UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MULTI-OBJECTIVE QUERY OPTIMIZATION FOR MOBILE-CLOUD DATABASE ENVIRONMENTS BASED ON A WEIGHTED SUM MODEL

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

FLORIAN HELFF
Norman, Oklahoma
2016

MULTI-OBJECTIVE QUERY OPTIMIZATION FOR MOBILE-CLOUD DATABASE ENVIRONMENTS BASED ON A WEIGHTED SUM MODEL

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

_____
Dr. Le Gruenwald, Chair

_____
Dr. Laurent d'Orazio

_____
Dr. Changwook Kim

Acknowledgements

I would like to thank Dr. Le Gruenwald, chair of my thesis committee and my supervisor at the University of Oklahoma, as well as Dr. Laurent d'Orazio, member of my thesis committee and member of the MOCCAD research team, for their guidance, leadership and support during my Master's studies. Furthermore, I would like to thank Dr. Changwook Kim for his interest in my thesis project and for being part on my thesis committee.

I would like to thank Mr. Chenxiao Wang for his help on challenging ideas and I would like to thank Mrs. Virginie Perez Woods for her help as Academic Programs Coordinator and designated CS-Mom.

I also would like to thank the National Science Foundation (NSF) for partially funding my thesis.

Finally, I would like to thank my fiancée Sara Collins for her support, encouragement and endless hours of proofreading.

This material is based upon work supported in part by the National Science Foundation under Grant No.1349285. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

List of Figures

Abstract

In mobile-cloud database environments, users request services executed on a cloud through mobile devices. Requested data might be partially cached on the mobile device itself or must be processed on the cloud which leads to multiple contradicting cost objectives such as monetary cost to use the cloud service, query execution time on the cloud or on the mobile device, and mobile device energy consumption. Choosing an optimal query execution plan is crucial for query optimization to minimize the overall cost, but is related to user preferences on those various costs. Single-objective optimization strategies are impractical since those do not consider tradeoffs between different costs. The existing multi-objective optimization strategies of Pareto-Set and Skyline Query lack a sophisticated user interaction since the resulting set tends to be large in size which makes it difficult for a user to select a tradeoff between costs. Furthermore, a user might not be aware of query cost constraints which makes his/her decision process impossible. To fill this gap, this thesis presents the multi-objective Normalized Weighted Sum Algorithm with its novel user-interaction model, using weights associated with cost objectives for query optimization which can be set prior to execution. The proposed model is compared with one- and multi-dimensional optimization strategies in terms of result quality and user interaction. Experiments show that the proposed solution improves the result quality regarding single-objective strategies (lexicographical ordering) and improves user interaction with multi-objective optimization strategies (Pareto-Set / Skyline Query) in terms of user response time and decision accuracy.

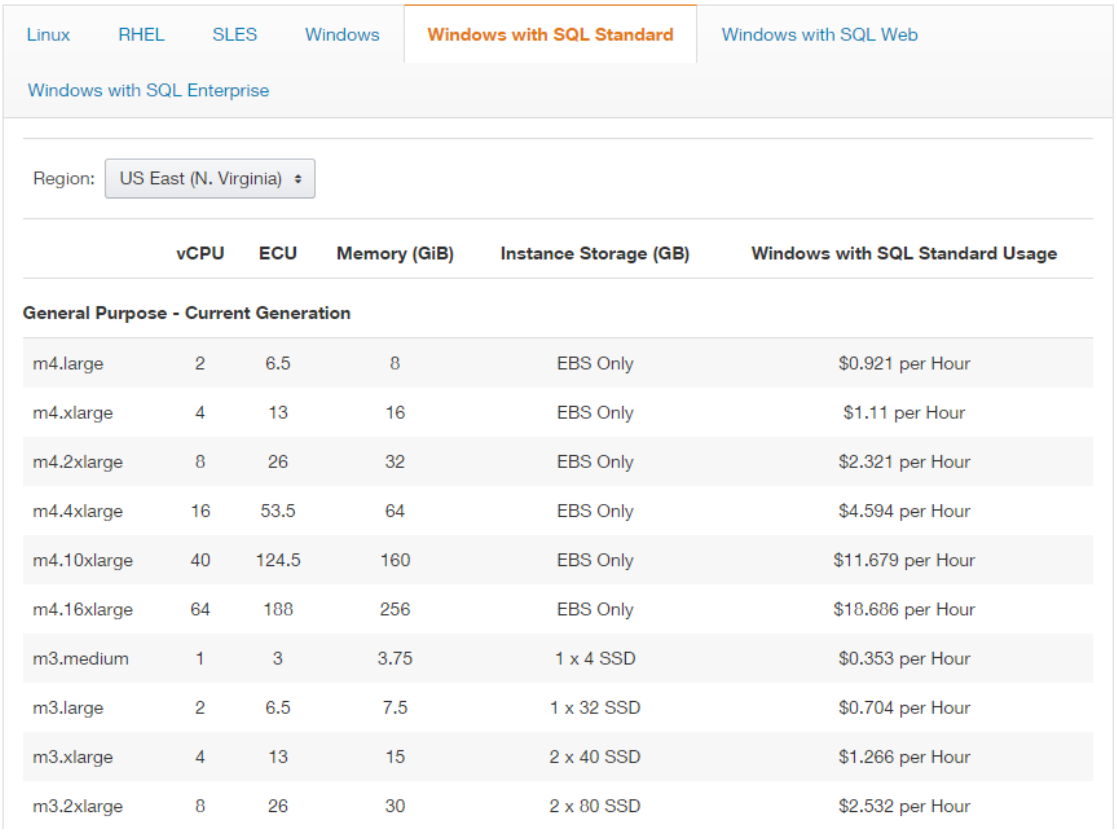
Chapter 1: Introduction

This chapter will introduce the problem domain. Section 1.1 introduces domain specific definitions and explains the given environment. Section 1.2 follows the problem definition and section 1.3 gives an overview about the following Chapters and organization of this thesis.

1.1 Definitions

1.1.1 Cloud Environment

The main characteristic of a cloud service is the high computing elasticity and resource management. Elasticity is defined as horizontal elasticity, when increasing the number of nodes (physical machines) a service is executed on, and vertical elasticity, when increasing the computation power and/or memory of nodes [1]. An example of elasticity can be seen in Figure 1 which represents Amazon's "Web Service prices - EC2 - On-Demand Pricing" [2]. In Amazon's price scheme, they offer different types of nodes with different computation power and memory for different rates. Assuming that a higher computation power/memory leads to a faster response time of services, cloud environments introduce the trade-off between service execution time and service monetary cost since one objective cannot be optimized without weakening the other objective.



Linux | RHEL | SLES | Windows | **Windows with SQL Standard** | Windows with SQL Web

Windows with SQL Enterprise

Region: US East (N. Virginia)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Windows with SQL Standard Usage
General Purpose - Current Generation					
m4.large	2	6.5	8	EBS Only	$0.921 per Hour
m4.xlarge	4	13	16	EBS Only	$1.11 per Hour
m4.2xlarge	8	26	32	EBS Only	$2.321 per Hour
m4.4xlarge	16	53.5	64	EBS Only	$4.594 per Hour
m4.10xlarge	40	124.5	160	EBS Only	$11.679 per Hour
m4.16xlarge	64	188	256	EBS Only	$18.686 per Hour
m3.medium	1	3	3.75	1 x 4 SSD	$0.353 per Hour
m3.large	2	6.5	7.5	1 x 32 SSD	$0.704 per Hour
m3.xlarge	4	13	15	2 x 40 SSD	$1.266 per Hour
m3.2xlarge	8	26	30	2 x 80 SSD	$2.532 per Hour

Figure 1: Amazon Web Service prices - EC2- On-Demand Pricing [2]

1.1.2 Mobile-Cloud Environment

The importance of mobility is the driving factor in the field of mobile computing. Independent of their position, users request services through their mobile devices. Nevertheless, with the advantage of transportable devices to guarantee mobility comes the trade-off of limited resources and computational power. This limitation leads to mobile-cloud computing [3, 4] and a mobile-cloud database environment [5], where a user issues a service request to the cloud from a mobile device to obtain data. This requested data is either stored on the cloud or retrieved from a cache on the mobile device. In addition to the two cost objectives

explained in Section 1.1.1, energy consumption on the mobile device is an additional cost to monetary cost and execution time.

1.2 Problem Definition

With monetary cost, query execution time and energy consumption, query scheduling is a multi-objective optimization problem. Based on the elasticity of the cloud (considering different cloud pricing models, such as AWS [2] and MS Azure [6]) and the possible execution of a query on a mobile device, a query can be executed in multiple ways, each with a distinct combination of costs. This optimization process is a strain of contradicting objectives as usually a faster execution leads to a higher monetary cost, which makes Multi-Objective Query Optimization a crucial problem to the mobile-cloud environment.

It is the goal of this thesis to develop an algorithm that can determine an optimal way to execute a query based on the preferences of a user without burdening him/her with complex user interactions. This algorithm will be based on the architecture of a mobile cloud database environment proposed in [7] which is shown in Figure 2. In this figure, a user is using a mobile device to issue a query that needs to access the database stored on the cloud (Step 1: Issue Query). This user can be seen as a client of the cloud database services. The site of the mobile device distinguishes itself from the other sites through the high mobility but limited resources and computational power.

After the query is issued, the implemented query cache on the mobile device is checked if the data requested by the query is already available in the cache [7] (Step 2). If the data is not available, the query is forwarded through the

data owner (Step 3) to the Cloud Services (Step 4). The data owner is owner of any produced data and is responsible for it. For example, it can be a business like a university or hospital, which is producing data and utilizes cloud services to store and process it. On the cloud, the query will be optimized during the step of query processing (Step 5). On the cloud site, based on its high elasticity, there may exist multiple ways for executing a query. Those options are defined by the available cloud resources (nodes) and result in a multi-dimensional cost.

After a way to execute the query (Query Execution Plan) is selected, the query will be executed (Step 6) and query results are received. This process is explained in more detail in Section 2.1. The query results are forwarded through the data owner (Step 7) to the user (Steps 8 and 9). Finally, based on the implemented cache replacement policies, the cache will be updated based on the last query (Step 10).

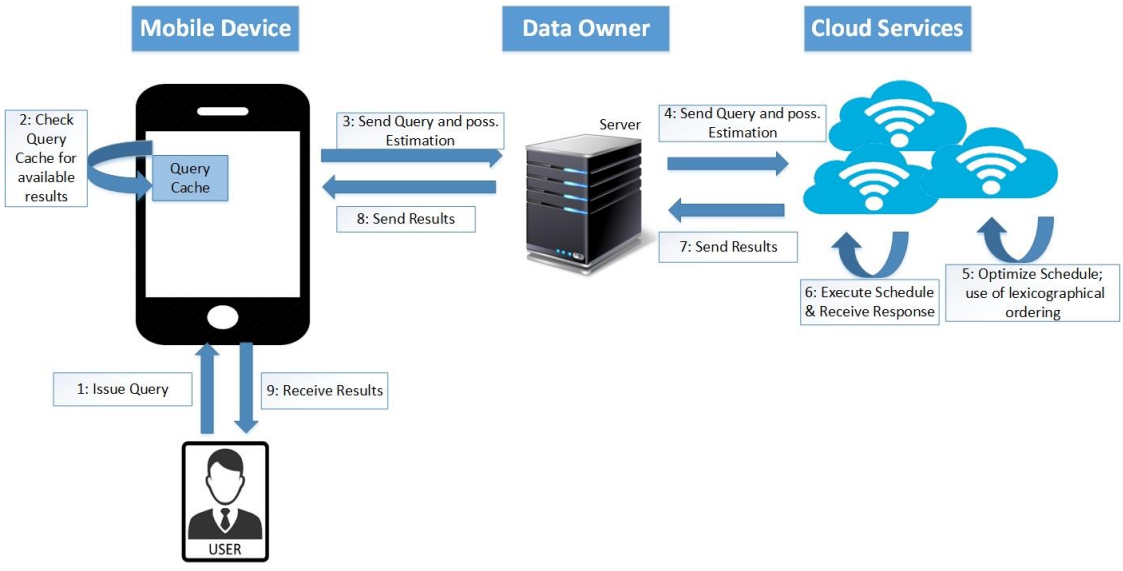


Figure 2: Mobile-Cloud Database Architecture

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the query processing background and a literature review on the topic of multi-objective query optimization. Chapter 3 presents the proposed solution, the Normalized Weighted Sum Algorithm (NWSA), the associated user interaction models, a NWSA based scheduling algorithm, and the proposed architecture including the preceding aspects. Chapter 4 describes the experiments conducted to study the performance of the proposed solution and their results. Finally, Chapter 5 provides the conclusions and presents future work.

Chapter 2: Literature Review

Chapter 2 contains the literature review, describing fundamental and related work. Section 2.1 focusses on Query Processing and the different steps needed for it, followed by section 2.2 describing different multi objective optimization strategies.

