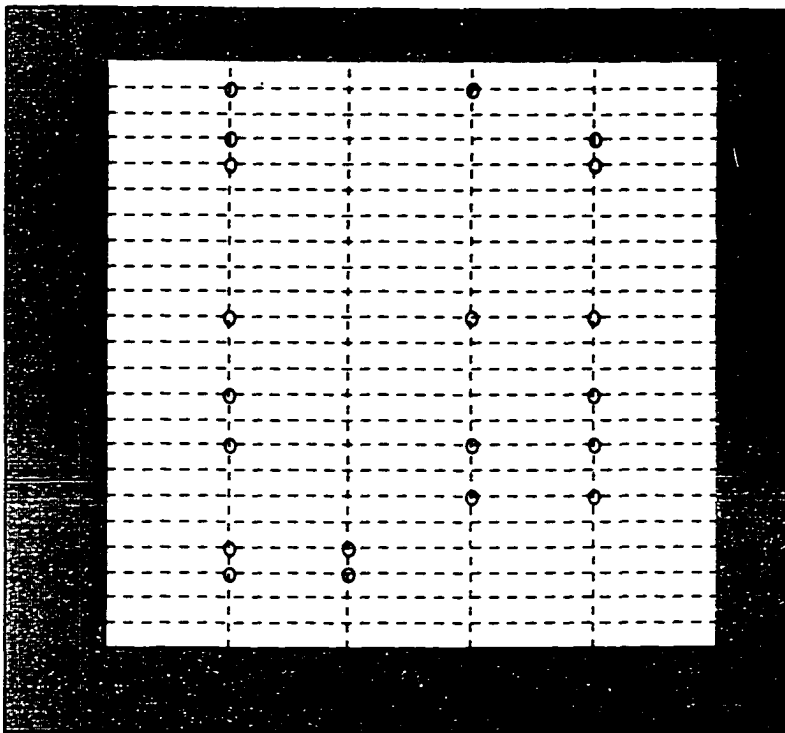


Figure C28. Resource-Activity Grid for Type 2 of Strategy Emphasized on Resource Availability.

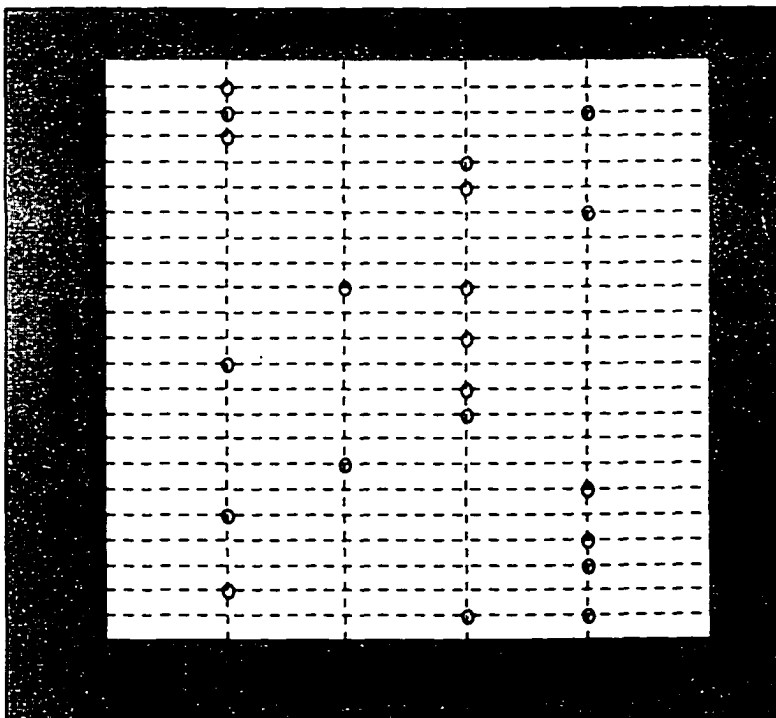


Figure C29. Resource-Activity Grid for Type 3 of Strategy Emphasized on Resource Availability.

Assume now that the mapping strategy is changed and that the availability of resource group or type 2 should be the only one considered. In addition, assume that preferences are granted only to resource units in group three, but only the first 25 time units. The time-effective capabilities are considered as previously, during the entire project schedule. This new strategy (that is, mapping objective) may be modeled as follows:

$$\text{-timedep} + 100 * \text{starttime} * \text{kroncker}(\text{restype},2) + \text{pref} * \text{kroncker}(\text{restype},3) * \text{interval}([0,25],\text{time})$$

The output of the above objective in terms of project duration and resource-activity mapping is shown in Figures B30-B33. Comparing Figures B26 and B30 we should notice that the new project schedule with a relaxed resource mapping strategy results in shorter project duration (i.e., in savings of over eight time units). Also by comparing the previous with the following *resource-activity mapping grids*, we notice that resource units assignments were also changed (although, as previously discussed, the required number of units needed for each activity is always held constant, regardless of the strategy).

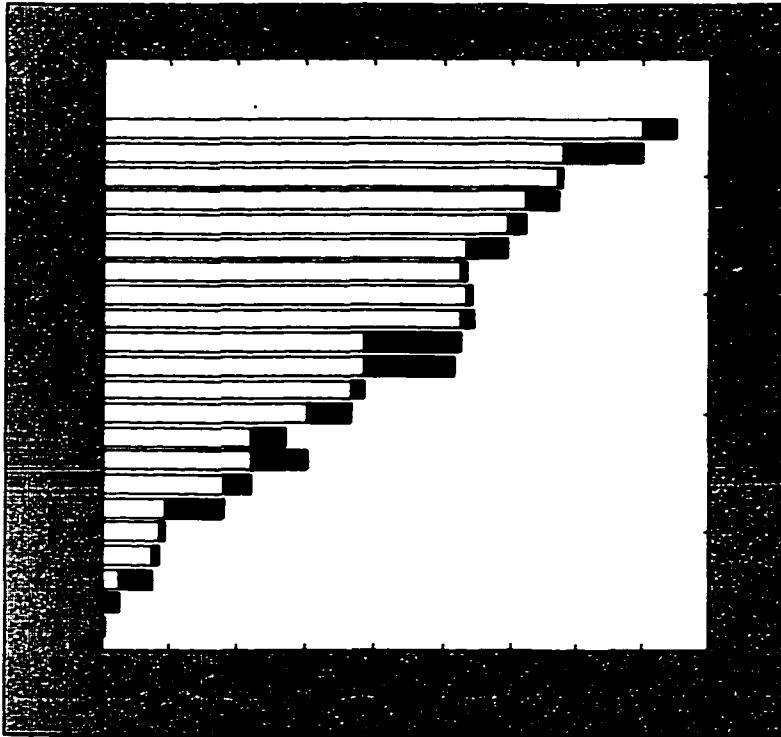


Figure C30. Relaxed Resource Mapping Strategy Results in Shorter Project Duration.

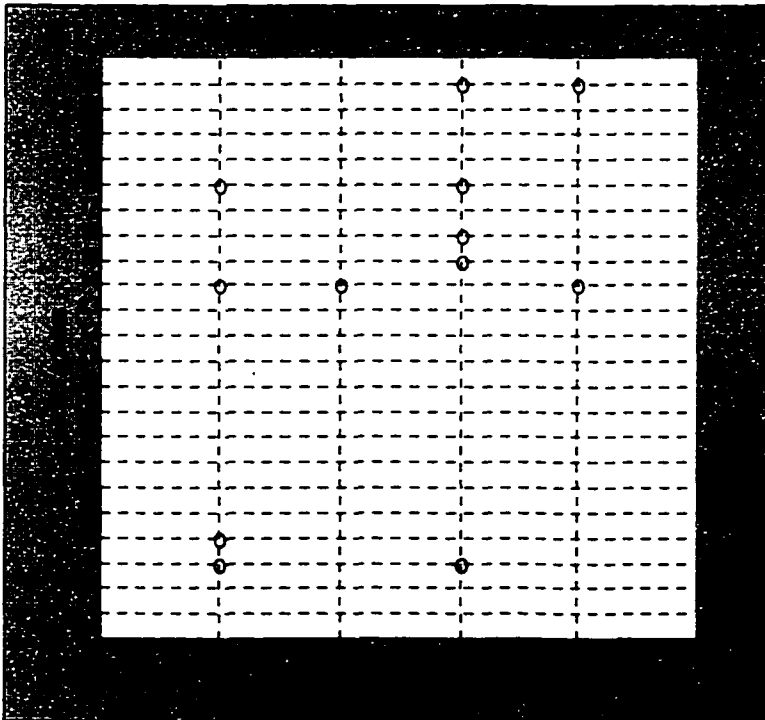


Figure C31. Resource Group 1 Assignments Resulting from a Change in Strategy.

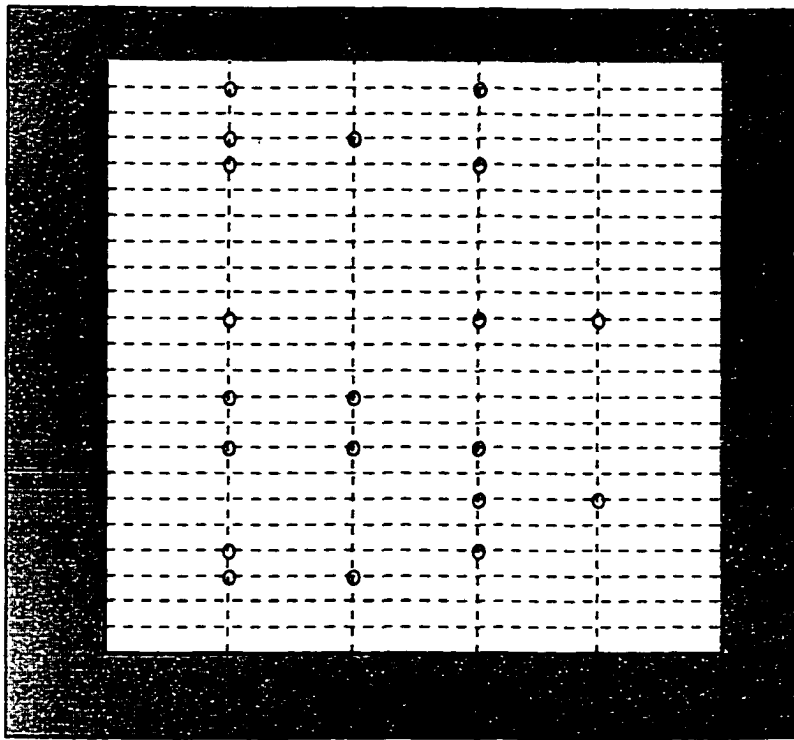


Figure C32. Resource Group 2 Assignments Resulting from a Change in Strategy.