2.1 Query Processing

Query processing processes an input query to compute its final query execution plan (QEP) and then executes this QEP to obtain the requested query data [8]. Query Processing is realized in multiple steps shown in Figure 3.

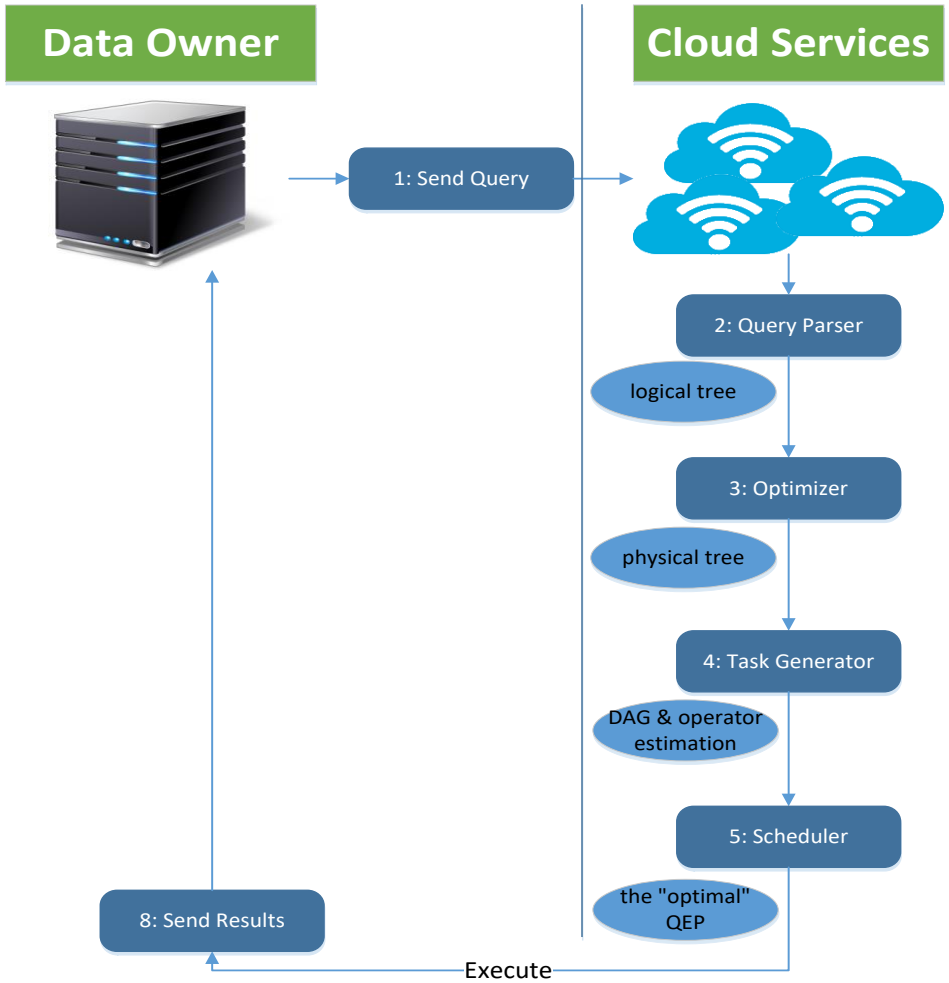


Figure 3: Query Processing Steps

After receiving a query (Step 1), the query parser converts the query into a logical query plan also known as a logical tree (Step 2). This tree uses relational algebra to eliminate the query language specific syntax. The optimizer then processes the logical tree in Step 3 using algebraic transformations to optimize the tree. An example of optimization is “pushing down” an early selection in the tree. This optimization usually reduces the size of an algebraic relation, which then results in a faster execution of the following operations. After applying algebraic transformations to get an optimized tree, the task generator (Step 4) will then use

the database statistics on the data size of the relations involved in the query to further optimize the tree and to convert the query operators from the tree into algorithms. This, for example, can include replacing join operators by next-loop join or hash join algorithms, or choosing a method for selection operators. The final step (Step 5) before executing the query is completed by the scheduler, which assigns each operator to a node and to a container on a node where the operator should be executed. This assignment is crucial because it defines the costs of a query in terms of monetary cost that must be paid to the cloud service provider, execution time and energy consumption based on the pricing models of the used nodes. Each possible assignment combination of operators to containers represents a possible query execution plan (QEP); all of them will yield the same results when executed, but consist of different associated costs. It is the task of a multi-objective optimization strategy to find the best QEP in terms of its costs for execution.

2.2 Multi-Objective Optimization

Multi-Objective Optimization is a crucial process in the mobile cloud environment. Based on the different occurring costs explained in Section 1.1 and Section 2.1, it is the goal of a multi-objective optimization strategy to find a trade-off among the given contradicting objectives of a QEP. Section 2.2.1 will provide an introduction of the state of the art single-objective optimization and explanations for why those strategies are not suitable for multi-objective optimization problems. Next, Section 2.2.2 and Section 2.2.3 will give an introduction of multi-objective optimization using Pareto-Set Optimization, Skyline Queries and Scoring

Functions. Finally, Section 2.2.4 gives an introduction to the Weighted Sum Model, which is the foundation of the proposed solution in Chapter 3.

2.2.1 Lexicographical Ordering

A user-friendly decision strategy to find the optimal QEP in terms of costs is the lexicographical ordering [9]. Lexicographical ordering focuses on a single main objective, such as execution time, and orders further objectives, like monetary cost and energy consumption, in a descending order. The complexity of lexicographical ordering is in linear relation to the count of alternatives it selects its solution from because it scans the alternative once for the lowest parameter. While it is an easy executable strategy, lexicographical ordering is not a feasible solution for multi-objective optimization as shown in following example:

QEP1: {M= $0.080; T= 0.5s; E= 0.012 mA}

QEP2: {M= $0.050; T= 3.0s; E= 0.300 mA}

QEP3: {M= $0.055; T= 0.6s; E= 0.013 mA}

Figure 4: Lexicographical Ordering: Execution Plan Costs Example

Considering the three query execution plans (QEPs) with their costs for monetary costs (M), execution time (T) and energy consumption (E) shown in Figure 4, focusing on a single objective always leads to the decision to select either plan QEP1 or QEP2 for execution since these QEPs have a minimum cost in one of the three objectives. Since QEP3 does not have a minimum value in any of the three costs, it will never be selected although it is a competitive choice when considering

all three objectives on the same level of importance. Therefore, a strategy which considers all objectives at the same time is needed in order to make a comprehensive decision in a multi-objective optimization environment.

2.2.2 Pareto-Set Optimization and Skyline Queries

This section describes the Pareto-Set Optimization, which is fundamental for every Multi-Objective Optimization problem. The goal of this optimization is the calculation of a set of non-dominated alternatives, the Pareto-Set. An alternative 'A' is dominating an alternative 'B' if at least one objective (decision variable) of 'A' is better than the associated objective in 'B' and all other objectives in 'A' are at least equal to the associated objectives in 'B'. Those dominating alternatives are called Pareto-Optimal [10, 11]. In the context of query optimization, every alternative represents the costs of a single QEP.

Figure 5 gives an example for a two-dimensional Pareto-Set query optimization. Alternatives a, b, and c are not dominated by any other point in this graph, since there exists no alternative with better cost values for both objectives. The covered squares of a point represent the dominated area of an alternative. Every point lying within a covered square is dominated by the alternative spanning the square.

Skyline queries [12, 13, 14] are one example of a strategy used to find a Pareto-Set. The definition of a Skyline Query is similar to a Pareto-Set. The result of a skyline query is called a skyline. This skyline contains objects which fulfill the requirements of not being dominated by another object in terms of a given utility

function [15]. Skyline queries were first mentioned in 2001 where three basic skyline computations were introduced [16].

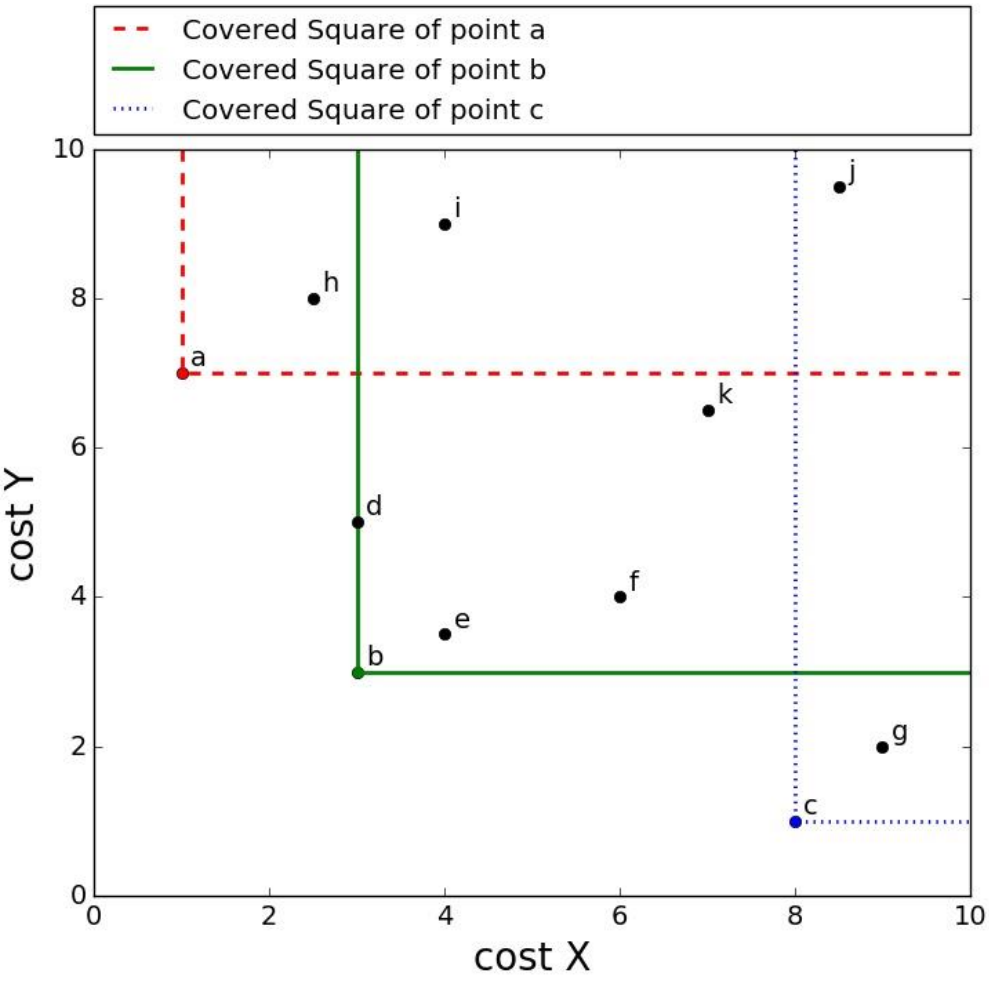


Figure 5: 2-Dimensional Pareto Set Optimization

Much research has been conducted to optimize the Pareto Set / Skyline algorithm itself [17]. Query optimization techniques to speed up the calculation time and sizing down the resulting set using heuristics [18] and preprocessing [19] were introduced in the research by I. Trummer and C. Koch. Those techniques are needed since the calculation of all possible QEPs is not computable.

However, the major drawback of a Pareto-Set/Skyline Query is the resulting set of alternatives, which tends to be very large in size so that users are left with a choice to select a single QEP out of a large pool of possible solutions [20]. Research on the size of a Pareto-Set already estimated its size to be $\Theta((\ln n)^{d-1} / (d-1)!)$ for *n* data objects and *d* objectives, assuming attribute independence [21, 20]. This problem was addressed by Lee et al. [15] by providing a way of reducing the skyline to a size manageable by users. Users are repeatedly asked about their preference between two chosen objectives. These questions are designed according to the answer's impact on the size of the skyline to minimize user interaction, but the approach still uses multiple user interactions.

Furthermore, the research on Skyline Queries neglects the fact that the user executing the query might not be aware of all constraints on the different costs to select a sophisticated QEP. In the use case of a larger organization, a query executing user might know the restrictions on execution time and energy consumption but might be unaware of the monetary cost he/she is allowed to spend since a budget is often centrally managed by his/her supervisor. An approach is needed to separate query executing users from another type of users, whom we call *superusers*, who do not execute queries, but know and have authorities to set the restrictions on the queries.

In summary, the skyline approach makes it possible for the user to consider multiple objectives at the same time, but also burdens the user with the crucial decision to select the final QEP.

2.2.3 Scoring Function

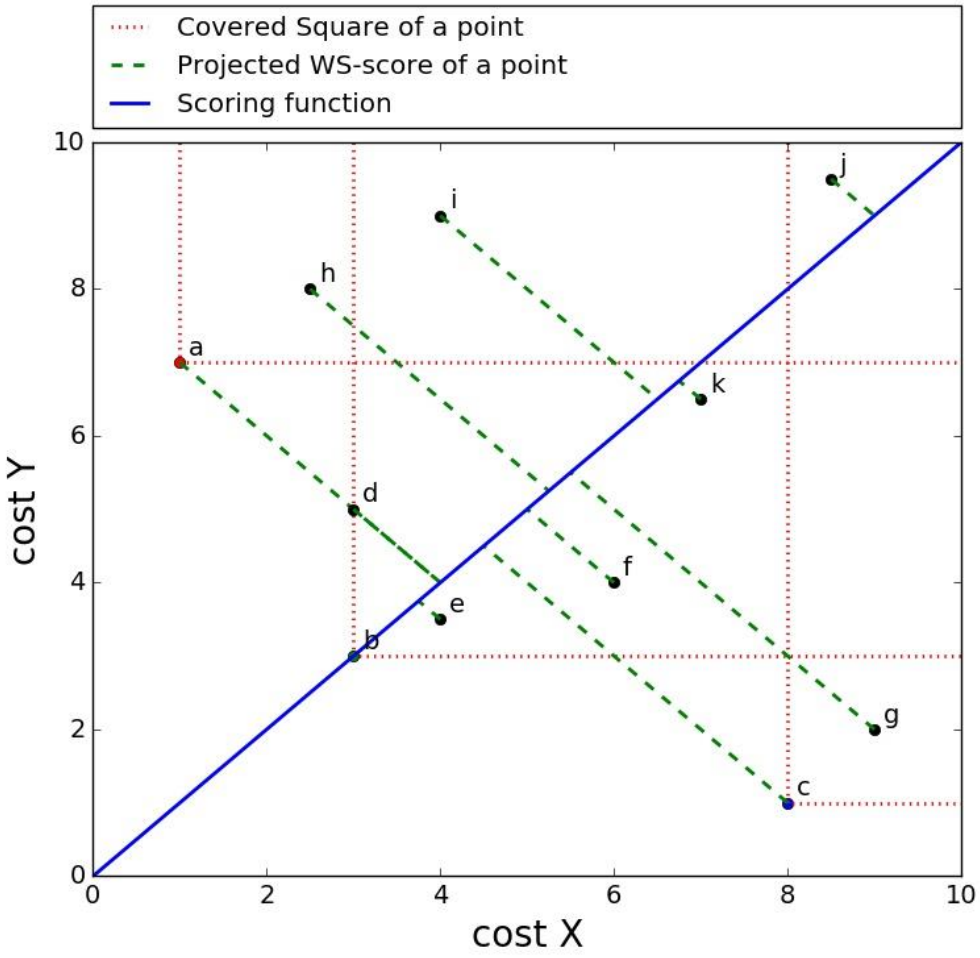


Figure 6: 2-Dimensional Scoring Function

The strength of finding a Pareto-Set is that every alternative in this set is optimal for at least one scoring function. A scoring function describes a specific stress configuration on the different objectives in order to set the importance to them and to compare alternatives in this Pareto-Set [22]. A disadvantage of a scoring function is that the stresses on objectives have to be defined prior to execution whereas the Pareto-Set does not require any further input besides the

QEPs. An advantage of a scoring function is that users do not have to select a QEP out of a set since a specific scoring function only has one optimal solution. Figure 6 shows a scoring function (blue): every point in this two-dimensional space will be projected (green) on a linear function using the stresses. The value of this projection (the intersection of the green projection and the blue scoring function) is the score. The alternative with the minimum score is an element of the Pareto-Set as proven in Section 3.1.2 and [23].

2.2.4 Weighted Sum Model

The Weighted Sum Model (WSM) **[24, 25, 26]** is most commonly used in multi-objective optimization problems. In this model, every possible alternative (a QEP in our application) is rated by a score. The score for each alternative includes all objectives, which are individually weighted to stress the importance of different objectives. This model is used in many multi-objective optimization problems in various fields of computer science, such as network and optimization problems **[27, 25]**, and also other fields such as economics (Cost-Utility Analysis) **[28, 29]**. The formulas used in this model are shown in Figure 7.

In these formulas, the WSM-score for an alternative A_i denoted as $A_i^{WSM-score}$ is calculated by adding the products of a weight w_j with its corresponding parameter a_{ij}, the value of this objective. This parameter is, for example, the monetary cost which has to be spent to execute the query. A set of weights w_j for every parameter a_j is called a weight profile (WP).

The best alternative is chosen as the one which has the maximum WSM score ($A_*^{WSM-score}$). The different objectives are assumed to be positive: the higher the score, the better the alternative. Assuming the objectives to be negative (in case of cost models), the best alternative has equivalently the lowest score.

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij}$$

$$A_*^{WSM-score} = \max_i \sum_{j=1}^{n} w_j a_{ij}$$

Figure 7: Weighted Sum Model Scoring Function

The weakness of this model is the process of summarizing the different objectives. The fact that different objectives might have different dimensions and units leads to the problem of adding apples and oranges [30]. Since the mobile-cloud database environment has to deal with multi objectives with different units, the Weighted Sum Model cannot be used without major changes in its strategy.

This section described different strategies for multi-objective optimization. I showed that the common lexicographical ordering is not suitable for the decision on multiple objectives and showed the disadvantages of Pareto-Set optimization in terms of user interaction. Furthermore, I introduced the concept of the weighted sum model, which I will expand in Chapter 3 to fit the described problem domain.

Chapter 3: Proposed Solution and Architecture

This chapter presents the proposed multi-objective optimization strategy, the Normalized Weighted Sum Algorithm (NWSA), and its effects on user interaction, query scheduling algorithm and system architecture. Specifically, Section 3.1 describes the algorithm itself and compares it to the multi-objective optimization strategy of Pareto-Sets. Section 3.2 discusses the user interaction with NWSA, compares it to the user interaction with Pareto-Sets, explains the concept of "superusers" and "query executing users", and demonstrates a developed sophisticated superuser interface to highlight the advantage of user-type separation. Section 3.3 then incorporates NWSA into a generic greedy query scheduler and discusses its impact. Finally, Section 3.4 explains the impact of NWSA to the existing mobile-cloud database architecture [7], previously explained in Section 1.1.2.

3.1 Normalized Weighted Sum Algorithm

This section describes the proposed algorithm called the Normalized Weighted Sum Algorithm (NWSA), the implementation details and the proof showing that NWSA always selects a query execution plan from the Pareto-Set.

As already pointed out in Section 2.2.4, one problem of the WSM is the addition of multiple dimensions or units. This problem can be resolved by normalizing the different parameters [25]. This normalization can be done in relation to a user-defined maximum of acceptance of each objective. The resulting values represent the fraction towards this maximum and do not contain a unit which makes them addable to each other. This normalization to a user-defined

maximum of parameters adapts the ideas by Goh and Cheng [31] of a user-based decision to eliminate constraint violating alternatives. Another advantage of this user-defined maximum of acceptance of each objective can be seen in the implementation of the algorithm, shown in Section 3.1.1. Alternatives which violate those regulations can be taken out of consideration to follow the defined conditions.

The second adjustment for NWSA regarding the general WSM is done in regards to the weights. To include environmental factors, the used weight is composed of a user-defined weight and an automatically generated environmental weight. Environmental factors are, for example, the current battery status, an ongoing charging process or factors describing the currently used cloud. The environmental weight can adjusts the user weight if, for example, a mobile device is being charged and energy consumption is obsolete, or a query is run overnight and execution time should be assigned a minor importance factor. If all environmental weights are kept equal, those will not have an impact on the previously defined user weight. The experiments of Chapter 4 will not further study a modification by environmental weights.

In conclusion, the Modified Weighted Sum Model Scoring Function can be expressed as in Figure 8.

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j \frac{a_{ij}}{m_j}$$

Figure 8: Modified Weighted Sum Model Scoring Function

The function consists of following variables: a_{ij} is the value of alternative *i* (QEP_i) for objective *j*, m_j the user-defined acceptable maximum value for objective j, and w_j the normalized composite weight of user and environment for objective j defined in Figure 9.

$$w_j = \frac{uw_j * ew_j}{\sum(uw * ew)}.$$

Figure 9: Composite Normalized Weight Factor

Figure 9 shows the computation of the composite weight where uw_j and ew_j describe the weight of the user and the environmental weight for objective *j*, respectively. Since the different objectives are representative of different costs, the algorithm chooses the alternative with the lowest score to minimize costs as shown in Figure 10. It can be noted, that the number of different alternatives does not influence this algorithm which makes NWSA adaptable to any number of different objectives.

$$A_*^{WSM-score} = \min_i \sum_{j=1}^{n} w_j \frac{a_{ij}}{m_j}$$

Figure 10: Modified Weighted Sum Model Scoring Function: Optimal Alternative

In the following Section 3.1.1, the proposed algorithm is described. It is then followed by a proof showing that the chosen alternative in this decision process is always an element of the corresponding Pareto-Set, which is independent of the chosen weights.

3.1.1 Algorithm

Algorithm 1: *NWSA- Decision Algorithm*

Input:

Alternatives A_i *with i=1..m and parameter* a_{ij} *to objective* O_j *with j=1..n* ;

uw_j *the user weight for objective* O_j;

ew_j *the environment weight for objective* O_j;

m_j *the maximum accepted value for objective* O_j

Output: *best alternative* A_i

```
1.  A_best ← null
2.  A_bestRestrictionViolating ← null

3.  for i=1 to m
4.      A_i^score ← CalculateScore(A_i) //calcualte NWSA-scores
5.  end for

6.  for i=1 to m
7.      violate ← false
8.      for j=1 to n
9.          if(a_ij>m_j)
10.             violate ← true //check alternative for constraint violation
11.             save violation
12.         end if
13.     end for
14.     if(violate=true)
15.         if(A_i^score< A_bestRestrictionViolating^score)
16.             A_bestRestrictionViolating ← A_i //check if the current alternative is the best one
17.         end if
18.     else
19.         if(A_i^score < A_best^score)
20.             A_best ← A_i
21.         end if
22.     end if
23. end for

24. if(A_best ← null)
25.     return A_bestRestrictionViolating , violations //returns the best alternative with its violation
26. else
27.     return A_best  //returns the best alternative
28. end if
```

Figure 11: Algorithm 1: NWSA - Decision Algorithm

The developed algorithm is shown in Figure 11. Its goal is to calculate the best alternative in a multi-objective decision process.

The modified WSM function to calculate the score of an alternative Ai (CalculateScore(Ai)), which is used in Line 5 of this algorithm, is the previously defined function in Figure 8. This function is executed for every alternative (Lines 3-5). As it can be seen in this algorithm, each alternative is checked to see if it violates the user-defined maximum value for each objective (Lines 8-13). The violation itself has to be saved for future use (Line 11). Afterwards, the best alternative (A_{best}), which is the one with the lowest score, is selected (Lines 14-22) and returned as the output of the algorithm. If all possible alternatives violate those restrictions, the algorithm will return the lowest score alternative ($A_{bestRestrictionViolating}$) as well as the previously saved restriction(s) that it violates (Lines 24-25). The complexity of this algorithm is in linear relation to the count of alternatives, which is also the complexity of the lexicographical ordering strategy as discussed in Section 2.2.1. The linear complexity of the given algorithm is based on the single iteration over all alternatives with each iteration having a constant complexity.

3.1.2 Proof for Pareto-Set

This section provides a proof demonstrating that, independent of possible weights and the number of objectives, the proposed algorithm always picks an alternative within the Pareto-Set.

Proof by contradiction:

It is assumed that the chosen algorithm picks an alternative A_{best} which is not an element of the Pareto-Set. In compliance with the used formula in the proposed algorithm (Figure 8) it can be determined that:

$$\boldsymbol{A_{best}^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij} = \max_i \sum_{j=1}^{n} w_j a_{ij}}$$

Since A_{best} is not an element of the Pareto-Set, an alternative A_{pareto} has to exist which dominates A_{best}. According to the definition of the Pareto-Set, this alternative has one higher value for at least one criterion than A_{best} without having a lower value for all other objectives. This is in contradiction to

$$A_{best}^{WSM-score} = \max_i \sum_{j=1}^{n} w_j a_{ij}$$

since the score is better for A_{pareto} with unchanged weights. A similar proof is shown by Zadeh [32] and in the context of skyline queries is shown by Chomicki [12].

3.2 User Interaction Models for NWSA and Pareto Set

Figure 12 shows the user-interaction models for Skyline Queries / Pareto-Set and NWSA, respectively. Comparing both models shows that both approaches have the query as input and use the Pareto-Set algorithm or NWSA algorithm, respectively, to handle the request. In addition to the query, NWSA also requires the input of a weight profile to stress the objectives. The result of the Pareto-Set algorithm is the set of Pareto-Optimal alternatives. It is the task of a user to

manually select one of the given alternatives for execution of the query. This decision on the Pareto-Set to select a single alternative is eliminated in the user interaction model of the NWSA approach. The NWSA algorithm directly computes a single optimal solution based on the given weight profile.