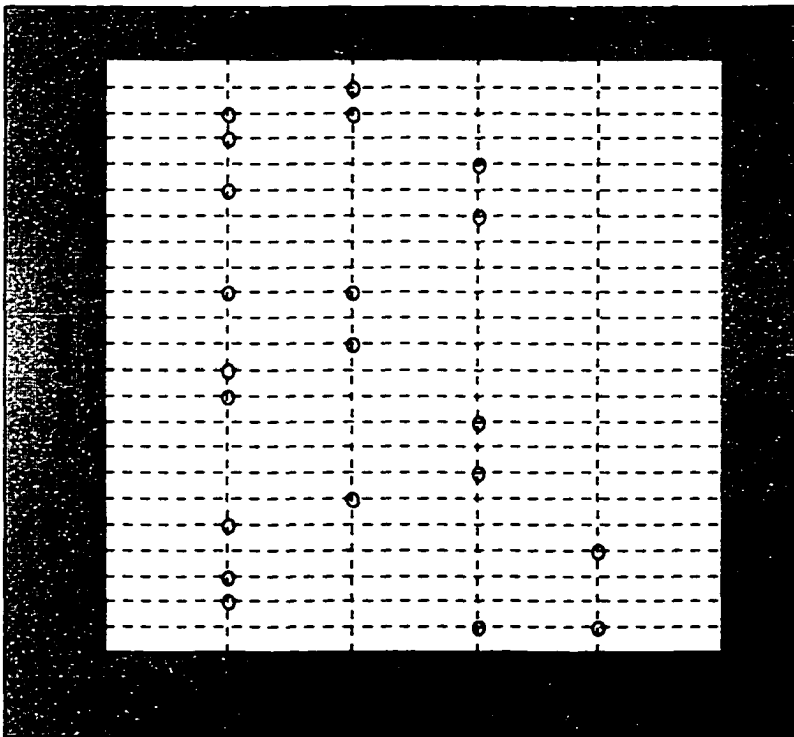


Figure C33. Resource Group 3 Assignments Resulting from a Change in Strategy.

APPENDIX D

COMPUTER CODES FOR *PROMAP* IMPLEMENTATION

```
function [a]=amatrix(actneeds,cand,finished,reslimits,inprogress,
typeselect)

real=actneeds(cand,:)' ;
[ row,col]=size(real);
first=zeros(row,1);
second=ones(row,1);

first=(reslimits-sum(actneeds(inprogress,:),1))';

if isempty(typeselect)==1
    a=[real first];
else
    modifiedreal=real(typeselect, :);
    second=sum(actneeds(finished,typeselect),1)';
    a=[real first; (-modifiedreal) -second];
end

function [a]=amatrixdown(actneeds, cand,finished,reslimits,inprogress)

real=actneeds(cand,:)' ;
[ row,col]=size(real);
first=zeros(row,1);

first=(reslimits-sum(actneeds(inprogress,:),1))';
second=sum(actneeds(finished,:),1)';
a=[real first];

function [cand]=candidates(dynpred)

cand=find(sum(dynpred,2)<1);
```

```

function [scheduled]=chart(time,newlyadded,scheduled,actdur)

z=length(newlyadded');
timemat=time*(ones(1,z));
if isempty(newlyadded)==0
    temp=[newlyadded'; timemat;timemat+actdur(newlyadded)];
else
    temp=[];
end
scheduled=[scheduled temp];

function [numsucc]=children(pred)

[x,y]=size(pred);

for g=1:x
    numsucc(g)=sum(sum(pred==g));
end
numsucc=numsucc/max(numsucc);

function [b]=constraints(inprogress,finished,reslimits,actneeds,
typeselect)

if isempty(inprogress)==1
    inprogress=zeros(sum(actneeds(inprogress,:),1),1);
end

bceiling=reslimits-sum(actneeds(inprogress,:),1);

if isempty(typeselect)==1
    b=[(bceiling)];
else
    bfloor=sum(actneeds(finished,typeselect),1);

b=[(bceiling) (-bfloor)];

end

```

```
function [b]=constraintsdown(inprogress,finished,reslimits,actneeds)
```

```
if isempty(inprogress)==1
    inprogress=zeros(sum(actneeds(inprogress,:),1));
end
```

```
bceiling=reslimits-sum(actneeds(inprogress,:),1);
```

```
b=[(bceiling)];
```

```
function [est,lst]=cpm(actdur,pred)
```

```
numnodes=length(actdur);
```

```
numarcs=length(find(pred));
```

```
f=ones(1,numnodes);
b=[];
c=[];
a=zeros(numarcs,numnodes);
incr=1;
for i=1:numnodes
    for j=1:sum(any(pred(i,:),i))
        c=[c;pred(i,j) i];
        test=[i j pred(i,j)];
        b=[b -actdur(pred(i,j))];
        a(incr,pred(i,j))=1;
        a(incr,i)=-1;
        incr=incr+1;
    end
end
```

```
a=[a;-eye(numnodes)];
b=[b zeros(1,numnodes)];
est=lp(f,a,b);
```

```
% Finding the terminal activities
% (Those with no successors)
```

```
terminal=[];
for m=1:numnodes
    if isempty(find(pred==m))==1
        terminal=[terminal m];
    end
end
```

```
% Finding the maximal EFT
```

```
eft=est+actdur';
eftmax=max(eft);
```

```
% Finding the activity with maximal LST
termmax=find(eft==eftmax);
```

```
% Calculating LST for terminal activities
```

```

addconst=zeros(length(terminal), numnodes);
addb=[];
for m=1:length(terminal)
    addconst(m,terminal(m))=1;
addb(m)=eftmax-actdur(terminal(m));
end

% Calculating LST
f=-f;
a=[addconst;a];
b=[addb b];
lst=lp(f,a,b);

%***** Plotting The Resource Utilization Graphs
%*****

figure;
abscis=[abscis scheduled(end)];
usage=[usage usage(:,end)];
for v=1:length(reslimits)
    subplot(length(reslimits)+1,1,v), stairs(abscis,usage(v,:))
    %stairs(abscis,usage(v,:));
    axis([0 scheduled(end) 0 reslimits(v)+1]);
    if v==1
        title(['Project is completed at t = '
num2str(scheduled(3,end))]);
    end

    xlabel('Time');
    ylabel('Resource Units')
end

%***** Plotting the Gantt Chart
%*****

for r=1:length(actdur)
    data(1,r)=scheduled(2,find(scheduled(1,:)==r));
end

data(2,:)=actdur;

subplot(length(reslimits)+1,1,length(reslimits)+1),
barh(data','stack'), colormap([1 1 1;0 0 0]);
set(gca,'color','white');
xlabel('Time')
ylabel('Activities')

%***** Plotting the Resource Units Assignment *****

if choice==3 | choice==4

for restype=1:length(reslimits)

```

```

figure;
grid;
xticks=1:reslimits(restype);
yticks=1:numact;
axis([0 reslimits(restype)+1 0 size(actneeds,1)+1]);
set(gca,'XTick',xticks);
set(gca,'YTick',yticks);
hold;
for nact=1:numact
    vect=find([acttype(nact,restype).unit(:).assigned]);
    plot(vect,nact, 'ro');
end
title(sprintf('Mapping Resource Type %.0f Units to Project
Activities', restype));
xlabel(sprintf('Resource Type %.0f Units',restype));
ylabel('Project Activities');

hold off;
end

for restype=1:length(reslimits)
figure;

xticks=1:reslimits(restype);
axis([0 sum(reslimits)+1 0 1]);
set(gca,'XTick',xticks);
hold;
maxunittime=zeros(1,reslimits(restype));
minunittime=zeros(1,reslimits(restype));
for nunit=1:reslimits(restype)
    for nact=1:numact

maxunittime(nunit)=maxunittime(nunit)+(acttype(nact,restype).unit(nunit).assigned)*actdur(nact);

minunittime(nunit)=minunittime(nunit)+(acttype(nact,restype).unit(nunit).assigned)*(acttype(nact,restype).unit(nunit).tuned);

    end
    maxunittime(nunit)= maxunittime(nunit)/scheduled(3,end);
    minunittime(nunit)= minunittime(nunit)/scheduled(3,end);

end

bar(maxunittime,'r');
bar(minunittime, 'b');
title(sprintf('Time Percentage of Resource Type %.0f Units Engagement
vs. Total Project Duration', restype));
xlabel(sprintf('Resource Type %.0f Units',restype));
ylabel('Percentage of Total Project Duration');

hold off;
end
end bend choice

```

```

function [fuzstart]=desstart(instart,time)

if length(instart)==3
    fuzstart=1/(1+ instart(1)*(time - instart(3))^instart(2));
elseif length(instart)==1
    fuzstart=1/(1+(time - instart)^2);
end

function [pulse]=kronecker(restype, destype)
if destype==restype
    pulse=1;
else
    pulse=0;
end

function [actdur]=duration(acttype, actneeds, numact, numres)

actdur=zeros(1,numact);
for i=1:numact
    for j=1:numres
        if isfield(acttype(i,j).unit(:),'tuned')==1
            actdriversort=sort([acttype(i,j).unit(:).tuned]);
            if length(actdriversort)<actneeds(i,j) |
isempty(actdriversort)
                actdur(i)=max(actdur(i),0);
            elseif actneeds(i,j)~=0
                actdur(i)=max(actdur(i),actdriversort(actneeds(i,j)));
            end %end if length
        end %end if isfield
    %break
    end %for j=1:numres
end %for

```

```

function fig = dynamo()
% This is the machine-generated representation of a Handle Graphics
object
% and its children. Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was saved.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

load dynamo

h0 = figure('Color',[0.8 0.8 0.8], ...
    'Colormap','mat0', ...
    'MenuBar','none', ...
    'Name','Welcome', ...
    'NumberTitle','off', ...
    'PointerShapeCData',mat1, ...
    'Position',[320 270 175 75], ...
    'Tag','Fig1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[0 37.5 132 18.75], ...
    'String','Resource Mapping Tool v.1.0', ...
    'Style','text', ...
    'Tag','StaticText1');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[0 18.75 131.25 18.75], ...
    'String','by', ...
    'Style','text', ...
    'Tag','StaticText2');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[0 0 131.25 18.75], ...
    'String','Milan Milatovic', ...
    'Style','text', ...
    'Tag','StaticText3');
if nargin > 0, fig = h0; end

drawnow;
for i=1:600000
end
close;

```

```

function [prior]=floatweight(cand,lst,actdur,time)

for i=1:length(cand)
    prior(i)=actdur(cand(i))/(lst(cand(i))-actdur(cand(i))-time);

    prior(i)=(time + actdur(cand(i)))/(actdur(cand(i))+lst(cand(i)));

end

```

```

function [crisp]=fuz(a,b,c,d)

if nargin==4
    crisp=(-(a)^2 - (b)^2 + (c)^2 + (d)^2 -(a*b) + (c*d))/(3*(-a - b +
c + d));
elseif nargin==3
    crisp=(-(a)^2 + (c)^2 - (a*b) + (b*c))/(3*(-a +c));
else
    error('Unrecognized Fuzzy Input');
end

```

```

function acttype=getarbitrary( reslimits, acttype, refactor, typedep,
unitdep, funcstr)

for act=1:length(refactor)
    for tdep=1:length(typedep)
        for udep=1:length(unitdep)

            if exist('acttype')==1 %finding the index where to put
the newly added function
                dummy=eval('size(acttype(refactor(act),
typedep(tdep)).unit(unitdep(udep)).func,2)+1','1');
            else
                dummy=1;
            end

            conditionleft=unitdep(udep); %making sure that units of a
particular res type are not exceeded
            conditionright=reslimits(typedep(tdep));

            if conditionleft<=conditionright

                acttype(refactor(act),
typedep(tdep)).unit(unitdep(udep)).func{dummy}=funcstr;
                else

                    break
            end %end if unitdep
        end
    end
end

```

```

        end
    end
end

function fig = getdata()
% This is the machine-generated representation of a Handle Graphics
object
% and its children. Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was saved.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

load getdata

h0 = figure('Color',[0.8 0.8 0.8], ...
    'Colormap',mat0, ...
    'MenuBar','none', ...
    'NumberTitle','off', ...
    'Name','Enter Basic Project Data',...
    'PointerShapeCData',mat1, ...
    'Position',[71 132 678 392], ...
    'Tag','Fig1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[248.25 18 246.75 255], ...
    'Style','frame', ...
    'Tag','Frame1');

% Activity Listbox

actlist_call=[
    'h_actlist=findobj(''Tag'', ''Listbox1'');'...
    'h_pred=findobj(''Tag'', ''EditText5'');'...
    'h_typedlist=findobj(''Tag'', ''Listbox2'');'...
    'h_actneedstext=findobj(''Tag'', ''StaticText4'');'...
    'actvalue=get(h_actlist, ''value'');'...
    'typevalue=get(h_typedlist, ''value'');'...
    'set(h_actneedstext, ''string'', sprintf(''Number of resource type
%.0f units required by activity %.0f:'', typevalue, actvalue));'...
    'predstr=num2str(actvalue);'...
    'set(h_pred, ''string'', predstr);'
];

h1 = uicontrol('Parent',h0, ...
    'callback',actlist_call,...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Position',[22.5 16.5 90 240], ...
    'String',' ', ...
    'max', 2,...