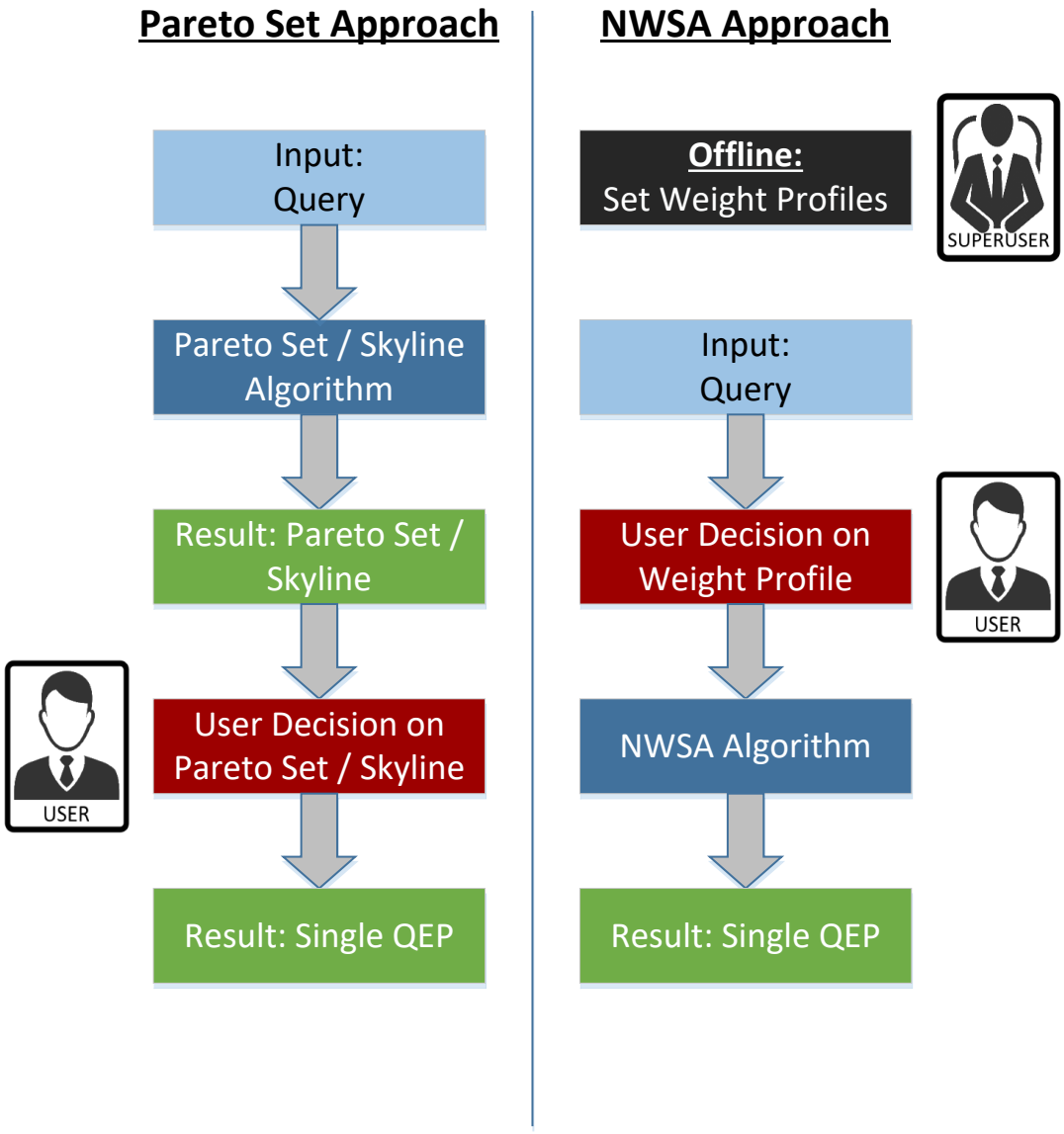


Figure 12: User Interaction Models: Pareto-Set Approach and NWSA Approach

The fact that user preferences/weights can be preset prior to execution of the algorithm is another advantage NWSA has over the Pareto-Set approach. To

take advantage of being able to preset weight profiles, we propose two types of users: users that preset weight profiles (superusers) and users that invoke the execution of a query (query executing users). A weight profile based on the application requirements can be preset by a superuser who is aware of all constraints on the different objectives. This weight profile can then be selected by a user executing the query, which is minimizing the decisions he/she has to make. Furthermore, a weight profile can be described by an application-based logical description, such as “emergency query” or “batch query”, to describe the preferences a weight profile reflects. In the example of emergency queries or batch queries, the description would reflect a weight profile with high importance on execution time or low importance on execution time, respectively. The number of different weight profiles is directly influenced by the superuser. This number should be kept low (5 or less) to not burden the query executing user with a difficult decision on a weight profile. A decision on a large number of weight profiles would be similar to a decision on a Pareto-Set, which we try to avoid. This aspect is shown in the experiment shown in section 4.2. A real world example would be the number of possible shipping types when ordering an item: a large number of shipping options would complicate the decision process whereas a decision on just a few alternatives is specific enough to find the desired trade-off.

3.2.1 Superuser Interface

Taking advantage of being able to preset weight profiles by superusers and making the decision simple for executing users still leaves a superuser with his/her decision on how to set weight profiles. A sophisticated example for a

superuser interface to setup weight profiles has been developed and is shown in Figure 13.

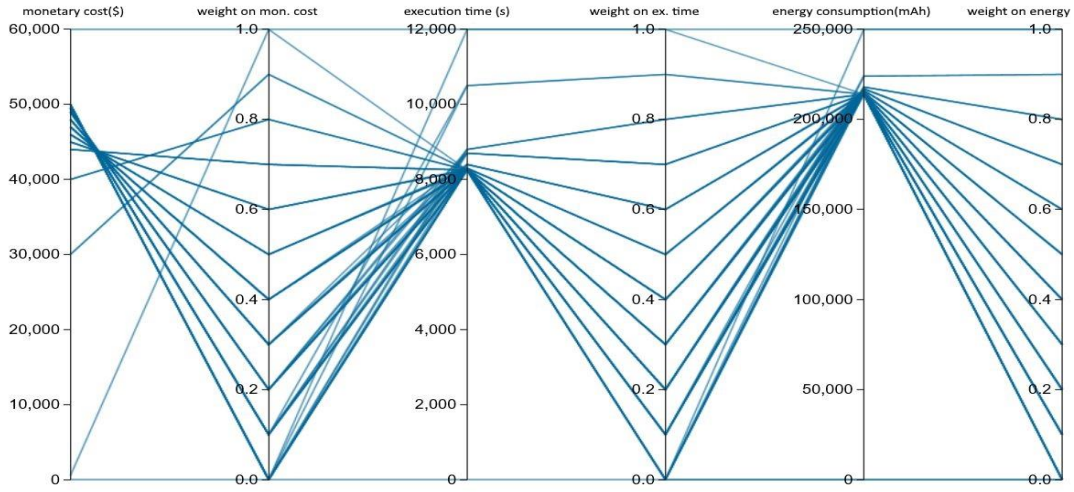


Figure 13: The Superuser Interface

The developed superuser interface has the purpose of giving a tool to the superusers to set accurate weight profiles based on their application requirements for the executing users to select. The main feature of this interface is the graphical representation of all possible weight profiles in an interactive parallel plot [33]. Parallel plots are used to show connections between high-dimensional objectives: a point of an *n*-dimensional space is represented by *n* vertices on *n* parallel dimensions which are connected by a polyline. The position of a vertex in any dimension corresponds to the coordinate of this point in this dimension. Figure 13 shows such a parallel plot with each line representing a weight profile (weights on monetary cost, execution time, and energy consumption) and its corresponding cost which is randomized in this example. An example of a weight profile would be 0.8 weight on monetary cost, 0.1 weight on execution time, and 0.1 weight on energy with its corresponding costs of $40,000, 8100 seconds, and 220000 mAh

(milliampere hour). The three different costs are calculated based on the average cost of a query using this specific weight profile, then multiplied by the expected number of queries to execute. This data can be received by the system when evaluating historical data of query execution during the previous time periods. Assuming one million queries, the corresponding average costs per query are $0.04, 0.0081s, and 0.22 mAh when using the weight profile in the previous example.

Furthermore, Figure 14 shows the use of a filter function, constraining the monetary cost to a limit of $45,000. Selecting a range on an axis reduces all polylines to those that have a vertex in the defined range. In the given example, the superuser is able to constrain the resulting costs of a weight profile to narrow his/her choice on a weight profile for the query executing user.

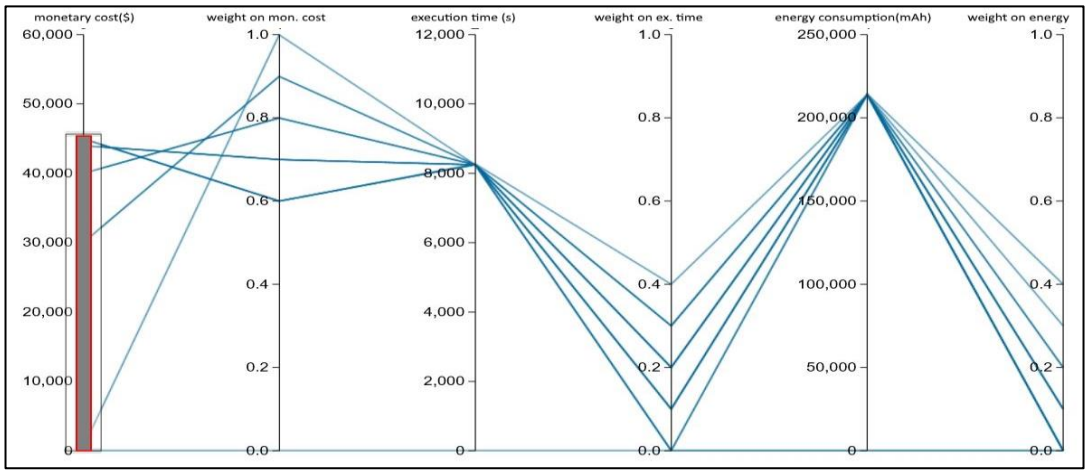


Figure 14: The Superuser Interface including a constraint selection on monetary cost

3.3 Normalized Weighted Sum Algorithm Based Scheduling Algorithm

This section will describe, how to incorporate the Normalized Weighted Sum Model in the scheduling process, previously described in section 2.1. This step is needed to reduce the complexity of a Pareto-Set scheduler as analyzed in section 3.3.1.

The "Normalized Weighted Sum Algorithm based scheduling algorithm" (NWSA-S), shown in Figure 15, is a two stage algorithm based on the generic nested loop scheduling algorithm presented by Kllapi et al. [34].

Algorithm 2: *NWSA based scheduling algorithm (NWSA-S)*

Input:

WP : The selected weight profile

G: The dataflow graph

CONST: Solution constraints

LIMIT: Container limit sequence generator

Output: *bestQEP: the best QEP of G given the other parameters.*

```
1. bestQEP ← ∅
2. while LIMIT.hasNext() and STOP.continue() do
3.     limit ← LIMIT.getNext()
4.     nextQEP ← SCHEDULER(WP; G; limit; CONST)
5.     bestQEP ← FILTER(bestQEP, nextQEP)
6.     STOP.addFeedback(nextQEP)
7. end while

8. return bestQEP
```

Figure 15: Algorithm 2: NWSA based scheduling algorithm (NWSA-S)

The input parameters for this Algorithm are the following:

- The weight profile (WP) selected by the query executing user and preset by the superuser.

- The dataflow graph (G) which is the result of the Task Generator (compare Section 2.1, Figure 3).
- The solution constraints (CONST) as the maximum allowed costs for each objective. These constraints are used for the normalization process of NWSA.
- The list of containers (LIMIT), including specifications of each container, available for operator scheduling.

In addition to the input, the algorithm uses four functions which are defined as follows:

- LIMIT.getNext returns a set of containers which should be used to schedule the operators for G in order to determine the QEP using this set of containers. In case of homogeneous containers, LIMIT.getNext returns an increasing number of containers with identical specifications.
- SCHEDULER (WP; G; limit; CONST) determines the optimal QEP based on the given weight profile, the dataflow graph, the set of containers and the maximum allowed costs for each objective. An NWSA greedy implementation of this algorithm is given in Figure 16.
- FILTER(bestQEP, nextQEP) determines the optimal QEP based on the score of bestQEP and nextQEP. The complexity of this filter is $O(1)$ since it is a simple comparison of two numbers.
- A boolean routine STOP determining whether or not to stop the exploration. The STOP routine used in NWSA-S in case of homogeneous containers is determined by the improvement of a score of a new QEP in

regards to the previous best QEP. If the improvement by the new QEP is below a certain threshold, the algorithm stops the exploration of further containers.

NWSA-S iterates over the different sets of containers specified in LIMIT and called by LIMIT.next() (Line 3), calculates a QEP based on the SCHEDULER (Line 4), compares the new QEP with the current best QEP (Line 5), and updates the STOP routine (Line 6) until there is no new set of containers or the STOP routine stops any further exploration (Line 2). Since NWSA-S is iterating over limit, the complexity of this algorithm is defined by the size of limit and the complexity of the scheduler used in the loop:

$$Complexity\ of\ NWSA-S \in O(n) + O(Scheduler), with\ n = size\ of\ limit$$

The "Scheduler: NWSA Greedy" (NWSA-S_G), shown in Figure 16, is a greedy algorithm based on the generic greedy scheduling algorithm presented by Kllapi [34] which is used as the SCHEDULER function in Algorithm 2 (Figure 15).

Algorithm 3: *Scheduler: NWSA Greedy*

Input:

WP : The selected weight profile

G: The dataflow graph

C: The maximum number of parallel containers to use

CONST: Solution constraints

Output: *S_G: The schedule of G with at most C containers*

1. $S_G \leftarrow \emptyset$
2. $scores_{op} \leftarrow$ NWSA(G, C)
3. ready $\leftarrow$ {operators in G that have no dependencies}
4. **while** ready $\neq \emptyset$ **do**
5. n $\leftarrow$ NEXT(ready, $scores_{op}$)
6. $S_G \leftarrow$ ASSIGN(C , n , WP)
7. ready $\leftarrow$ ready – {n}
8. ready $\leftarrow$ ready + {operators in G that dependencies no longer exist}
9. **end while**
10. **return** S_G

Figure 16: Algorithm 3: Scheduler: NWSA Greedy

The input parameters for this Algorithm are the following:

- The weight profile (WP) selected by the query executing user and preset by the superuser.
- The dataflow graph (G) which is the result of the Task Generator (compare Section 2.1, Figure 3).
- The solution constraints (CONST) as the maximum allowed costs for each objective. These constraints are used for the normalization process of NWSA.
- A set of containers (C), including specifications of each container, available for operator scheduling.

In addition to the input, the algorithm uses three functions which are defined as follows:

- NWSA(G, C) computes the scores for each operator in G for each container in C. In case of heterogeneous containers, NWSA(G, C) has a complexity of $O(op * c)$ with op equals the number of operators in G and c equals the number of containers in C, since a score is computed for every operator on every container In case of homogeneous containers, the complexity of NWSA(G, C) decreases to $O(op)$ since independent from the number of containers, the score of one operator will be equal for all containers in C which leads to one score computation for each operator.
- NEXT(ready, $scores_{op}$) selects one operator out of the set of ready operators. This selection is done by reading the scores of ready operators on all containers and choosing the operator with the overall lowest score for any container in C.

➔ ASSIGN(C, n ,WP): A score for each possible assignment of n to a container in C is calculated. Afterwards, the operator n will be assigned to the container with the lowest score for the resulting schedule. The complexity of ASSIGN is dependent on the number of containers c: $O(c)$ since a score for each container assignment is calculated.

NWSA-S_G starts with an empty schedule (Line 1), computes the score for each operator in G for each container in C (Line 2), and receives all operators without any dependencies (Line 3). The algorithm then loops over the following sequence of operations until no operator is ready to be assigned(Line 4): the operator with

the overall lowest score for any container in C is selected (Line 5) and is assigned to the container whose resulting score for the global schedule is the lowest (Line 6). The selected operator is now assigned to a container and part of the global schedule S_G. Finally, the list of assignable operators is updated by removing the assigned operator (Line 7) and adding the operators in G, which's dependencies no longer exist (Line 8). After all operators are assigned to their containers, the resulting schedule S_G represents the QEP with respect to WP, G, C, and CONST.

3.3.1 Trade-off of NWSA-S_G

NWSA-S_G is a greedy scheduler which will always find the local optimum [35] when assigning a new operator based on the already assigned operators. The complexity of a Pareto-Set Scheduler is element of $O(c^{op})$ with op equals the number of operators in G and c equals the number of containers in C. This is based on the fact that each possible combination of assigning operators to containers represents one QEP. Calculating all those QEPs is not computable in a reasonable amount time since, for example, a schedule of 50 operators and 10 containers would have 10^{50} possible QEPs and would need 10^{50}computations. NWSA-S_G has a comparably low complexity with $O(op * c)$ since operators are iteratively assigned and assume a fixed assignment of previously assigned operators. Given the previous example of 50 operators and 10 containers, there are 10 alternatives of assignments for each operator, resulting in 500 possible QEPs Nevertheless, since the execution time of a schedule S_G, defined as the maximum execution time of all containers in C ($\max_{c \in C} time(c)$), is a non-linear function, the resulting QEP is not guaranteed the global optimum [35] for its cost with respect to WP, G, C, and

CONST. Figure 17 shows an example where NWSA-S_G is not able to find the global optimum schedule.

WP: Weight_Time: 0.7 ; Weight_Money: 0.3

G: {op1, op2, op3; all independent}

C: {c1, c2}

Scores-Table of G x C

<normalized_cost_Time, normalized_cost_Money>	c1	c1_Score	c2	c2_Score
op1	<5,5>	5	<3,9>	4.8
op2	<3,2>	2.7	<2,7>	3.5
op3	<9,10>	9.3	<6,12>	7.8

Schedule 1: c1 = {op2} , c2 = {op1, op3}

➔ Total time: max{3, 3+6} = 9 Total money: 2+9+12 = 23

Schedule 2: c1 = {op1, op2}, c2 = {op3}

➔ Total time: max{5+3, 6} = 8 Total money: 5+2+12 = 19

Figure 17: Trade-off example NWSA-S_G

Following NWSA-S_G, the algorithm first assigns op2 (the operator with the lowest overall score) to c1 (the container with the lowest schedule score), followed by assigning op1 (the operator with the lowest remaining score) to c2 (the container with lowest schedule score), and assigning op3 to c2 (the container with the lowest schedule score). This schedule 1 results in a total time of 9 and a monetary cost of 23. Comparing it to the schedule 2 with c1 = {op1, op2} and c2 = {op3}, the total time of 8 and monetary cost of 19 dominates schedule 1. This

schedule 2 is not found by NWSA-S_G because the assignment for op1 to c1 is not the local optimum assignment but takes into consideration the assignment of op3 to reach the global optimum of the schedule. This problem is tied to the greedy structure of this algorithm since a greedy algorithm always optimizes toward the local optimum.

3.4 Proposed Architecture

Based on the proposed solutions in Sections 3.1, 3.2, and 3.3, the architecture of the Mobile-Cloud Database Environment changes to incorporate the different types of users as well as to incorporate the weight profiles needed for the scheduler. The resulting Mobile-Cloud Database Architecture is shown in Figure 18.

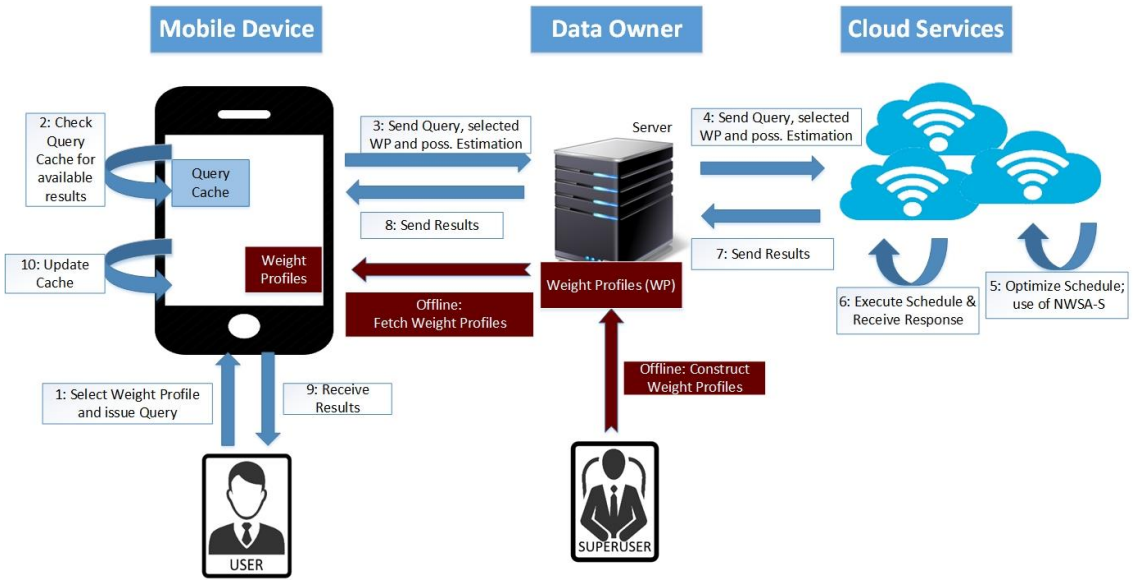


Figure 18: Proposed Mobile-Cloud Database Architecture

As described in Section 3.2, the weight profiles are preset by superusers before a query executing user starts to issue a query. Because of their centralized management, the weight profiles are stored at the data owner and fetched to every mobile device client upon request.

Compared to the previous architecture in Section 1.1.2, Figure 2, an issue of a query not only requires the definition of a query, but it now also requires the selection of a preset weight profile (Step 1). The weight profile is carried along Steps 2 – 4, which do not change from the previous architecture. Step 5 internally changes from using the lexicographical ordering to using the proposed scheduling algorithm NWSA-S, described in Section 3.3.

Chapter 4: Evaluation and Results

This chapter presents the conducted experiments and their corresponding results to evaluate the performance of the proposed model, NWSA. In three experiments, NWSA is compared with the single-objective optimization strategy of lexicographical ordering (Section 4.1), compared with the user interaction model of the multi-objective optimization strategy of Pareto-Set / Skyline Query (Section 4.2), and compared with the Pareto-Set based scheduling algorithm by Kllapi [34].

It is the overall goal of the conducted experiments to show the increased quality of results in regards to single-objective optimization strategies without an additional overhead in computation time. Furthermore, it is the goal of the conducted experiments to show an improved user interaction in terms of user response time and decision accuracy in regards to multi-objective optimization strategies without a loss of quality in terms of the results and without a generated computational overhead.

4.1 Normalized Weighted Sum Algorithm

4.1.1 Simulation Model

In the described mobile-cloud database environment, each QEP consists of three costs: monetary cost for using the cloud provider, query execution time as time to run a query plan, and energy consumption on the mobile device. The last cost becomes important under the condition of using a cache on the mobile device to have the option of receiving partial or total requested data from the mobile device itself [7]. This obviously results in a lower monetary cost since the cloud

provider is less or not used, but also results in a higher amount of consumed energy since processing the cache consumes more energy than waiting for data to be retrieved from the cloud database. A full review of such a system is given in [5].

The simulation to test the impact of NWSA in regards to single-objective optimization is built as follows: the simulation consists of one million experiments, where the proposed NWSA as well as the lexicographical ordering strategy have to choose a single QEP out of a set of 20 QEPs. The cost of each QEP is generated randomly within the following ranges: Monetary Cost (M) has a range of 0 up to 10 cents and was chosen according to the current Amazon EC2 pricing models [2]; the range for query execution time was selected to be between 0 and 10 seconds (including data transfer time), and energy between 0 and 0.5 mAh. This simulation is repeated for multiple weight compositions.

4.1.2 Simulation Results

In comparison to the lexicographical ordering strategy the experimental results show two facts. First, the NWSA computes the same results under the same costs as the lexicographical ordering when focusing on only one objective. Second, NWSA produces negligible overhead in computing this selection. As it was already discussed in the previous sections, 2.2.1 and 3.1, both algorithms are running linear execution time related to the size of QEPs to choose from. That leads to a total algorithm execution time of less than one millisecond per experiment for both algorithms so that the difference is negligible. Concluding this comparison, negligible overhead is incurred and no higher cost alternatives results are selected.

Looking at the performance of NWSA, this evaluation shows the increased possibilities of this strategy to select QEPs with their trade-offs.