```

```

        'Style','listbox', ...
        'Tag','Listbox1', ...
        'Value',1);
    % Resource Type Listbox

    typelist_call=[
        'h_avail=findobj(''tag'', ''EditText3'');'...
        'h_numres=findobj(''Tag'', ''EditText2'');'...
        'h_typelist=findobj(''Tag'', ''Listbox2'');'...
        'h_availtext=findobj(''Tag'', ''StaticText3'');'...
        'h_actneedsedit=findobj(''Tag'', ''EditText4'');'...
        'h_actneedstext=findobj(''Tag'', ''StaticText4'');'...
        'h_actlist=findobj(''Tag'', ''Listbox1'');'...
        'typevalue=get(h_typelist, ''value'');'...
        'reslimits=get(h_avail, ''userdata'');'...
        'actneeds=get(h_actneedsedit, ''userdata'');'...
        'actvalue=get(h_actlist, ''value'');'...
        'set(h_availtext, ''string'', sprintf(''Units of resource type %.0f
available:'', typevalue));'...
        'if typevalue <= length(reslimits) & reslimits(typevalue)~=0, '...
        'set(h_avail, ''string'', num2str(reslimits(typevalue)));'...
        'else, set(h_avail, ''string'', '');'...
        'end;'...
        'set(h_actneedstext, ''string'', sprintf(''Number of resource type
%.0f units required by activity %.0f:'', typevalue, actvalue));'
    ];

    h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'callback',typelist_call,...
        'BackgroundColor',[1 1 1], ...
        'Position',[135 16.5 90 240], ...
        'String',' ', ...
        'max', 2,...
        'Style','listbox', ...
        'Tag','Listbox2', ...
        'Value',1);

    Number of activities Edit

    numactedit_call=[
        'h_numact=findobj(''Tag'', ''EditText1'');'...
        'h_actlist=findobj(''Tag'', ''Listbox1'');'...
        'h_popup=findobj(''Tag'', ''popupmenu1'');'...
        'numact=get(h_numact, ''String'');'...
        'numact=str2num(numact);'...
        'actstr='''Activity 1''';'...
        'for i=2:numact, actstr=[actstr sprintf(''|Activity
%.0f'', i)];end;'...
        'set(h_actlist, ''string'', actstr);'...
        'set(h_numact, ''userdata'', numact);'...
        'for j=1:numact, popstr(j)={sprintf(''Predecessors of Activity
%.0f'', j)};end;'...
        'set(h_popup, ''string'', popstr);'
    ];

```

```

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'callback',numactedit_call,...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'HorizontalAlignment','left',...
    'Position',[382.5 226.5 45 22.5], ...
    'Style','edit', ...
    'Tag','EditText1');

% Number of Resource Types Edit

numresedit_call=[
    'h_numres=findobj(''Tag'', ''EditText2'');'...
    'h_typelist=findobj(''Tag'', ''Listbox2'');'...
    'numres=get(h_numres, ''String'');'...
    'numres=str2num(numres);'...
    'typestr='''Resource Type 1'';'...
    'for j=2:numres, typestr=[typestr sprintf(''%Resource Type
%.0f'',j)];end;'...
    'set(h_typelist, ''string'', typestr);'...
    'set(h_numres, ''userdata'', numres);'
    ];

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'callback', numresedit_call,...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'HorizontalAlignment','left',...
    'Position',[382.5 189 45 22.5], ...
    'Style','edit', ...
    'Tag','EditText2');

% Number of Resource Type Available

typeavail_call=[
    'h_avail=findobj(''tag'', ''EditText3'');'...
    'h_numres=findobj(''Tag'', ''EditText2'');'...
    'h_typelist=findobj(''Tag'', ''Listbox2'');'...
    'h_availtext=findobj(''Tag'', ''StaticText3'');'...
    'reslimits=get(h_avail, ''userdata'');'...
    'typevalue=get(h_typelist, ''value'');'...
    'avail=get(h_avail, ''string'');'...
    'avail=str2num(avail);'...
    'reslimits(typevalue)=avail;'...
    'set(h_avail, ''userdata'', reslimits);'...
    'numres=get(h_numres, ''userdata'');'...
    'if typevalue < numres, set(h_typelist, ''value'', typevalue+1);'...
    'set(h_avail, ''string'', '');'...
    'set(h_availtext, ''string'', sprintf(''%Units of resource type %.0f
available:'', typevalue+1));'...
    'else,'...
    'set(h_typelist, ''value'', 1);'...
    'set(h_availtext, ''string'', ''Units of resource type 1
available:'');'...
    'end;'
    ];

```

```

h1 = uicontrol('Parent',h0, ...
    'callback', typeavail_call,...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'HorizontalAlignment', 'left',...
    'Position',[382.5 151.5 45 22.5], ...
    'Style','edit', ...
    'Tag','EditText3');

```

% Activity Resource Requirements

```

actneeds_call=[
    'h_actneedsedit=findobj(''Tag'', ''EditText4'');'...
    'h_actlist=findobj(''Tag'', ''Listbox1'');'...
    'h_typelist=findobj(''Tag'', ''Listbox2'');'...
    'h_actneedstext=findobj(''Tag'', ''StaticText4'');'...
    'h_numres=findobj(''Tag'', ''EditText2'');'...
    'h_numact=findobj(''Tag'', ''EditText1'');'...
    'h_dummy=findobj(''tag'', ''EditText5'');'...
    'h_avail=findobj(''tag'', ''EditText3'');'...
    'h_availtext=findobj(''Tag'', ''StaticText3'');'...
    'reslimits=get(h_avail, ''userdata'');'...
    'actvalue=get(h_actlist, ''value'');'...
    'typevalue=get(h_typelist, ''value'');'...
    'actneeds=get(h_actneedsedit, ''userdata'');'...
    'line=get(h_actneedsedit, ''string'');'...
    'actneeds(actvalue, typevalue)=str2num(line);'...
    'set(h_actneedsedit, ''userdata'', actneeds);'...
    'numact=get(h_numact, ''userdata'');'...
    'numres=get(h_numres, ''string'');'...
    'numres=str2num(numres);'...
    'if typevalue < numres, set(h_typelist, ''value'', typevalue+1);'...
    'set(h_actneedsedit, ''string'', '');'...
    'set(h_actneedstext, ''string'', sprintf(''Number of resource type
%.0f units required by activity %.0f: '', typevalue+1, actvalue));'...
    'availstr=num2str(reslimits(typevalue+1));'...
    'set(h_avail, ''string'', availstr);'...
    'set(h_availtext, ''string'', sprintf(''Units of resource type %.0f
available'', typevalue+1));'...
    'else, '...
    'set(h_typelist, ''value'', 1);'...
    'set(h_actneedsedit, ''string'', '');'...
    'if actvalue < numact, set(h_actlist, ''value'', actvalue+1);'...
    'typevalue=get(h_typelist, ''value'');'...
    'set(h_actneedstext, ''string'', sprintf(''Number of resource type
%.0f units required by activity %.0f: '', typevalue, actvalue+1));'...
    'end;'...
    'end;'
];

```

```

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...

```

```

'callback',actneeds_call,...
'BackgroundColor',[1 1 1], ...
'ListboxTop',0, ...
'Position',[382.5 106.5 45 22.5], ...
'HorizontalAlignment','left',...
'Style','edit', ...
'Tag','EditText4');

```

Activity Predecessors Edit

```

predec_call=[
'h_pred=findobj('Tag','EditText5');'...
'h_actlist=findobj('Tag','Listbox1');'...
'h_popup=findobj('Tag','popupmenul');'...
'h_numact=findobj('Tag','EditText1');'...
'pred=get(h_pred,'userdata');'...
'activity=get(h_popup,'value');'...
'line=get(h_pred,'string');'...
'line=str2num(line);'...
'pred(activity,1:length(line))=line;'...
'set(h_pred,'userdata',pred);'...
'numact=get(h_numact,'string');'...
'numact=str2num(numact);'...
'if activity < numact, set(h_popup,'value',activity+1);'...
'else, set(h_popup,'value',1);end;'...
'set(h_pred,'string','');'
];

h1 = uicontrol('Parent',h0, ...
'callback',predec_call,...
'Units','points', ...
'BackgroundColor',[1 1 1], ...
'ListboxTop',0, ...
'HorizontalAlignment','left',...
'Position',[382.5 69 105 22.5], ...
'Style','edit', ...
'Tag','EditText5');

h1 = uicontrol('Parent',h0, ...
'Units','points', ...
'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
'HorizontalAlignment','right', ...
'ListboxTop',0, ...
'Position',[277.5 226.5 105 15], ...
'String',' Number of activities:', ...
'Style','text', ...
'Tag','StaticText1');

h1 = uicontrol('Parent',h0, ...
'Units','points', ...
'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
'HorizontalAlignment','right', ...
'ListboxTop',0, ...
'Position',[277.5 189 105 15], ...
'String','Number of resource types:', ...
'Style','text', ...
'Tag','StaticText2');

```

```

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[277.5 153.75 105 20.25], ...
    'String','Units of resource type 1 available:', ...
    'Style','text', ...
    'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[278.25 105.75 105 21], ...
    'String','Number of resource type 1 units required by activity 1:',
...
    'Style','text', ...
    'Tag','StaticText4');

    Popup menu

popup_call=[
    'h_popup=findobj(''Tag'', ''popupmenu1'');'...
    'h_pred=findobj(''Tag'', ''EditText5'');'...
    'activity=get(h_popup, ''value'');'...
    'pred=get(h_pred, ''userdata'');'...
    'if activity <= size(pred,1),'...
    'line=pred(activity,:);'...
    'line=num2str(line);'...
    'set(h_pred, ''string'',line);'...
    'end;'
    ];

h1 = uicontrol('Parent',h0, ...
    'callback', popup_call,...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[255 69 127.5 15], ...
    'String','Predecessors of Activity 1:', ...
    'Style','popupmenu', ...
    'Tag','popupmenu1');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[22.5 256.5 90 15], ...
    'String','Activities', ...