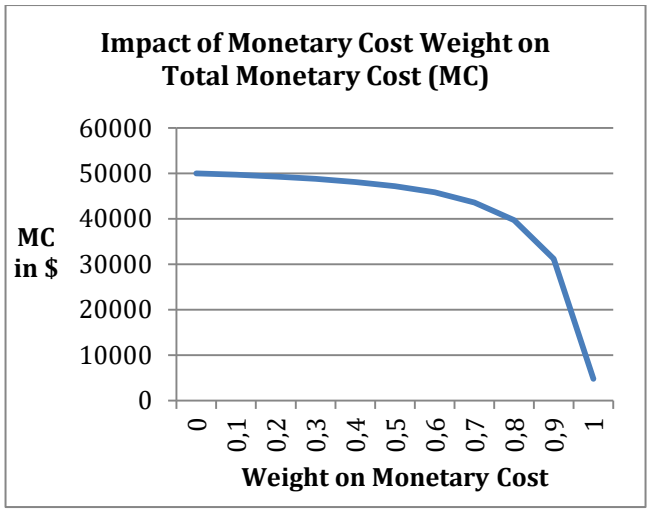


Figure 19: Impact of Monetary Cost Weight on Total Monetary Cost of QEPs selected by NWSA

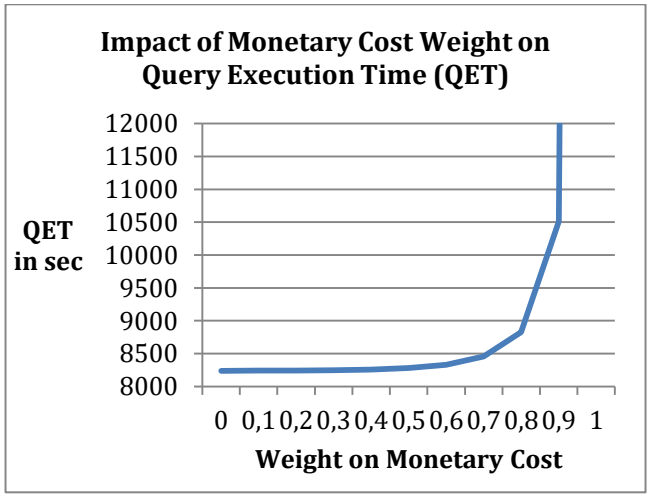


Figure 20: Impact of Monetary Cost Weight on Total Execution Time of QEPs selected by NWSA

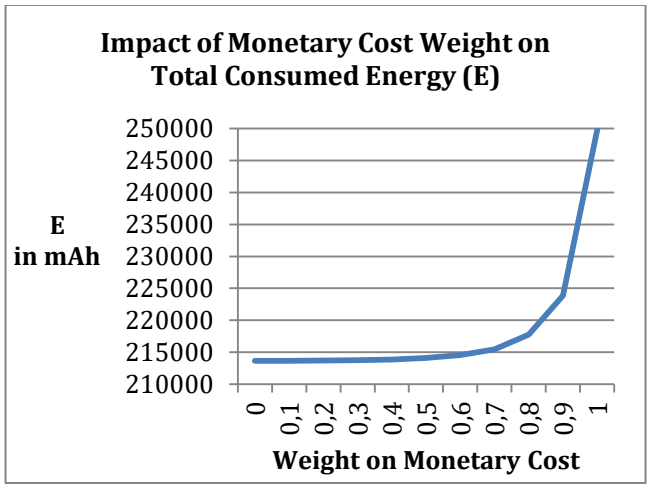


Figure 21: Impact of Monetary Cost Weight on Total Consumed Energy of QEPs selected by NWSA

The different options on how to stress the weights on the different objectives can change the total cost in terms of monetary cost, execution time and energy consumption by a large margin as it can be seen in Figure 19, Figure 20, and Figure 21. The figures show the changes of the total cost of the one million chosen QEPs as the weight on monetary cost increases. The remaining weight is divided equally between execution time and energy consumption.

It can be seen that when the monetary cost weight increases, the monetary costs decreases, while the query execution time and energy consumption increase. It is notable that the minimum and maximum values of an objective span a large gap, so the impact of having weights is easily seen. Having a small weight on one objective can lead to a big difference in the total cost.

It is notable that a consistent change in weights does not lead to an even distribution of points in the Pareto-Set. This well-known problem is analyzed by Das and Dennis [36].

4.2 User Study on Optimization Strategies

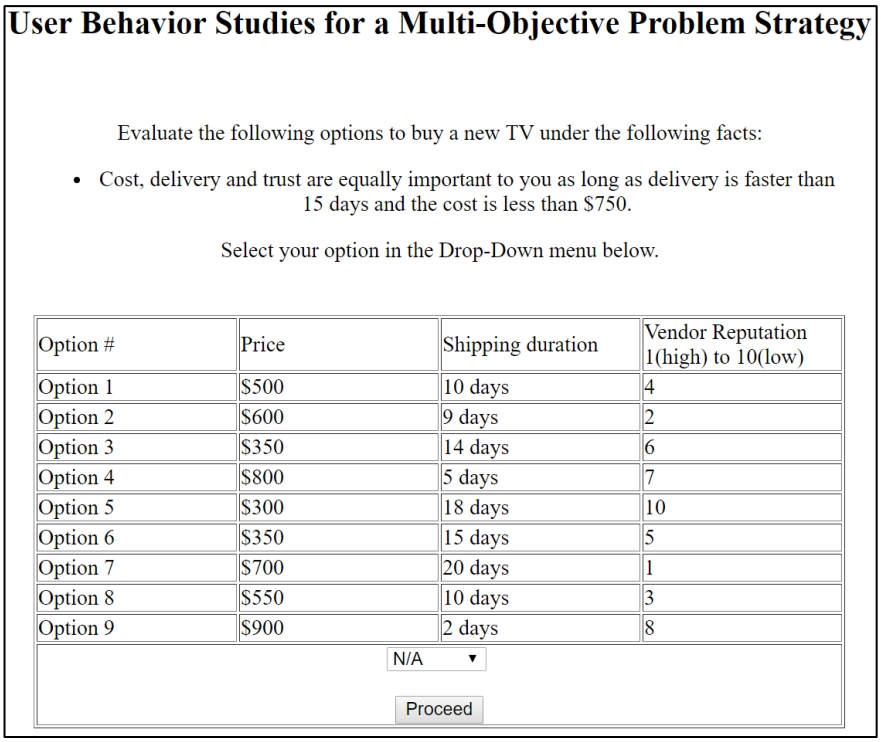
This section describes an evaluation of the difference between the user decision on a single QEP in the Skyline/Pareto Set model and the weight profile selection process needed by the NWSA. The goal of this user study is to compare the three approaches, the Skyline Query selection, the weight selection process in NWSA, and the preset weight profile selection of NWSA, in terms of the accuracy of the decision that a user makes and the amount of time the user needs to make such a decision.

4.2.1 Simulation Model

The participants of this user study, volunteers with and without a background in computer science, were given three sets of questions representing the decision a user has to make in the Skyline approach by selecting an alternative based on the given Pareto-Set (an example can be seen in Figure 22), the decision a user has to make in the NWSA approach by selecting weights to stress the different objectives (an example can be seen in Figure 23), and the decision based on the preset weight profiles including a logical description (an example can be seen in Figure 24). The sets appeared in alternating order for different participants of the study to remove any bias towards any of the three approaches because of the order of the sets.

This study has been transposed to an easy equivalent multi-objective question so that no specific knowledge or large introduction to the field was needed. In each question, users are asked to select one alternative to buy a TV based on the given alternatives. In this transposition, a TV is equivalent to a QEP,

the monetary cost to purchase a TV is equivalent to the monetary execution cost, the delivery time is equivalent to the execution time, and the vendor reputation is equivalent to the energy consumption. With each question in a set, users are presented with 3, 5, 7, and 9 alternatives to choose from.



User Behavior Studies for a Multi-Objective Problem Strategy

Evaluate the following options to buy a new TV under the following facts:

- Cost, delivery and trust are equally important to you as long as delivery is faster than 15 days and the cost is less than $750.

Select your option in the Drop-Down menu below.

Option #	Price	Shipping duration	Vendor Reputation 1(high) to 10(low)
Option 1	$500	10 days	4
Option 2	$600	9 days	2
Option 3	$350	14 days	6
Option 4	$800	5 days	7
Option 5	$300	18 days	10
Option 6	$350	15 days	5
Option 7	$700	20 days	1
Option 8	$550	10 days	3
Option 9	$900	2 days	8

N/A

Proceed

Figure 22: User Study Set 1 representing the Skyline approach

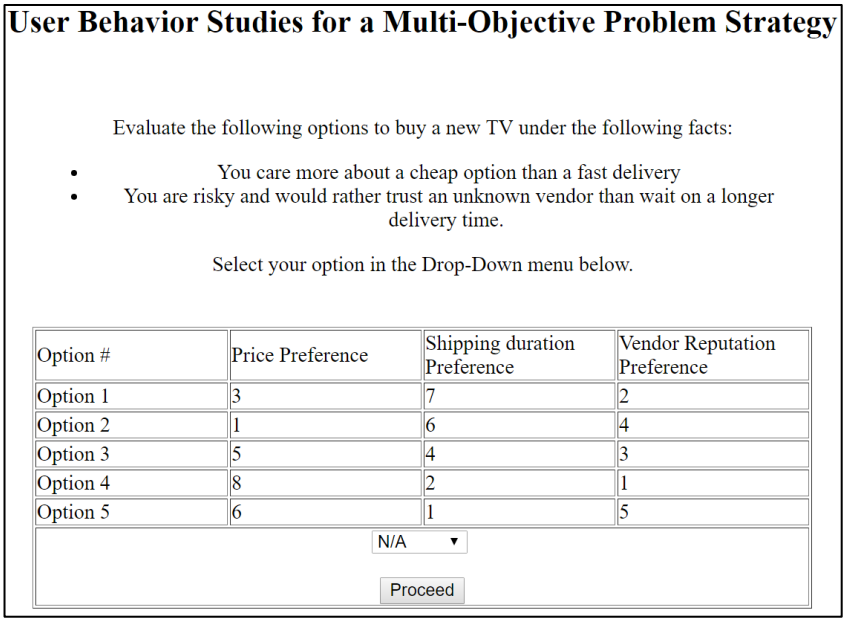


Figure 23: User Study Set 2 representing the NWSA approach with weight profiles

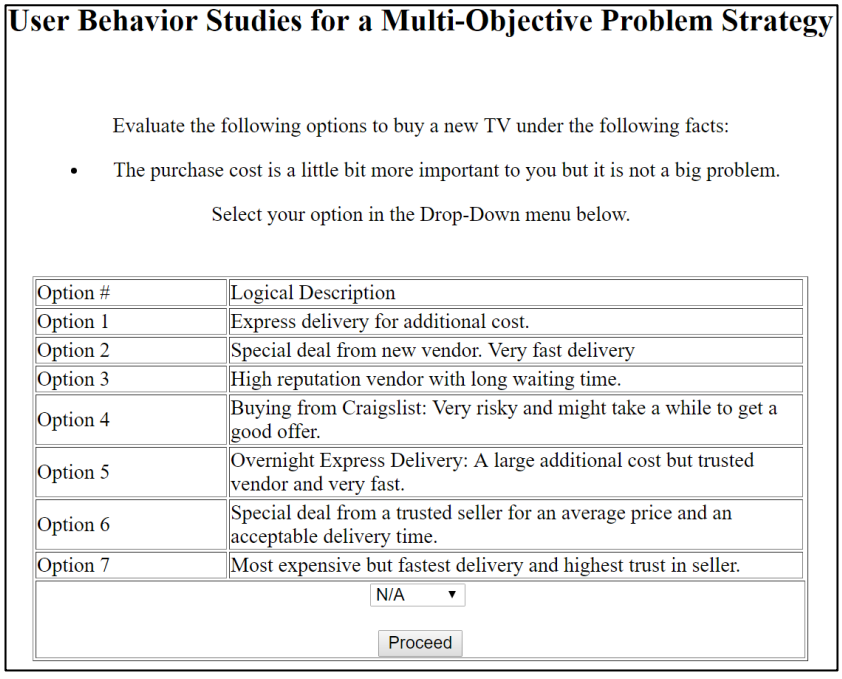


Figure 24: User Study Set 3 representing the NWSA approach with logical descriptions

4.2.2 Simulation Results

The preliminary results of the user study show that the preset weight profile selection of NWSA given a logical description of a weight profile is by far the easiest decision in terms of both the time a user takes to make a decision and the accuracy of the decision. The participants in the study answered those decision questions in average nearly twice as fast (~42 seconds) as the decision with a given weight profile without logical description (~80 seconds). The participants needed similar time for selecting one solution out of the list of alternatives in the Skyline approach (~85 seconds). The accuracy of selecting the optimal answer was low for both the Skyline approach as well as the weight profile selection without logical description (both < 50%). In contrast to that, the participants selected the optimal alternative with accuracy greater than 80% given the logical descriptions from the study set 3. Furthermore, giving a participant more than five alternatives to choose from in the Skyline approach or in the NWSA approach without a logical description of weight profiles increases the time needed to make a decision significantly (increase of ~40%). Given a logical description of the weight profile in the NWSA approach reduced this increase to only 10% more time to answer a question of 7 or 9 alternatives compared to 5 given alternatives.

4.3 Performance study on the NWSA based scheduling algorithm

This section describes a performance study of the difference between the scheduling algorithm by Kllapi [34] (further referenced as Kllapi) and the proposed NWSA based scheduling algorithm (Section 3.3). There are two goals for this performance study. The first goal is to show that there is no loss in quality of

the computed QEPs by using NWSA. The second goal is to show that there is no generated overhead by using NWSA. We used HIVE [37] as the underlying database system for all experiments. Furthermore, we reduced the cost-dimensions of a QEP to monetary cost and execution time since the experiment was conducted on the cloud without mobile interaction and therefore without energy consumption on a mobile device. This reduction has no impact on the scheduler itself.

4.3.1 Simulation Model: Quality of the QEPs

For this simulation model, we use the database schema and queries of the TPC-H benchmark [38]. We generated a data size of 2GB for the database using the given TPC-H dbgen. The simulation model was set for 20 containers, using two of each container-type specified in Figure 25.

Container-Type	Speed (in bytes/sec)	monetary cost (in $/sec)
1	3162277.5	1.0E-7
2	1.0 E14	1.0E-6
3	1.4 E14	1.9E-6
4	1.7 E14	2.8E-6
5	1.9 E14	3.7E-6
6	2.1 E14	4.6E-6
7	2.4 E13	5.5E-6
8	2.5 E13	5.5E-6
9	2.7 E13	7.3E-6
10	2.9 E13	8.2E-6

Figure 25: Container Specification

We executed multiple instances for each of the 22 TPC-H queries, generated by TPC-H's qgen, and compared the resulting QEPs of Kllapi to the resulting QEPs of NWSA. Since each instance of NWSA with a given weight profile results in a single QEP, we combined QEPs over different weight profiles defined as <x, 1-x>

for <weight_Time, weight_Money> with an increasing x of 0.05 to receive a curve of the computed QEPs.

It is the goal of this simulation to show that the QEPs generated by Kllapi and generated by NWSA are equal.

4.3.2 Simulation Model: Computation time of the schedulers Kllapi [34] and NWSA

It is the goal of this simulation to show that the NWSA scheduler does not generate a significant overhead compared to the Kllapi scheduler.

For this simulation model, we use the database schema and queries of the TPC-H benchmark [38]. This simulation has the purpose of exploring the effects of the parameters on the execution time of Kllapi and NWSA. To first show the differences of both schedulers under the average conditions (average database size and average number of containers), we show an extensive simulation of randomized types of the TPC-H queries. 2GB was selected as the average size of the database since the size of the database does not have a significant influence on the schedulers' execution time (later proven in Section 4.3.4). Furthermore, a container limit of 20 containers was used since this was the maximum achieved parallelism without restricting the number of containers. Since both schedulers are dependent on the number of operators and containers, we modified the number of containers (from 4-100) and the underlying size of the database (2-10GB) for a second and third experiment..

4.3.3 Simulation Results: Quality of the QEPs

Figure 26 and Figure 27 show the generated QEPs when using the NWSA scheduler and the Kllapi scheduler, respectively, on the TPC-H Query 1(Q1). It can be observed that the generated QEPs using NWSA are the elements of the generated QEPS using Kllapi. Furthermore, it can be observed that Kllapi generated 3 additional QEPs. This is the result of increasing the weight profile <x, 1-x> in steps of 0.05. Since the missing QEPs are close to the other generated QEPs, the size of the steps is too large to catch these. Decreasing the size of the steps would generate the remaining QEPs. The experiments for the TPC-H Queries 2-22 (Q2-Q22) show identical results.

The experiment shows that NWSA does not compute QEPs with a loss of quality in regards to the computed QEPs by Kllapi. There is no loss of QEPs and furthermore no QEPs in NWSA that are dominated by QEPs in Kllapi.

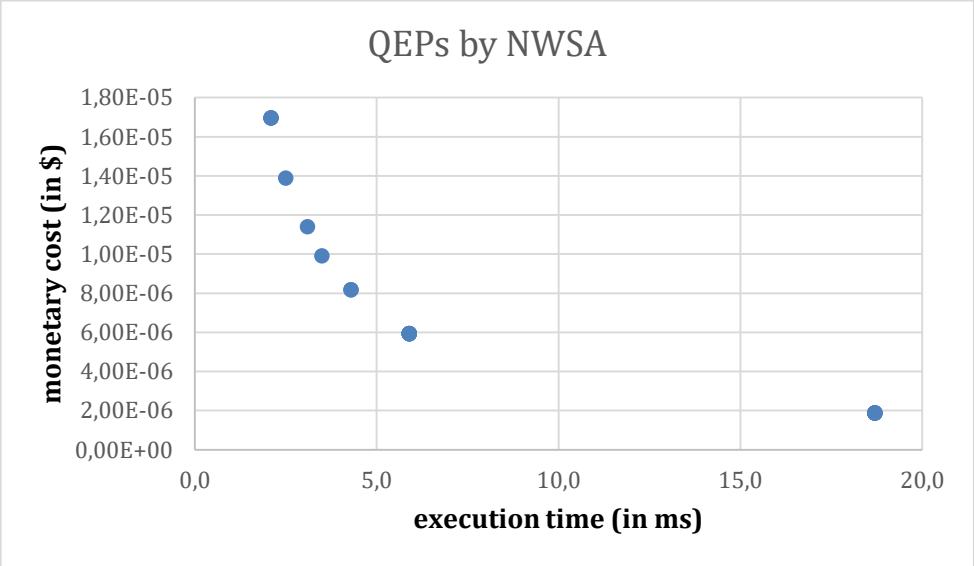


Figure 26: Generated QEPs for TPC-H (Q1) by NWSA Scheduler

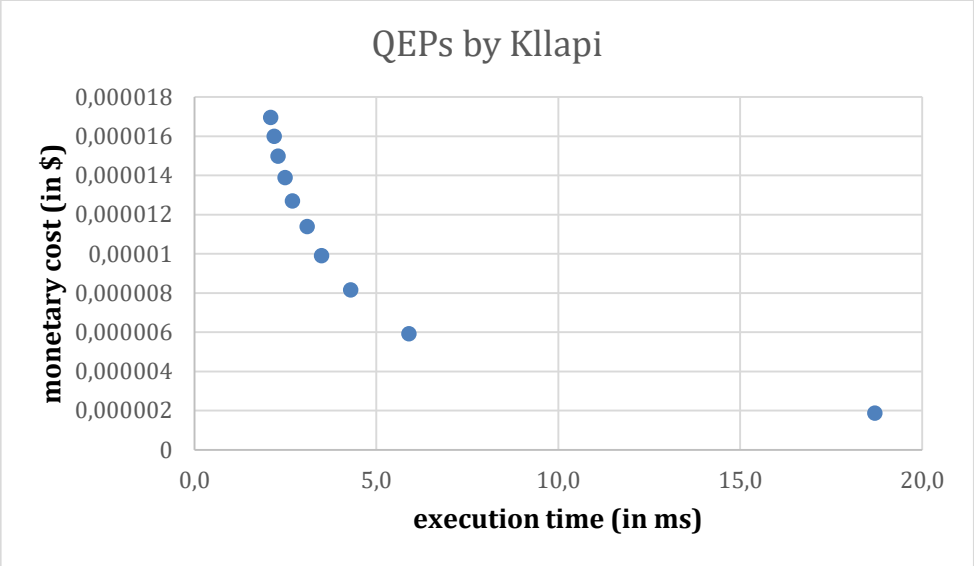


Figure 27: Generated QEPs for TPC-H (Q1) by Kllapi Scheduler

4.3.4 Simulation Results: Computation time of the schedulers Kllapi [34] and NWSA

Figure 28 and Figure 29 show the average execution time of Kllapi and NWSA, respectively, when executing a batch of 500 instances of type randomized TPC-H queries. It can be observed that the NWSA scheduler needs in average about 200 ms more to process a batch of 500 queries with heterogeneous containers and about 350 ms more to process a batch of 500 queries with homogeneous containers. This difference is explainable by the needed computation of a score for each operator within NWSA. Since the additional overhead per single query is in average less than one millisecond, we can consider this overhead negligible.

Batch(500 queries)	Kllapi (in ms)	NWSA (in ms)
Batch 1	1476	1753
Batch 2	1705	1935
Batch 3	1442	1789
Batch 4	1567	1631
Batch 5	1581	1896
Batch 6	1480	1671
Batch 7	1552	1646
Batch 8	1489	1529
Batch 9	1569	1870
Batch 10	1422	1619
Batch 11	1523	1797
Batch 12	1436	1708
Batch 13	1502	1734
Batch 14	1398	1772
Batch 15	1600	1839
Batch 16	1344	1587
Batch 17	1372	1522
Batch 18	1569	1656
Batch 19	1544	1569
Batch 20	1383	1562
Average per batch	1498	1704
Average per query	2.9954	3.4086

Figure 28: Average execution time of the Kllapi and NWSA Scheduling algorithms on Heterogeneous Containers

Batch(500 queries)	Kllapi (in ms)	NWSA (in ms)
Batch 1	1063	1395
Batch 2	1068	1416
Batch 3	958	1348
Batch 4	988	1338
Batch 5	1183	1530
Batch 6	1090	1466
Batch 7	1077	1495
Batch 8	969	1334
Batch 9	910	1264
Batch 10	1028	1424
Batch 11	1089	1402
Batch 12	987	1312
Batch 13	991	1387
Batch 14	934	1256
Batch 15	977	1278
Batch 16	937	1238
Batch 17	944	1213
Batch 18	1058	1440
Batch 19	955	1311
Batch 20	950	1305
Average per batch	1008	1357
Average per query	2.016	2.715

Figure 29: Average execution time of the Kllapi and NWSA Scheduling algorithms on Homogeneous Containers

Figure 30 and Figure 31 show execution times of Kllapi and NWSA when executing a batch of 500 instances of type randomized TPC-H queries with increasing number of containers. Since the additional overhead per query is in average less than one millisecond, we can consider this overhead negligible. Furthermore, given the TPC-H queries, the resulting QEPs only achieve a maximum parallelism of up to 20 containers. Up to 20 containers, the additional overhead per query is barely measurable.

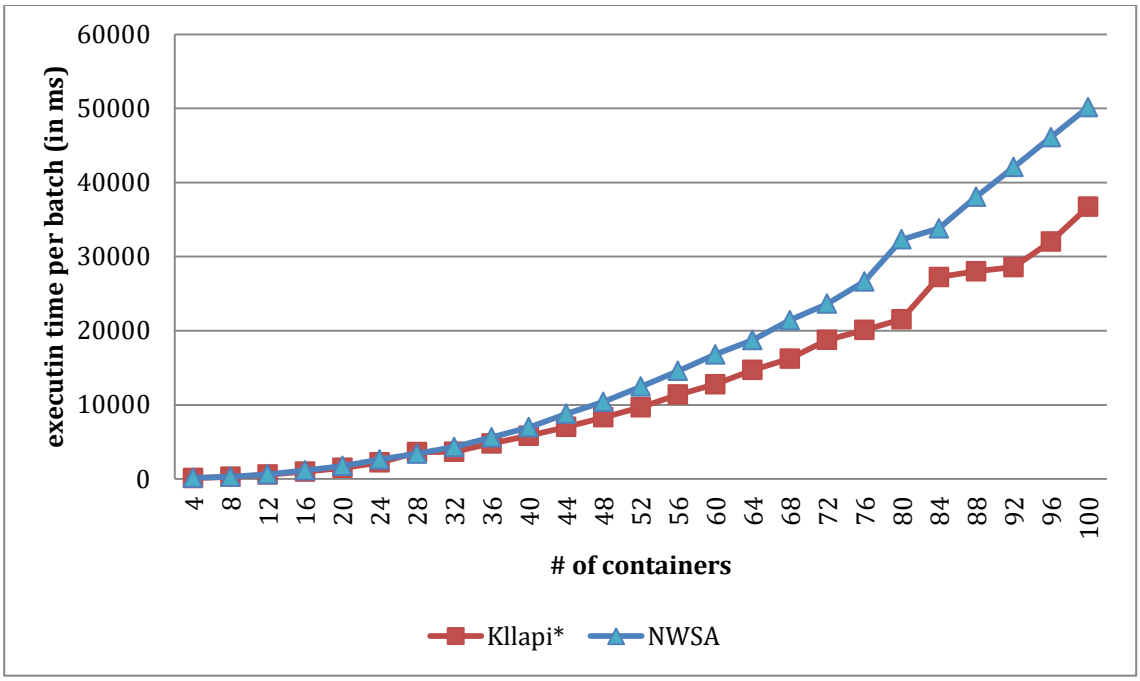


Figure 30: Influence of the number of heterogeneous containers on the execution time of the Kllapi and NSWA Scheduling algorithms

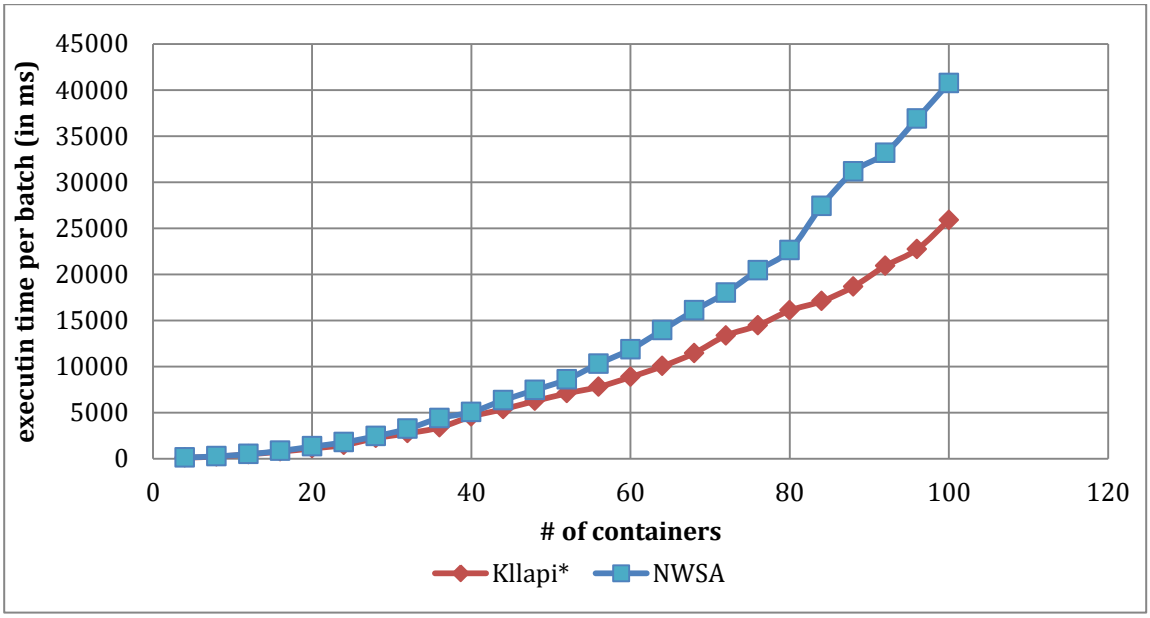


Figure 31: Influence of the number of homogeneous containers on the execution time of the Kllapi and NWSA Scheduling algorithms

Last, we studied the effects of increasing the size of the database on the schedulers' execution time. As represented in Figure 32, the size of the database

does not seem to have a relevant effect on the generated overhead. The averaged overhead per query is less than one millisecond and is therefore negligible.