```

```

        'Style','text', ...
        'Tag','StaticText6');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[135 256.5 90 15], ...
    'String','Resource Types', ...
    'Style','text', ...
    'Tag','StaticText7');

! Accept Pushbutton

accept_call=[
    'h_accept=findobj(''Tag'', ''Pushbutton1'');'...
    'h_exit=findobj(''Tag'', ''Pushbutton2'');'...
    'set(h_accept, ''userdata'', 1);'...
    'set(h_accept, ''string'', ''Done'');'
    ];

h1 = uicontrol('Parent',h0, ...
    'callback', accept_call,...
    'Units','points', ...
    'ListboxTop',0, ...
    'Position',[ 382.5 22.5 45 22.5 ], ...
    'String','Accept All', ...
    'Tag','Pushbutton1');

exit_call=[
    'h_exit=findobj(''Tag'', ''Pushbutton2'');'...
    'set(h_exit, ''userdata'', 1);'
    ];

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'callback','close',...
    'ListboxTop',0, ...
    'Position',[ 442.5 22.5 45 22.5 ], ...
    'String','Exit', ...
    'Tag','Pushbutton2');

h_accept=findobj('Tag','Pushbutton1');
h_exit=findobj('Tag','Pushbutton2');

while ~length(get(h_accept,'userdata')) &
~length(get(h_exit,'userdata'))
    drawnow
end

h_numact=findobj('Tag','EditText1');
h_numres=findobj('Tag','EditText2');
h_avail=findobj('tag','EditText3');
h_actneedsedit=findobj('Tag','EditText4');
h_pred=findobj('Tag','EditText5');

numact=get(h_numact,'userdata');

```

```
numres=get(h_numres, 'userdata');  
reslimits=get(h_avail, 'userdata');  
actneeds=get(h_actneedsedit, 'userdata');  
pred=get(h_pred, 'userdata');  
  
uiwait(h0);  
  
if nargout > 0, fig = h0; end
```

```

function [acttype] = getfunctions(numact,numres,reslimits,acttype)
% This is the machine-generated representation of a Handle Graphics
object
% and its children. Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was saved.
%
% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

```

```

load getfunctions

```

```

actstr='Activity 1';
for i=2:numact
    actstr=[actstr '|Activity 'num2str(i)'];
end

```

```

restypestr='Type 1';
for j=2:numres
    restypestr=[restypestr '|Type 'num2str(j)'];
end

```

```

unitstring='Unit 1';
for k=2:reslimits(1)
    unitstring=[unitstring '|Unit 'num2str(k)'];
end

```

```

h0 = figure('Color',[0.8 0.8 0.8], ...
    'Colormap',mat0, ...
    'MenuBar','none', ...
    'Name','Resource Functional Dependencies',...
    'NumberTitle','off', ...
    'PointerShapeCData',mat1, ...
    'Position',[48 39 690 527], ...
    'Tag','Fig1');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.7529 0.75294 0.75294], ...
    'ListboxTop',0, ...
    'Position',mat2, ...
    'Style','frame', ...
    'Tag','Frame2');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'ListboxTop',0, ...
    'Position',mat3, ...
    'Style','frame', ...
    'Tag','Frame4');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...

```

```

        'ListboxTop',0, ...
        'Position',mat4, ...
        'Style','frame', ...
        'Tag','Frame1');

% ACTIVITY LIST BOX

actlist_call=[
    'h_actlist=findobj('Tag','Listbox1');'...
    'val=get(h_actlist,'Value');'...
    'val=num2str(val);'...
    'h_actrefedit=findobj('Tag','EditText7');'...
    'set(h_actrefedit,'string',val);'
];

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'callback', actlist_call,...
    'BackgroundColor',[1 1 1], ...
    'Max',2, ...
    'Position',[0.05652 0.47438 0.188405 0.455407], ...
    'Style','listbox', ...
    'string',actstr,...
    'Tag','Listbox1', ...
    'Value',1);

% TYPE LIST BOX

restypelist_call=[
    'h_typelist=findobj('Tag','Listbox2');'...
    'val=get(h_typelist,'Value');'...
    'limits=get(h_typelist,'userdata');'...
    'unitstring='Unit 1';'...
    'for x=2:max(reslimits(val)),unitstring=[unitstring sprintf('Unit
%.0f',x)];,end;'...
    'h_unitlist=findobj('Tag','Listbox3');'...
    'set(h_unitlist,'Value',1);'...
    'set(h_unitlist,'string',unitstring);'...
    ];

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'callback', restypelist_call,...
    'BackgroundColor',[1 1 1], ...
    'Max',2, ...
    'Position',[0.404347 0.474383 0.1884057 0.455407], ...
    'Style','listbox', ...
    'string', restypestr,...
    'Tag','Listbox2', ...
    'Value',1);

% UNIT LIST BOX

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[1 1 1], ...
    'Max',2, ...

```

```

'Position',[0.75217 0.474383 0.1884057 0.455407], ...
'Style','listbox', ...
'string', unitstring,...
'Tag','Listbox3', ...
'Value',1);

```

```

FUNCTION EDITOR

```

```

funcedit_call=[
'h_typedrvedit=findobj('Tag','EditText6');'...
'h_typedepedit=findobj('Tag','EditText5');'...
'h_accept=findobj('Tag','Pushbutton9');'...
'h_unitdrvedit=findobj('Tag','EditText4');'...
'h_unitdepedit=findobj('Tag','EditText3');'...
'h_typelist=findobj('Tag','Listbox2');'...
'h_unitlist=findobj('Tag','Listbox3');'...
'h_refactedit=findobj('Tag','EditText7');'...
'h_funcedit=findobj('Tag','EditText1');'...
'h_actlist=findobj('Tag','Listbox1');'...
'h_popup1=findobj('Tag','popup1');'...
'typedrvstr=get(h_typedrvedit,'string');'...
'typedepstr=get(h_typedepedit,'string');'...
'unitdrvstr=get(h_unitdrvedit,'string');'...
'unitdepstr=get(h_unitdepedit,'string');'...
'refactstr=get(h_refactedit,'string');'...
'funcstr=get(h_funcedit,'string');'...
'mode=get(h_popup1,'Value');'...
'refact=str2num(refactstr);'...
'typedrv=str2num(typedrvstr);'...
'typedep=str2num(typedepstr);'...
'unitdrv=str2num(unitdrvstr);'...
'unitdep=str2num(unitdepstr);'...
'if mode==3,'...
'acttype=getarbitrary( reslimits, acttype, refactor, typedep,
unitdep, funcstr);'...
'else,'...
'acttype=gettime( reslimits, acttype, refactor, typedrv, unitdrv,
typedep, unitdep, funcstr, mode);'...
'if isempty(typedrv) | isempty(unitdrv),'...
'if unitdep<reslimits(typedep),'...
'set(h_unitdepedit,'String',unitdep(end)+1);'...
'else,'...
'set(h_unitdepedit,'String',1);'...
'end;'...
'end;'...
'end;'...
'set(h_accept, 'userdata',acttype);'...
'set(h_funcedit,'string','');'...
'if ~isempty(typedrv) & ~isempty(unitdrv),'...
'if unitdrv<reslimits(typedrv),'...
'set(h_unitdrvedit,'String',unitdrv(end)+1);'...
'else,'...
'set(h_unitdrvedit,'String',1);'...
'end;'...
'end;'
];

```

```
[0.1623188 0.055028 0.782608 0.03795]
```

```
h1 = uicontrol('Parent',h0, ...  
    'callback', funcedit_call,...  
    'Units','normalized', ...  
    'BackgroundColor',[1 1 1], ...  
    'HorizontalAlignment','left', ...  
    'ListboxTop',0, ...  
    'Position',[0.3188 0.055028 0.6261 0.03795], ...  
    'Style','edit', ...  
    'Tag','EditText1');
```

```
% OFFSET EDIT
```

```
offset_call=[  
    'h_offset=findobj(''Tag'', ''EditText2'');'...  
    'h_accept=findobj(''Tag'', ''Pushbutton9'');'...  
    'h_typedepedit=findobj(''Tag'', ''EditText5'');'...  
    'h_unitdepedit=findobj(''Tag'', ''EditText3'');'...  
    'h_refactedit=findobj(''Tag'', ''EditText7'');'...  
    'h_actlist=findobj(''Tag'', ''Listbox1'');'...  
    'h_typelist=findobj(''Tag'', ''Listbox2'');'...  
    'h_unitlist=findobj(''Tag'', ''Listbox3'');'...  
    'h_popup2=findobj(''Tag'', ''popup2'');'...  
    'popoption=get(h_popup2, ''Value'');'...  
    'acttype=get(h_accept, ''userdata'');'...  
    'typedepstr=get(h_typedepedit, ''string'');'...  
    'acttype=get(h_accept, ''userdata'');'...  
    'unitdepstr=get(h_unitdepedit, ''string'');'...  
    'refactstr=get(h_refactedit, ''string'');'...  
    'offsetstr=get(h_offset, ''string'');'...  
    'offsetparam=offsetstr;'...  
    'refact=str2num(refactstr);'...  
    'typedep=str2num(typedepstr);'...  
    'unitdep=str2num(unitdepstr);'...  
    'if popoption==1 | popoption==2 | popoption==3,'...  
    'manualdur=0;'...  
    'acttype=gettuned(reslimits, acttype, refact, typedep, unitdep,  
offsetparam, popoption);'...  
    'set(h_accept, ''userdata'', acttype);'...  
    'unitval=get(h_unitlist, ''Value'');'...  
    'typeval=get(h_typelist, ''Value'');'...  
    'actval=get(h_actlist, ''Value'');'...  
    'if unitval < reslimits(typeval),'...  
    'set(h_unitlist, ''Value'', unitval+1);'...  
    'set(h_unitdepedit, ''String'', unitval+1);'...  
    'elseif actval < numact,'...  
    'set(h_actlist, ''Value'', actval+1);'...  
    'set(h_refactedit, ''String'', actval+1);'...  
    'set(h_unitlist, ''Value'', 1);'...  
    'set(h_unitdepedit, ''String'', 1);'...  
    'else,'...  
    'set(h_actlist, ''Value'', 1);'...  
    'set(h_refactedit, ''String'', 1);'...  
    'end;'...  
    'set(h_offset, ''string'', '');'...  
    'elseif popoption==4,'...  
    'if exist(''actdur'')==1 &  
length(actdur)>numact, actdur=[];, end;'...  
end;
```

```

'actdur(refact)=eval(offsetparam);'...
'manualdur=1;'...
'actval=get(h_actlist,'Value');'...
'if actval < numact,'...
'set(h_actlist,'Value',actval+1);'...
'set(h_refactedit,'String',actval+1);'...
'else,'...
'set(h_actlist,'Value',1);'...
'set(h_refactedit,'String',1);'...
'end;'...
'set(h_offset,'string','');'...
'end;'
];

h1 = uicontrol('Parent',h0, ...
'Units','normalized', ...
'callback', offset_call,...
'BackgroundColor',[1 1 1], ...
'HorizontalAlignment','left', ...
'ListboxTop',0, ...
'Position',[0.3188 0.11195 0.232 0.03795], ...
'Style','Edit', ...
'Tag','EditText2');

% SET UNIT DEPENDENT EDIT

h1 = uicontrol('Parent',h0, ...
'Units','normalized', ...
'BackgroundColor',[1 1 1], ...
'HorizontalAlignment','left', ...
'string',1,...
'ListboxTop',0, ...
'Position',mat6, ...
'Style','edit', ...
'Tag','EditText3');

% SET UNIT DRIVER EDIT

h1 = uicontrol('Parent',h0, ...
'Units','normalized', ...
'BackgroundColor',[1 1 1], ...
'HorizontalAlignment','left', ...
'ListboxTop',0, ...
'Position',[0.5942028 0.2087286 0.347826 0.0379506], ...
'Style','edit', ...
'Tag','EditText4');

% SET TYPE DEPEND EDIT

h1 = uicontrol('Parent',h0, ...
'Units','normalized', ...
'BackgroundColor',[1 1 1], ...
'HorizontalAlignment','left', ...
'ListboxTop',0, ...
'string',1,...
'Position',mat7, ...
'Style','edit', ...