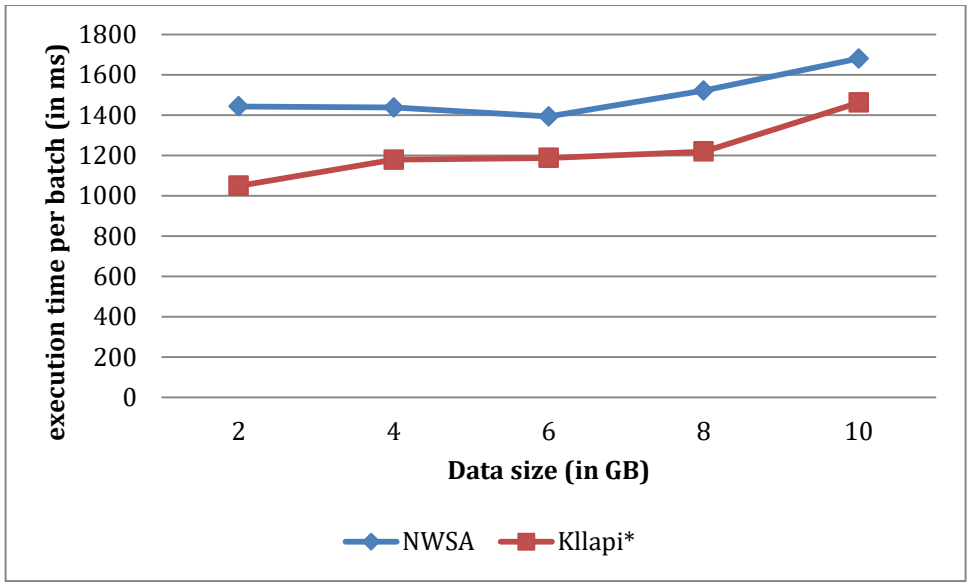


Figure 32: Influence of the database size on the execution time of the Kllapi and NWSA Scheduling algorithms

All conducted experiments show negligible overhead generated by the NWSA scheduler in regards to the Kllapi scheduler [34].

4.4 Summary of Experiment Results

The first overall goal of the conducted experiments was to show the increased quality of QEP results produced by the proposed model, NWSA, compared to the single-objective optimization strategies without an additional overhead in computation time. This hypothesis is confirmed by the results of the experiments reported in Section 4.1.

Furthermore, it was the goal of the conducted experiments to show an improved user interaction in terms of user response time and decision accuracy compared to the multi-objective optimization strategies without a loss of quality in

terms of the results and without a generated computational overhead. The results of the first part are shown in the experiments reported in Section 4.2, where the user case study shows an improved accuracy and response time of a user when selecting the logical descriptions of the weight profiles over the weight profiles or the query execution plans. The second part was shown in the experiments reported in Section 4.3, where the proposed scheduling NWSA algorithm (NWSA-S) was compared with the multi-objective optimization scheduling algorithm by Kllapi [34]. NWSA-S does not generate a significant overhead and has no loss in quality of the computed QEPs.

Chapter 5: Conclusion and Future Work

This thesis presents a multi-objective optimization (MOO) strategy, named Normalized Weighted Sum Algorithm (NWSA). It is its goal to solve the multi-objective optimization problem with the use of user preferences on optimization objectives (weight profile). The experiments evaluating NWSA in the context of a mobile-cloud query optimization have been presented. NWSA is able to select the query execution plan that is an element of the Pareto set, while avoiding the expensive cost of computing the Pareto set. NWSA is highly adaptable to any multi-objective decision problem since it is not limited to any number of objectives as pointed out in section 3.1. The experimental results show that NWSA incurs negligible computational overhead in comparison to the existing lexicographical ordering strategy.

Furthermore, this thesis presents a new user interaction model for NWSA and MOO strategies, introducing the user-types of *query executing users* and *superusers*. With superusers, who are aware of query execution constraints, presetting weight profiles, query executing users can select these weight profiles and are not burdened with the decision on a query execution plan (QEP). These QEPs are generated in the process of multi-objective query optimizations (MOQO) in terms of monetary cost, execution time, and energy consumption of a query. This model was compared with the existing user-interaction of the Skyline/Pareto Set approach within a user study. The comparison shows that the user interaction of deciding on a Pareto optimal QEP, which is necessary while using the Skyline approach, can be eliminated by using NWSA. The user study shows that using a

logical description of a weight profile substantially increases the accuracy of a query executing user selecting the optimal alternative and also speeds up the time a user needs to select his/her answer. These weight profiles can be preset by a superuser, calculated based on possible constraints. This process can be done using the superuser interface we have developed and presented in this thesis.

Finally, this thesis presents a scheduler for query processing based on NWSA (called NWSA-S). With the implementation of NWSA-S, query processing utilizes the previously explained improvements in user interactions with MOQO. Experiments, in which NWSA-S is compared to a multi-objective Pareto-Set based scheduler by Kllapi [34] show no significant generated overhead in execution time of the scheduler and also shows no loss in quality of the computed QEPs, leaving the improved user interaction as benefit without additional cost.

5.1 Future Work

As shown in Section 3.3.1, the developed greedy-based scheduler NWSA-S has a significant trade-off between computational complexity and finding the global optimum. Evolutionary algorithms [10] or heuristics [18] can potentially improve the quality of this algorithm. These strategies are able to search for the global optimum which would increase the quality of the selected QEP.

Another future work is the integration of NWSA in a cache replacement policy to extend semantic caching [7]. Based on the computed score of a QEP, the new policy can help to keep more valuable data in the semantic cache [39]. The higher the score of data in the cache, the higher the cost to regain those results will

be (the recovery cost value). This could replace the often used metrics of minimizing cache misses.

Finally, the existing mobile prototype for query executing users needs to be assembled with the query processing prototype on the cloud site. The interfaces of the query processing prototype for entering queries and the mobile prototype for executing a query are highly adaptable, based on the fact that the interfaces of both components match each other. This process also enables simulations to include energy consumption as an objective for the scheduler.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[2] Amazon, "Amazon Web Service prices - EC2 - On Demand Pricing," 2016. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/?nc1=h_ls. [Accessed 01 08 2016].

[3] N. Fernando, S. W. Loke and W. Rahayu, "Mobile cloud computing: A survey," in *Future Generation Computer Systems*, 2013.

[4] H. T. Dinh, C. Lee, D. Niyato and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," in *Wireless communications and mobile computing 13.18*, 2013.

[5] J. Mullen, M. Perrin, F. Helff, L. Gruenwald and L. d'Orazio, "A Vision of Time-, Energy-, and Monetary Cost-Aware Query Processing in Mobile Cloud Database Environment," March 2015.

[6] Microsoft, "Microsoft Azure Pricing," [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/cloud-services/. [Accessed 20 11 2016].

[7] M. Perrin, "Time-, Energy-, and Monetary Cost-Aware Cache Design for a Mobile Cloud Database System," May 2015.

[8] H. Garcia-Molina, J. D. Ullman and J. Widom, Database Systems, The Complete Book, 2002.

[9] A. Ben-Tal, Characterization of Pareto and lexicographic optimal solutions, Springer Berlin Heidelberg, 1980.

[10] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," in *IEEE Transactions On Evolutionary Computation*, 1999.

[11] C. H. Papadimitriou and M. Yannakakis, "Multiobjective Query Optimization," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2001.

[12] J. Chomicki, P. Ciaccia and N. Meneghetti, "Skyline queries, front and back," in *ACM SIGMOD Record 42.3*, 2013.

[13] D. Kossmann, F. Ramsak and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *Proceedings of the 28th international conference on Very Large Data Bases (VLDB)*, 2002.

[14] D. Papadias, Y. Tao, G. Fu and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data (ACM)*, 2003.

[15] J. Lee, G.-w. You, S.-w. Hwang, J. Selke and W.-T. Balke, "Interactive skyline queries," in *Information Sciences*, 2012.

[16] S. Borzsony, D. Kossmann and K. Stocker, "The skyline operator," in *17th International Conference on Data Engineering*, 2001.

[17] C. Lei, Z. Zhuang, E. A. Rundensteiner and M. Eltabakh, "Shared Execution of Recurring Workloads in MapReduce∗," in *Proceedings of the VLDB*

Endowment, 2015.

[18] I. Trummer and C. Koch, "A Fast Randomized Algorithm for Multi-Objective Query Optimization," in *SIGMOD*, 2016.

[19] I. Trummer and C. Koch, "Multi-Objective Parametric Query Optimization," in *Proceedings of the VLDB Endowment*, 2014.

[20] Y. Tao, X. Xiao and J. Pei, "Efficient Skyline and Top-k Retrieval in Subspaces," in *IEEE Transactions on Knowledge and Data Engineering*, 2007.

[21] J. L. Bentley, H. T. Kung, M. Schkolnick and C. D. Thompson, "On the average number of maxima in a set of vectors and applications," in *Journal of the ACM (JACM)*, 1978.

[22] D. V. Lindley, Scoring Rules and the Inevitability of Probability, 50 ed., International Statistical Review Vol 50, 1982, pp. 1-11.

[23] F. Helff, L. Gruenwald and L. d'Orazio, "Weighted Sum Model for Multi-Objective Query Optimization for Mobile-Cloud Database Environments," in *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference EDBT/ICDT 2016*, Bordeaux, France, 2016.

[24] E. Triantaphyllou, Multi-criteria decision making methods: a comparative study, 44 ed., Springer Science & Business Media, 2013.

[25] I. Kim and O. de Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," in *Structural and multidisciplinary optimization 29.2* , 2005, pp. 149-158.

[26] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," in *Structural and multidisciplinary optimization*, 2010.

[27] T. Shibata and T. Ohmi, "A functional MOS transistor featuring gate-level weighted sum and threshold operations," in *IEEE Transactions on Electron devices*, 1992.

[28] M. G. M. Hunink und M. C. Weinstein, Decision Making in Health and Medicine: Integrating Evidence and Values, 2nd Hrsg., Cambridge University Press, 2014.

[29] P. Kind, J. E. Lafata, K. Matuszewsk and D. Raisch, "The use of QALYs in clinical and patient decision-making: Issues and prospects," in *Value In Health*, 2009.

[30] P. C. Fishburn, "Methods of estimating additive utilities," in *Management science 13.7*, 1967.

[31] C.-H. Goh, Y.-C. A. Tung and C.-H. Cheng, " A revised weighted sum decision model for robot selection," in *Computers & Industrial Engineering Vol.30(2)*, 1996.

[32] L. Zadeh, "Optimality and non-scalar-valued performance criteria," in *IEEE transactions on Automatic Control*, 1963.

[33] A. Inselberg, Parallel Coordinates: Visual Multidimensional Geometry and Its Applications, Springer, 2009.

[34] H. Kllapi, E. Sitaridi and M. M. Tsangaris, "Schedule Optimization for Data Processing Flows on the Cloud," in *Proceedings of the 2011 ACM SIGMOD*

International Conference on Management of data (ACM), 2011.

[35] P. E. Black, Dictionary of algorithms and data structures, National Institute of Standards and Technology, 2004.

[36] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems," in *Structural optimization*, 1997.

[37] "Apache HIVE," [Online]. Available: http://hive.apache.org/. [Accessed 20 11 2016].

[38] Transaction Processing Performance Council (TPC), "TPC BENCHMARK H: Revision 2.17.1," 2014.

[39] J. Jeong and M. Dubois, "Cost-sensitive cache replacement algorithms," in *High-Performance Computer Architecture*, 2003.