```

```

        'Tag','EditText5');

% SET TYPE DRIVER EDIT

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[1 1 1], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[0.59420289 0.2846299 0.347826 0.03795066], ...
    'Style','edit', ...
    'Tag','EditText6');

REFERENCE ACTIVITY EDIT

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[1 1 1], ...
    'HorizontalAlignment','left', ...
    'string',1,...
    'ListboxTop',0, ...
    'Position',[0.160869565 0.37950664 0.7797 0.03795], ...
    'Style','edit', ...
    'Tag','EditText7');

% SET TYPE DRIVER PUSHBUTTON

settypedrv_call=[
    'h_typedrvlist=findobj(''Tag'', ''Listbox2'');'...
    'val=get(h_typedrvlist, ''Value'');'...
    'val=num2str(val);'...
    'h_typedrvedit=findobj(''Tag'', ''EditText6'');'...
    'set(h_typedrvedit, ''string'', val);'
];

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'callback', settypedrv_call,...
    'ListboxTop',0, ...
    'Position',[0.30289 0.5692599 0.086956 0.0569259], ...
    'String','Set Driver', ...
    'Tag','Pushbutton1');

% SET TYPE DEPEND PUSHBUTTON

settypedep_call=[
    'h_typedrvlist=findobj(''Tag'', ''Listbox2'');'...
    'val=get(h_typedrvlist, ''Value'');'...
    'val=num2str(val);'...
    'h_typedepedit=findobj(''Tag'', ''EditText5'');'...
    'set(h_typedepedit, ''string'', val);'
];

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...

```

```

        'callback',settypedep_call,...
        'ListboxTop',0, ...
        'Position',[0.3028985 0.4743833 0.086956 0.05692599], ...
        'String','Set Depen', ...
        'Tag','Pushbutton2');

% SET UNIT DRIVER PUSHBUTTON

setunitdrv_call=[
    'h_unitlist=findobj(''Tag'', ''Listbox3'');'...
    'val=get(h_unitlist, ''Value'');'...
    'val=num2str(val);'...
    'h_unitdrvedit=findobj(''Tag'', ''EditText4'');'...
    'set(h_unitdrvedit, ''string'',val);'
];

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'callback', setunitdrv_call,...
    'ListboxTop',0, ...
    'Position',[0.65072 0.5692599 0.0869565 0.05692599], ...
    'String','Set Driver', ...
    'Tag','Pushbutton3');

% SET UNIT DEPENDENT PUSHBUTTON

setunitdep_call=[
    'h_unitlist=findobj(''Tag'', ''Listbox3'');'...
    'val=get(h_unitlist, ''Value'');'...
    'val=num2str(val);'...
    'h_unitdepedit=findobj(''Tag'', ''EditText3'');'...
    'set(h_unitdepedit, ''string'',val);'
];

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'callback', setunitdep_call,...
    'ListboxTop',0, ...
    'Position',[0.6507246 0.4743833 0.0869565 0.05692599], ...
    'String','Set Depen', ...
    'Tag','Pushbutton4');

% EXIT PUSHBUTTON

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'Callback','close', ...
    'ListboxTop',0, ...
    'Position',[ 0.840579710144927 0.113851992409867 0.101449275362319
0.0379506641366224 ], ...
    'String','Exit', ...
    'Tag','Pushbutton5');

% ARBITRARY PUSHBUTTON

```

```

arbitrary_call=[
    'h_refactedit=findobj(''Tag'', ''EditText7'');'...
    'h_typedepedit=findobj(''Tag'', ''EditText5'');'...
    'h_unitdepedit=findobj(''Tag'', ''EditText3'');'...
    'h_funcedit=findobj(''Tag'', ''EditText1'');'...
    'h_accept=findobj(''Tag'', ''Pushbutton9'');'...
    'acttype=get(h_accept, ''userdata'');'...
    'typedepstr=get(h_typedepedit, ''string'');'...
    'unitdepstr=get(h_unitdepedit, ''string'');'...
    'refactstr=get(h_refactedit, ''string'');'...
    'funcstr=get(h_funcedit, ''string'');'...
    'refact=str2num(refactstr);'...
    'typedep=str2num(typedepstr);'...
    'unitdep=str2num(unitdepstr);'...
    'objselect=1;'...
    'acttype=getarbitrary( reslimits, acttype, refact, typedep,
unitdep, funcstr, objselect);'...
    'set(h_accept, ''userdata'', acttype);'
];

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'ListboxTop',0, ...
    'Position',[ 0.579710144927536 0.113851992409867 0.101449275362319
0.0379506641366224 ], ...
    'Tag','Pushbutton6');

% ADD PUSHBUTTON

add_call=[
    'h_typedrvedit=findobj(''Tag'', ''EditText6'');'...
    'h_typedepedit=findobj(''Tag'', ''EditText5'');'...
    'h_unitdrvedit=findobj(''Tag'', ''EditText4'');'...
    'h_unitdepedit=findobj(''Tag'', ''EditText3'');'...
    'h_refactedit=findobj(''Tag'', ''EditText7'');'...
    'h_funcedit=findobj(''Tag'', ''EditText1'');'...
    'h_accept=findobj(''Tag'', ''Pushbutton9'');'...
    'acttype=get(h_accept, ''userdata'');'...
    'typedrvstr=get(h_typedrvedit, ''string'');'...
    'typedepstr=get(h_typedepedit, ''string'');'...
    'unitdrvstr=get(h_unitdrvedit, ''string'');'...
    'unitdepstr=get(h_unitdepedit, ''string'');'...
    'refactstr=get(h_refactedit, ''string'');'...
    'funcstr=get(h_funcedit, ''string'');'...
    'refact=str2num(refactstr);'...
    'typedrv=str2num(typedrvstr);'...
    'typedep=str2num(typedepstr);'...
    'unitdrv=str2num(unitdrvstr);'...
    'unitdep=str2num(unitdepstr);'...
    'mode=''add'';'...
    'acttype=gettime( reslimits, acttype, refact, typedrv, unitdrv,
typedep, unitdep, funcstr, mode);'...
    'set(h_accept, ''userdata'', acttype);'
];

```

```

percent_call=[
    'h_typedrvedit=findobj(''Tag'', ''EditText6'');'...
    'h_typedepedit=findobj(''Tag'', ''EditText5'');'...
    'h_unitdrvedit=findobj(''Tag'', ''EditText4'');'...
    'h_unitdepedit=findobj(''Tag'', ''EditText3'');'...
    'h_refactedit=findobj(''Tag'', ''EditText7'');'...
    'h_funcedit=findobj(''Tag'', ''EditText1'');'...
    'h_accept=findobj(''Tag'', ''Pushbutton9'');'...
    'acttype=get(h_accept, ''userdata'');'...
    'typedrvstr=get(h_typedrvedit, ''string'');'...
    'typedepstr=get(h_typedepedit, ''string'');'...
    'unitdrvstr=get(h_unitdrvedit, ''string'');'...
    'unitdepstr=get(h_unitdepedit, ''string'');'...
    'refactstr=get(h_refactedit, ''string'');'...
    'funcstr=get(h_funcedit, ''string'');'...
    'refact=str2num(refactstr);'...
    'typedrv=str2num(typedrvstr);'...
    'typedep=str2num(typedepstr);'...
    'unitdrv=str2num(unitdrvstr);'...
    'unitdep=str2num(unitdepstr);'...
    'mode='''percent'';'...
    'acttype=gettime( reslimits, acttype, refact, typedrv, unitdrv,
typedep, unitdep, funcstr, mode);'...
    'set(h_accept, ''userdata'', acttype);'
];

accept_call=[
    'h_accept=findobj(''Tag'', ''Pushbutton9'');'...
    'h_exit=findobj(''Tag'', ''Pushbutton5'');'...
    'set(h_exit, ''userdata'', 1);'...
    'acttype_pure=acttype;'...
    'set(h_accept, ''string'', ''Done'');'
];

h1 = uicontrol('Parent', h0, ...
    'callback', accept_call, ...
    'Units', 'normalized', ...
    'ListboxTop', 0, ...
    'Position', [ 0.710144927536232 0.113851992409867 0.101449275362319
0.0379506641366224 ], ...
    'string', 'Accept All', ...
    'Tag', 'Pushbutton9');

: STATIC TEXTS

h1 = uicontrol('Parent', h0, ...
    'Units', 'normalized', ...
    'BackgroundColor', [0.75294 0.75294 0.75294], ...
    'FontWeight', 'bold', ...
    'HorizontalAlignment', 'left', ...
    'ListboxTop', 0, ...
    'Position', mat8, ...
    'String', 'Dependents', ...
    'Style', 'text', ...
    'Tag', 'StaticText1');
h1 = uicontrol('Parent', h0, ...
    'Units', 'normalized', ...

```

```

        'BackgroundColor',[0.75294 0.75294 0.75294], ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',mat9, ...
        'String','Drivers', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[0.07536 0.28273 0.086956 0.03795], ...
    'String','Set Type(s):', ...
    'Style','text', ...
    'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[0.072463768 0.208728 0.086956 0.0379506], ...
    'String','Set Unit(s):', ...
    'Style','text', ...
    'Tag','StaticText4');

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[0.0464 0.11195 0.2463 0.03795], ...
    'String','Varying Resource Time Req.|Desired Resource Start
Time|Resource Interval Availability|Fixed Activity Duration', ...
    'Style','popupmenu', ...
    'Tag','popup2');

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'ListboxTop',0, ...
    'Position',[0.0464 0.0550 0.2463 0.0380], ...
    'HorizontalAlignment','right',...
    'String','Additive Time Dependency|Percentual Time
Dependency|Arbitrary Time Dependency|Preference|Cost', ...
    'Style','popupmenu', ...
    'Tag','popup1');

h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'FontWeight','bold', ...
    'ListboxTop',0, ...
    'Position',[0.0579710144 0.9354838 0.1884057 0.0379506], ...

```

```

        'String','Activities', ...
        'Style','text', ...
        'Tag','StaticText7');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'FontWeight','bold', ...
    'ListboxTop',0, ...
    'Position',[0.4043478 0.92979 0.18840579 0.03795], ...
    'String','Resource Types', ...
    'Style','text', ...
    'Tag','StaticText8');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'FontWeight','bold', ...
    'ListboxTop',0, ...
    'Position',[0.753623188 0.929791 0.18840579 0.03795], ...
    'String','Resource Units', ...
    'Style','text', ...
    'Tag','StaticText9');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[0.50724637 0.2846299 0.086956 0.03795066], ...
    'String','Set Type(s):', ...
    'Style','text', ...
    'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294117 0.75294117 0.75294117], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[0.50724637 0.208728 0.086956 0.0379506], ...
    'String','Set Unit(s):', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','normalized', ...
    'BackgroundColor',[0.75294 0.75294 0.75294], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[0.02898 0.38140417 0.131884 0.034155], ...
    'String','Reference Activity:', ...
    'Style','text', ...
    'Tag','StaticText10');

h_typedlist=findobj('Tag','Listbox2');
set(h_typedlist,'userdata',reslimits);

h_actlist=findobj('Tag','Listbox1');
set(h_actlist,'userdata',numact);

h_exit=findobj('Tag','Pushbutton5');

h_accept=findobj('Tag','Pushbutton9');

```

```

if exist('acttype')
    set(h_accept, 'userdata', acttype);
end

while ~length(get(h_exit, 'userdata'))
    drawnow
end

h_accept=findobj('Tag','Pushbutton9');
acttype=get(h_accept, 'userdata')

uiwait(h0);

if nargout > 0, fig = h0; end

function acttype=gettime( reslimits, acttype, refact, typedrv,
unitdrv, typedep, unitdep, funcstr, mode)

jump=0;
switch mode
case {'add'}

    for act=1:length(refact)
        for tdep=1:length(typedep)
            for udep=1:length(unitdep)
                % jump=0;
                if isempty(typedrv) | isempty(unitdrv)
                    unitdrv=nan;
                    typedrv=nan;
                end; %end if isempty(typedrv) | isempty(unitdrv)

                for tdrv=1:length(typedrv)
                    for udrv=1:length(unitdrv)
                        % jump=jump+1;
                        if exist('acttype')==1
                            dummy=eval('size(acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).func,2)+1','l');
                        else
                            dummy=1;
                        end

                        switch mode
                            case {1}

                                conditionleft=unitdep(udep);

                                conditionright=reslimits(typedep(tdep));
                                if conditionleft<=conditionright

                                    acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).func{dummy}=sprintf('acttype(%.0f,%.
%.0f).unit(%.0f).assigned*(acttype(%.0f,%.0f).unit(%.0f).tuned +
%s)', refact(act), typedrv(tdrv), unitdrv(udrv), refact(act), typedrv(tdrv)
, unitdrv(udrv), funcstr);

                                    else
                                        break
                                    end %end if condition

                                case {2}

```

```

conditionleft=unitdep(udep);

conditionright=reslimits(typedep(tdep));
    if conditionleft<=conditionright
        acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).func{dummy}=sprintf('acttype(%.0f,%.0f).unit(%.0f).assigned*(acttype(%.0f,%.0f).unit(%.0f).tuned*(%s+1))',refact(act),typedrv(tdrv),unitdrv(udrv),refact(act),typedrv(tdrv),unitdrv(udrv),funcstr);
        else
            break
        end % end if condition

        case {4}
            if exist('acttype')==1

                prefind=eval('size(acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).pref,2)+1','1');
                else
                    prefind=1;
                end

                conditionleft=unitdep(udep);

                conditionright=reslimits(typedep(tdep));
                    if conditionleft<=conditionright

                        if isfinite(typedrv) &
isfinite(unitdrv)
                            acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).pref{prefind}=sprintf('acttype(%.0f,%.0f).unit(%.0f).assigned*(%s)',refact(act),typedrv(tdrv),unitdrv(udrv),funcstr);
                            else
                                acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).pref{prefind}=sprintf('%s',funcstr)
                                ;
                            end

                            else
                                break
                            end % end if condition

                                case {5}
                                    if exist('acttype')==1

                                        costind=eval('size(acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).cost,2)+1','1');
                                        else
                                            costind=1;
                                        end

                                        conditionleft=unitdep(udep);

                                        conditionright=reslimits(typedep(tdep));
                                            if conditionleft<=conditionright

```

```

                                if isfinite(typedrv) &
isfinite(unitdrv)
                                acttype(refact(act),
                                typedep(tdep)).unit(unitdep(udep)).cost{costind}=sprintf('acttype(%f',
                                ,%f).unit(%f).assigned*('%s)',refact(act),typedrv(tdrv),unitdrv(udr
                                v),funcstr);
                                else
                                acttype(refact(act),
                                typedep(tdep)).unit(unitdep(udep)).cost{costind}=sprintf('%s',funcstr)
                                ;
                                end
                                else
                                break
                                end ~ end if condition
                                end ~ end switch

```

```

                                end
                                end
                                end
                                end
                                end

```

```

function acttype=gettuned(reslimits, acttype, refactor, typedep,
unitdep, offsetparam,popoption)

```

```

if popoption==1

```

```

    for act=1:length(refact)
        for tdep=1:length(typedep)
            for udep=1:length(unitdep)
                conditionleft=unitdep(udep);
                conditionright=reslimits(typedep(tdep));
                if conditionleft<=conditionright
                    acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).tuned=eval(offsetparam);
                    else
                        break
                    end
                end
            end
        end
    end
end

```

```

elseif popoption==2

```

```

    for act=1:length(refact)
        for tdep=1:length(typedep)
            for udep=1:length(unitdep)

```

```

        conditionleft=unitdep(udep);
        conditionright=reslimits(typedep(tdep));
        if conditionleft<=conditionright
            acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).start=sprintf('(1/(1-(time -
ts)^2))',offsetparam);
            acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).start=sprintf('desstart(%s,time)',o
ffsetparam);

        else
            break
        end
    end
end
end
elseif popoption==3
    for act=1:length(refact)
        for tdep=1:length(typedep)
            for udep=1:length(unitdep)
                conditionleft=unitdep(udep);
                conditionright=reslimits(typedep(tdep));
                if conditionleft<=conditionright

                    acttype(refact(act),
typedep(tdep)).unit(unitdep(udep)).start=sprintf('interval(%s,time)',o
ffsetparam);
                else
                    break
                end
            end
        end
    end
end
end
end
endif popoption

```

```

function [wind]=interval(fromto,time)

if time < fromto(1)
    wind=0;
elseif time >= fromto(1) & time <= fromto(2)
    wind=1;
elseif time > fromto(2)
    wind=0;
end

```

```

% master scheduling file
%..... Start and Initiate the variables
%.....
clear;
acttype=acttype_pure;
[acttype]=setassigned(acttype, numact, numres, reslimits);
[actneeds,pred,reslimits,actrestime]= readfile;
[actdur]=duration(actrestime, actneeds, reslimits);

if exist('manualdur')==1 & manualdur==1
    if length(actdur)<numact
        h_oops=errorldg('Some of the Activity Durations are not
Specified!','I am Crashing...!');
    end
end

if exist('manualdur')==0
    manualdur=0;
end

if manualdur==0
[actdur]=duration(acttype, actneeds, numact, numres);
end

if exist('utility')==0
    utility=0;
end

    if exist('optchoice')==0 | isempty(optchoice)==1
        errorldg('You did not specify which objective to optimize. I am
going into default mode.','Read my User Manual!!!!');
        optchoice=1;
    end

mindur=actdur;
[est,lst]=cpm(actdur,pred);
dynpred=pred;
%weight=input('Enter the importance of resource leveling (a
nonnegative number)');
%choice=menu('Project Scheduling','Don't Balance Resources', 'Balance
Resources','Don't Balance, but assign resources','Balance Resources
and Assign Resource Units');

%Choices 1-Schedule Only, 2-Balance Only, 3-Map Only, 4 - Balance and
Map

if exist('w')==0
    w=0;
end

if exist('choice')==0
    choice=1;
end

if choice==1 | choice==3
    weight=0;
else
    weight=w;
end

```

```

end

tic;
multiplier=weight;

scheduled=[];
newlyadded=[];
finished=[];
inprogress=[];
time=0;
direction=1;
abscis=[];
usage=[];
[numsucc]=children(pred);
assigned=zeros(size(actrestime));
assigned=sparse(zeros(numact,sum(reslimits)));
.....

hf_wait=waitbar(0,'Please wait, I am steaming...');
while size(scheduled,2)<length(actdur)
    for b=1:l
        waitbar(size(scheduled,2)/length(actdur));
        if isempty(scheduled)==0

            time=min(scheduled(3,(find(scheduled(3,:)>time))));

            finished=scheduled(1,find(scheduled(3,:)==time));
            newlyadded=[];
            for z=1:length(finished)
                inprogress(inprogress==finished(z))=[];
            end

            for i=1:length(finished)
                if isempty(finished)==0
                    dynpred(find(dynpred==finished(i)))=0;
                end
            end
        end
    end

    [cand]=candidates(dynpred);

    if isempty(cand)==0
        [prior]=floatweight(cand,lst,actdur,time);
    ..

    if direction==1
        scheduler;
        if isempty(scheduled)==0
            % multiplier=multiplier*(length(lst)-
            length(scheduled(1,:)))/length(lst);
            multiplier=multiplier*(sum(mindur)-
            sum(mindur(scheduled(1,:))))/sum(mindur);

```

```

        end %if isempty
        if x(end)~=0
            %disp(['reached the resource peak at time ' num2str(time) ',
going down now...'])
            multiplier=weight;
            schedulerdown;
            multiplier=multiplier/2;
            direction=0;
            end
    else
        schedulerdown
        multiplier=multiplier/2;
    end

    %.....
    newlyadded=cand(find(x));
    end %if isempty(cand)==0

    if isempty(newlyadded)==0;
        % Insert 'rescheduler' here to schedule resource units
        if choice==3 | choice==4
            rescheduler;

        if manualdur==0 & optchoice~=2 & optchoice~=3
            [actdur]=updateactdur(actdur, newlyadded, acttype, reslimits);
        end %end if manualdur==0
        end
        [scheduled]=chart(time,newlyadded,scheduled,actdur);
        inprogress=[inprogress newlyadded'];
        dynpred(newlyadded,:)=nan;
    end

    abscis=[abscis time]; % needed for resource loading graph
    usage=[usage sum(actneeds(inprogress,:),1)']; % needed for resource
loading graph

end %while
close(hf_wait);

if optchoice~=2 & optchoice~=3
    % scheduled % un-remark this for scheduled to be displayed
end

```

```

function
[c]=objective(prior,actneeds,reslimits,cand,actdur,multiplier,numsucc,
mindur)

durweight=mindur(cand)/max(mindur);
maxobj=prior.*numsucc(cand).*durweight; %numsucc(cand) is already
scaled
minobj=actneeds(cand,:);
for s=1:length(reslimits)
    minobj(:,s)=minobj(:,s)/reslimits(s);
end
%minobj=multiplier*(1-sum(minobj,2))';
minobj=sum(minobj,2)';
minobj=minobj/max(minobj);
minobj=multiplier*(1-minobj);

append = 2*sum(actdur);
c=fix([(-minobj-maxobj) append]*10000);

```

```

function [c]=objectivedown(prior,actneeds,reslimits,cand,actdur,
multiplier,numsucc, mindur)

durweight=mindur(cand)/max(mindur);
maxobj=prior.*numsucc(cand).*durweight;
minobj=actneeds(cand,:);
for s=1:length(reslimits)
    minobj(:,s)=minobj(:,s)/reslimits(s);
end

minobj=sum(minobj,2)';
minobj=minobj/max(minobj);
minobj=multiplier*(minobj);

append = 2*sum(actdur);
c=fix([(-minobj-maxobj) append]*10000);

```

```

function fig = promap()
% This is the machine-generated representation of a Handle Graphics
object
% and its children. Note that handle values may change when these
objects
% are re-created. This may cause problems with any callbacks written
to
% depend on the value of the handle at the time the object was saved.

% To reopen this object, just type the name of the M-file at the
MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.

```

```

dynamo;
clear
load promap

```

```

h0 = figure('Color',[0.8 0.8 0.8], ...
    'Colormap',mat0, ...
    'MenuBar','none', ...
    'Name','PROMAP: Project-Resource Mapper',...
    'NumberTitle','off', ...
    'PointerShapeCDData',mat1, ...
    'Position',[240 316 300 1], ...
    'Tag','Fig1');
h1 = uimenu('Parent',h0, ...
    'Label','&Project', ...
    'Tag','project');

```

```

newproject_call=[
    'getdata;'\...
    'if isempty(numact)==0 & isempty(numres)==0 &
isempty(reslimits)==0,'\...
    'if ~exist('acttype')'\...
    'getfunctions(numact, numres, reslimits);\...
    'else;\...
    'getfunctions(numact, numres, reslimits,acttype);\...
    'end;\...
    'end;'\...
    ];

```

```

h2 = uimenu('Parent',h1, ...
    'callback',newproject_call,...
    'Label','&New Project', ...
    'Tag','new');

```

```

open_call=[
    '[nam,pat]=uigetfile(' '*.mat' ', 'Open Existing Project');'\...
    'if nam~=0,'\...
    'nam=strcat(pat,nam);\...
    'load(nam);\...
    'acttype=acttype_pure;\...
    'end;'\...
    ];

```

```

h2 = uimenu('Parent',h1, ...
    'callback',open_call,...
    'Label','&Open Project', ...

```

```

    'Tag','open');

save_call=[
    '[namput,patput]=uinputfile(''projectdata.mat'', ''Save Project
Data'');'...
    'if namput~=0 & exist(''acttype_pure'')==1,'...
    'namput=strcat(patput, namput);'...
    'save(eval(''namput''),'acttype_pure',
    'actneeds','numact','numres','reslimits','pred');'...
    'end;'
    ];

h2 = uimenu('Parent',h1, ...
    'callback',save_call,...
    'Label','&Save Project', ...
    'Tag','save');

h2 = uimenu('Parent',h1, ...
    'callback','close',...
    'Label','&Close', ...
    'Separator','on', ...
    'Tag','finish');

h1 = uimenu('Parent',h0, ...
    'Label','&Run', ...
    'Tag','run');

schedule_call=[
    'h_balandmap=findobj(''Tag'', ''balandmap'');'...
    'h_balonly=findobj(''Tag'', ''balanceonly'');'...
    'h_maponly=findobj(''Tag'', ''maponly'');'...
    'bm=get(h_balandmap, ''checked'');'...
    'bo=get(h_balonly, ''checked'');'...
    'mo=get(h_maponly, ''checked'');'...
    'if strcmp(bm, ''off'')==1 & strcmp(bo, ''off'')==1 &
strcmp(mo, ''off'')==1,'...
    'choice=1;'...
    'end;'...
    'master;'
    ];

h2 = uimenu('Parent',h1, ...
    'callback', schedule_call,...
    'Label','Sche&dule', ...
    'Tag','schedule');

optim_call=[
    'choicesstring={'Time
Effectiveness','Preferences','Costs','Resource
Availability','Composite Utility Function'};'...
    '[optchoice,uredu]=listdlg(''Name'', ''Select
Objective'', ''PromptString'', ''Map resources according

```

```

to...','SelectionMode','Single','ListString',choicesstring,'Li
stSize',[160,80]);',...
    'if optchoice==5,utility=inputdlg('Enter the Composite Utility
Function, U(timedep,pref,cost,starttime):','Composite Utility
Function',1);,end;
    ];

h2= uimenu('Parent',h1,...
    'callback',optim_call,...
    'Label','&Optimizing Objectives',...
    'Tag','optimize');

ballevel_call=[
    'w=inputdlg('Enter the Resource Centralizing Priority Weight',
    'Balancing Priority',1,{'0'});',...
    'if isempty(w)==1,',...
    'w=0;',...
    'else,',...
    'w=str2num(char(w));',...
    'end;
    ];

h2 = uimenu('Parent',h1, ...
    'callback',ballevel_call,...
    'Label','Set Centralizing &Importance &Level', ...
    'Tag','level');

centrtype_call=[
    'numres=length(reslimits);',...
    'for j=1:numres, restypestr(j)={sprintf('Resource Type
%.0f',j)};end;',...
    '[typeselect,izbor]=listdlg('PromptString','Selecet Resource
Types','ListString',restypestr,'ListSize',[160,160]);'
    ];

h2=uimenu('Parent',h1,...
    'callback',centrtype_call,...
    'Label','Resource &Types to Centralize',...
    'Tag','choosetypes');

balandmap_call=[
    'h_balandmap=findobj('Tag','balandmap');',...
    'h_balonly=findobj('Tag','balanceonly');',...
    'h_maponly=findobj('Tag','maponly');',...
    'h_level=findobj('Tag','level');',...
    'bmcheck=get(h_balandmap,'checked');',...
    'if strcmp(bmcheck,'on')==1,',...
    'choice=1;',...
    'set(h_balandmap,'checked','off');',...
    'set(h_level,'enable','off');',...
    'else,',...
    'set(h_balandmap,'checked','on');',...
    'set(h_level,'enable','on');',...
    'choice=4;',...
    'set(h_balonly,'checked','off');',...
    'set(h_maponly,'checked','off');',...

```

```

        'end;';
];

h2 = uimenu('Parent',h1, ...
    'callback', balandmap_call,...
    'Label','Map &and Centralize', ...
    'Separator','on', ...
    'Tag','balandmap');

balanceonly_call=[
    'h_balandmap=findobj(''Tag'', ''balandmap'');'...
    'h_balonly=findobj(''Tag'', ''balanceonly'');'...
    'h_maponly=findobj(''Tag'', ''maponly'');'...
    'h_level=findobj(''Tag'', ''level'');'...
    'bmcheck=get(h_balonly, ''checked'');'...
    'if strcmp(bmcheck, ''on'')==1, '...
    'choice=1; '...
    'set(h_balonly, ''checked'', ''off'');'...
    'set(h_level, ''enable'', ''off'');'...
    'else, '...
    'set(h_balonly, ''checked'', ''on'');'...
    'set(h_level, ''enable'', ''on'');'...
    'choice=2; '...
    'set(h_balandmap, ''checked'', ''off'');'...
    'set(h_maponly, ''checked'', ''off'');'...
    'end;';
];

h2 = uimenu('Parent',h1, ...
    'callback',balanceonly_call,...
    'Label','&Centralize Only', ...
    'Tag','balanceonly');

maponly_call=[
    'h_balandmap=findobj(''Tag'', ''balandmap'');'...
    'h_balonly=findobj(''Tag'', ''balanceonly'');'...
    'h_maponly=findobj(''Tag'', ''maponly'');'...
    'h_level=findobj(''Tag'', ''level'');'...
    'bmcheck=get(h_maponly, ''checked'');'...
    'if strcmp(bmcheck, ''on'')==1, '...
    'choice=1; '...
    'set(h_maponly, ''checked'', ''off'');'...
    'set(h_level, ''enable'', ''off'');'...
    'else, '...
    'set(h_maponly, ''checked'', ''on'');'...
    'set(h_level, ''enable'', ''on'');'...
    'choice=3; '...
    'set(h_balandmap, ''checked'', ''off'');'...
    'set(h_balonly, ''checked'', ''off'');'...
    'end;';
];

```

```

h2 = uimenu('Parent',h1, ...
    'callback',maponly_call,...
    'Label','&Map Only', ...
    'Tag','maponly');

h1 = uimenu('Parent',h0, ...
    'Label','&Graph', ...
    'Tag','graph');

gantt_call=[
    'figure;';...
    'for r=1:length(actdur),'...
    'data(1,r)=scheduled(2,find(scheduled(1,:)==r));'...
    'end;';...
    'data(2,:)=actdur;';...
    'barh(data','stack');';...
    'colormap([1 1 1;0 0 0]);';...
    'set(gca,'color','white');';...
    'title(['Project is completed at t = '
num2str(scheduled(3,end))]);';...
    'xlabel('Time');';...
    'ylabel('Activities');';
    ];

h2 = uimenu('Parent',h1, ...
    'callback',gantt_call,...
    'Label','Ga&ntt', ...
    'Tag','gantt');

loading_call=[
    'abscis=[abscis scheduled(end)];';...
    'usage=[usage usage(:,end)];';...
    'for v=1:length(reslimits),'...
    'figure;';...
    'stairs(abscis,usage(v,:));';...
    'yticks=1:reslimits(v);';...
    'set(gca,'yTick',yticks);';...
    'axis([0 scheduled(end) 0 reslimits(v)+1]);';...
    'title(sprintf('Resource Type %.0f Loading Graph', v));';...
    'xlabel('Time');';...
    'ylabel('Resource Units');';...
    'end;';
    ];

h2 = uimenu('Parent',h1, ...
    'callback',loading_call,...
    'Label','&Resource Loading', ...
    'Tag','loading');

unitmapping_call=[
    'if choice==3 | choice==4,'...
    'for restype=1:length(reslimits),'...

```

```

'figure;'...
'grid;'...
'xticks=1:reslimits(restype);'...
'yticks=1:numact;'...
'axis([0 reslimits(restype)+1 0 size(actneeds,1)+1]);'...
'set(gca,'XTick',xticks);'...
'set(gca,'YTick',yticks);'...
'hold;'...
'for nact=1:numact,'...
'vect=find([acttype(nact,restype).unit(:).assigned]);'...
'if ~isempty(vect)'...
'plot(vect,nact, 'ro');'...
'end;'...
'end;'...
'title(sprintf('Mapping Resource Type %.0f Units to Project
Activities', restype));'...
xlabel(sprintf('Resource Type %.0f Units',restype));'...
ylabel('Project Activities');'...
'hold off;'...
'end;'...
'end;'...
];

h2 = uimenu('Parent',h1, ...
'callback',unitmapping_call,...
'Label','&Unit Mapping', ...
'Tag','unitmapping');

util_call=[
'if choice==3 | choice==4,'...
'for restype=1:length(reslimits),'...
'figure;'...
'xticks=1:reslimits(restype);'...
'axis([0 reslimits(restype)+1 0 1]);'...
'set(gca,'XTick',xticks);'...
'hold;'...
'maxunittime=zeros(1,reslimits(restype));'...
'minunittime=zeros(1,reslimits(restype));'...
'for nunit=1:reslimits(restype),'...
'for nact=1:numact,'...

'maxunittime(nunit)=maxunittime(nunit)+(acttype(nact,restype).unit(nun
it).assigned)*actdur(nact);'...
'if isfinite(acttype(nact,restype).unit(nunit).tuned)'...

'minunittime(nunit)=minunittime(nunit)+(acttype(nact,restype).unit(nun
it).assigned)*(acttype(nact,restype).unit(nunit).tuned);'...
'else'...
'minunittime(nunit)=minunittime(nunit)+0;'...
'end;'...
'end;'...
'maxunittime(nunit)= maxunittime(nunit)/scheduled(3,end);'...
'minunittime(nunit)= minunittime(nunit)/scheduled(3,end);'...
'end;'...
'bar(maxunittime,'r');'...
'bar(minunittime, 'b');'...

```

```

        'title(sprintf(''Time Percentage of Resource Type %.0f Units
Engagement vs. Total Project Duration'', restype));'...
        'xlabel(sprintf(''Resource Type %.0f Units'',restype));'...
        'ylabel(''Percentage of Total Project Duration'');'...
        'hold off;'...
        'end;'...
        'end;'
    ];

h2 = uimenu('Parent',h1, ...
    'callback',util_call,...
    'Label','Unit Utili&zation', ...
    'Tag','utilization');

unitcost_call = [
    'warning off;'...
    'if (optchoice==3 | optchoice==5) & (choice==3 | choice==4),'...
    'for restype=1:length(reslimits),'...
    'figure;'...
    'xticks=1:reslimits(restype);'...
    'set(gca, 'XTick',xticks);'...
    'hold;'...
    'unitcost=zeros(1,reslimits(restype));'...
    'for nunit=1:reslimits(restype),'...
    'for nact=1:numact,'...
    'if isfinite(acttype(nact,restype).unit(nunit).mastercost),'...

    'unitcost(nunit)=unitcost(nunit)+(acttype(nact,restype).unit(nunit).as
signed)*(acttype(nact,restype).unit(nunit).mastercost);'...
    'else,'...
    'unitcost(nunit)=unitcost(nunit)+0;'...
    'end;'...
    'end;'...
    'end;'...
    'bar(unitcost, 'g');'...
    'title(sprintf(''Project Cost For Type %.0f Resource Units'',
restype));'...
    'xlabel(sprintf(''Resource Type %.0f Units'',restype));'...
    'ylabel(''Total Unit Cost'');'...
    'hold off;'...
    'end;'...
    'end;'...
    'warning on;'
    ];

h2= uimenu('Parent',h1,...
    'callback',unitcost_call,...
    'Label','Total Unit &Costs',...
    'Tag','unit_costs');

if nargout > 0, fig = h0; end

```

```

for restype=1:length(reslimits)
    % restype=q;

[rc, acttype]=resobjective(acttype, restype, reslimits,newlyadded,
time, optchoice, utility);
[resmat, rc]=resmatrix(rc,
newlyadded,reslimits,restype,acttype,inprogress);
[rb,numeq]=resconstraints(newlyadded,restype,reslimits,actneeds);
optPar(13)=numeq;

optPar(1)=0;
optPar(14)=1000000000;
rc=fix(rc*100000); %making sure the objective coefficients are
integers
sol=balas(resmat,rb,rc,optPar)'; % see http://www.ima.mdh.se/tom/
%find(sol)

%***** Calculating the assigned indices

if isempty(sol)
    sol=zeros(1,length(rc));
end %if isempty(rc)

fromsol=1; % formerly fromrc
tosol=0; % formerly torc

for h=1:length(newlyadded)
    tosol=tosol+reslimits(restype);

    for g=1:reslimits(restype)
        acttype(newlyadded(h),restype).unit(g).assigned=sol(fromsol);
        fromsol=fromsol+1;
    end
    fromsol=tosol+1;
end

end %for restype=1:length(reslimits) where q=restype

function
[rb,numeq]=resconstraints(newlyadded,restype,reslimits,actneeds)

%*** Eliminating resource units that are in progress %*****
%***Takes only one row %*****

binprogress=0;

%*****

```

```

***** Satisfying the needs of newlyadded activities *****
***** Number of rows equal to the length(newlyadded) *****

bneeds=actneeds(newlyadded,restype)';

*****

***** Ensuring the uniqueness of the variable assignment *****

bunique=ones(1,reslimits(restype));

*****

rb=[binprogress bneeds bunique];
numeq=length([binprogress bneeds]);

function [resmat,
rc]=resmatrix(rc,newlyadded,reslimits,restype,acttype,inprogress)

***** Calculating the indices
***** Identifying resource units that are in progress *****

resinprogress=zeros(1,reslimits(restype));

for i=1:length(inprogress)
    for j=1:reslimits(restype)
        if acttype(inprogress(i),restype).unit(j).assigned==1 &
isempty(inprogress)~=1
            resinprogress(j)=1;
        end ! end if
    end
end
end

tempres=resinprogress;
for g=1:(length(newlyadded)-1)
    resinprogress=[resinprogress tempres];
end

- Taking care of crisp resource calendar unavailability

for z=1:length(newlyadded)
    for x=1:reslimits(restype)
        if isfield(acttype(newlyadded(z),restype).unit(x),
'masterstart')==1 &
acttype(newlyadded(z),restype).unit(x).masterstart==0
            resinprogress((z-1)*reslimits(restype) + x)=1;
        end
    end
end

```

```

        end
    end

    %-----Eliminating units having 'inf' as objective coefficients-----
    %-----

    resinprogress=resinprogress + isinf(rc);

    if sum(isinf(rc))~=0

end

%-----
%-----
%----- Identifying needs of currently added
activities-----
resneeds=zeros(length(newlyadded),length(newlyadded)*reslimits(restype)
);
from=1;
for i=1:length(newlyadded)
    resneeds(i,from:i*reslimits(restype))=1;
    from=from+reslimits(restype);
end
%-----
%----- Ensuring the unique assignment of resource units
%-----
unique=[];

for i=1:length(newlyadded)
    unique=[unique eye(reslimits(restype))];
end
%-----
%-----

resmat=[resinprogress; resneeds; unique];

function [rc, acttype]=resobjective(acttype, restype,
reslimits,newlyadded, time, optchoice, utility)
%function [rc]=resobjective(actrestime, restype, reslimits,newlyadded)

%----- Calculating the indices -----
%-----

rc=[];

for i=1:length(newlyadded)
    for j=1:reslimits(restype)

        funcheck=isfield(acttype(newlyadded(i),restype).unit(j),
'func');
    end
end

```

```

tunedcheck=isfield(acttype(newlyadded(i),restype).unit(j),'tuned');

    if funcheck==1

        numfuncs=eval('size(acttype(newlyadded(i),restype).unit(j).func,2)');
        maxfun=0;

        if
isempty(acttype(newlyadded(i),restype).unit(j).func)==1
maxfun=eval('min([acttype(newlyadded(i),restype).unit(:).tuned]', 'min'
([acttype(newlyadded(i),1).unit(:).tuned]');
        maxfun=inf;
        end

        for k=1:numfuncs

            funct=eval(acttype(newlyadded(i),restype).unit(j).func{k},inf);

            if funct > maxfun
                maxfun=funct;
            end %end if

        end %end for k=1:numfuncs

        if tunedcheck==0
            acttype(newlyadded(i),restype).unit(j).tuned=maxfun;
        elseif tunedcheck==1 &
isempty(acttype(newlyadded(i),restype).unit(j).tuned)==1
            acttype(newlyadded(i),restype).unit(j).tuned=maxfun;
        elseif tunedcheck==1 &
isempty(acttype(newlyadded(i),restype).unit(j).tuned)==0
            acttype(newlyadded(i),restype).unit(j).tuned=max(maxfun,acttype(newlyad
dded(i),restype).unit(j).tuned);
        end %end if tunedcheck==0

    end % end if funcheck==1

    if funcheck==0
        if tunedcheck==0
            acttype(newlyadded(i),restype).unit(j).tuned=eval('min([acttype(newlyad
dded(i),restype).unit(1:j-
1).tuned]', 'min([acttype(newlyadded(i),1).unit(:).tuned]');
            %defaulting the tuned duration if not specified in any way
            acttype(newlyadded(i),restype).unit(j).tuned=inf;
        elseif tunedcheck==1
            if
isempty(acttype(newlyadded(i),restype).unit(j).tuned)==1
            acttype(newlyadded(i),restype).unit(j).tuned=eval('min([acttype(newlyad
dded(i),restype).unit(1:j-
1).tuned]', 'min([acttype(newlyadded(i),1).unit(:).tuned]');
            acttype(newlyadded(i),restype).unit(j).tuned=inf;
        end % end isempty(acttype)

```

```

        end %end tunedcheck
    end %end funcheck

    if optchoice==1
        rc=[rc acttype(newlyadded(i),restype).unit(j).tuned];
    end

    end %end for j=1:reslimits(restype)
end %end for i=1:length(newlyadded)

if optchoice==2 | optchoice==5

    for i=1:length(newlyadded)
        for j=1:reslimits(restype)

            prefcheck=isfield(acttype(newlyadded(i),restype).unit(j),
'pref');

            if ~isfield(acttype(newlyadded(i),restype).unit(j),
'masterpref') |
isempty(acttype(newlyadded(i),restype).unit(j).masterpref)
                acttype(newlyadded(i),restype).unit(j).masterpref=0;
            end %end if isfield

            if prefcheck==1 &
isempty(acttype(newlyadded(i),restype).unit(j).pref)==0

numprefs=eval('size(acttype(newlyadded(i),restype).unit(j).pref,2)');
                for k=1:numprefs

                    pref=eval(acttype(newlyadded(i),restype).unit(j).pref{k},'0');

acttype(newlyadded(i),restype).unit(j).masterpref=acttype(newlyadded(i),restype).unit(j).masterpref + pref;

                    end %end for k=1:numprefs
                end %end if prefcheck
                if optchoice==2
                    rc=[rc -acttype(newlyadded(i),restype).unit(j).masterpref];
                end %end if optchoice==2 (YES, I need it to be checked
twice!)

            end %end for j=1:reslimits(restype)
        end %end i=1:length(newlyadded)
    end %end optchoice==2

if optchoice==3 | optchoice==5

    for i=1:length(newlyadded)
        for j=1:reslimits(restype)

            costcheck=isfield(acttype(newlyadded(i),restype).unit(j),
'cost');

```

```

        if costcheck==1 &
            (~isfield(acttype(newlyadded(i),restype).unit(j), 'mastercost') |
            isempty(acttype(newlyadded(i),restype).unit(j).mastercost))
                acttype(newlyadded(i),restype).unit(j).mastercost=0; % it
was .mastercost=0;
            elseif costcheck==0 &
            (~isfield(acttype(newlyadded(i),restype).unit(j), 'mastercost') |
            isempty(acttype(newlyadded(i),restype).unit(j).mastercost))
                acttype(newlyadded(i),restype).unit(j).mastercost=inf;
            it was .mastercost=0;
            end %if isfield

        if costcheck==1 &
            isempty(acttype(newlyadded(i),restype).unit(j).cost)==0

        numcosts=eval('size(acttype(newlyadded(i),restype).unit(j).cost,2)');
        for k=1:numcosts

            cost=eval(acttype(newlyadded(i),restype).unit(j).cost{k},inf);
            %it was ...cost{k},0);

            acttype(newlyadded(i),restype).unit(j).mastercost=acttype(newlyadded(i),
            restype).unit(j).mastercost + cost;

                end %end for k=1:numcosts
            end %end if costcheck
            if optchoice==3
                rc=[rc acttype(newlyadded(i),restype).unit(j).mastercost];
            end % end if optchoice==3 (YES, I need it to be checked
twice!)

        end %end for j=1:reslimits(restype)
    end %end i=1:length(newlyadded)
end %end optchoice==3

if optchoice==4 | optchoice==5 | optchoice==1 | optchoice==2 |
optchoice==3

    for i=1:length(newlyadded)
        for j=1:reslimits(restype)

            startcheck=isfield(acttype(newlyadded(i),restype).unit(j),
            'start');

            if startcheck==0 |
            isempty(acttype(newlyadded(i),restype).unit(j).start)==1
                acttype(newlyadded(i),restype).unit(j).masterstart=1;
            else

            acttype(newlyadded(i),restype).unit(j).masterstart=eval(acttype(newlya
            dded(i),restype).unit(j).start);

            end %end if startcheck
            if optchoice==4
                rc=[rc acttype(newlyadded(i),restype).unit(j).masterstart];

```

```

        end % end if optchoice==4 (YES, I need it to be checked
twice!)

end % end for j=1:reslimits(restype)
end % end i=1:length(newlyadded)
end % end optchoice==4

if optchoice==5
    for i=1:length(newlyadded)
        for j=1:reslimits(restype)

            timedep=acttype(newlyadded(i),restype).unit(j).tuned;
            pref=acttype(newlyadded(i),restype).unit(j).masterpref;
            cost=acttype(newlyadded(i),restype).unit(j).mastercost;
            starttime=acttype(newlyadded(i),restype).unit(j).masterstart;

            utility=char(utility);
            composutility=eval(utility);

            if isnan(composutility)
                composutility=inf;
            end

            rc=[rc -composutility];
        end % end for j=1:reslimits(restype)
    end % end i=1:length(newlyadded)
end % end optchoice==5

% Scheduler File
if ~exist('typeselect')
    typeselect=[];
end

[b]=constraints(inprogress,finished,reslimits,actneeds, typeselect);
[c]=objective(prior,actneeds,reslimits,cand,
actdur,multiplier,numsucc, mindur);
[[a]=amatrix(actneeds, cand,finished);
[a]=amatrix(actneeds,cand,finished,reslimits,inprogress, typeselect);
x=balas(a,b,c,0); % see http://www.ima.mdh.se/tom/

% End Scheduler File

% Scheduler "Down" File

[b]=constraintsdown(inprogress,finished,reslimits,actneeds);
[c]=objectivedown(prior,actneeds,reslimits,cand,
actdur,multiplier,numsucc, mindur);
[a]=amatrixdown(actneeds, cand,finished,reslimits,inprogress);
x=balas(a,b,c,0); % see http://www.ima.mdh.se/tom/

% End Scheduler "Down" File

function [acttype]=setassigned(acttype, numact, numres, reslimits)

```

```

for i=1:numact
    for j=1:numres
        for k=1:reslimits(j)
            acttype(i,j).unit(k).assigned=0;
        end
    end
end
end

```

```

function [actdur]=updateactdur(actdur, newlyadded, acttype, reslimits)

```

```

for w=1:length(newlyadded)
    for rt=1:length(reslimits)
        actdur(newlyadded(w))=max(actdur(newlyadded(w)),
            eval('max([acttype(newlyadded(w),rt).unit(:).tuned].*[acttype(newlyadded(w),rt).unit(:).assigned])','0'));
    end
end
end

```