Finding the Non-dominated Set of the Bi-objective Quadratic Knapsack Problem

by

Aslı Pınar Yapıcı

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

Koç University

2013

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Aslı Pınar Yapıcı

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Prof. Dr. Serpil Sayın

Assoc. Prof. Dr. Emre Alper Yıldırım

Asst. Prof. Dr. Yalcın Akçay

Date:

ABSTRACT

Multi-objective optimization aims to model and solve real life problems. In bi-objective problems, as in the case of multi-objective problems, non-dominated solutions are of interest. The bi-objective linear knapsack problem is well-studied and is known to be a difficult problem with a lot of non-dominated solutions. We have studied the quadratic version of this problem, the bi-objective quadratic knapsack problem (BQKP). Our goal is to investigate the structure of the non-dominated set in BQKP since reports on implementations in the literature are inadequate so far. We use the epsilon constraint method to find non-dominated solutions and use a linear scalarization to obtain supported non-dominated solutions.

Randomly generated BQKP's with different percentage of fullness values are solved on the commercial solver CPLEX and results of these problems are reported with illustrations and coverage errors. BQKP's with small percentage of fullness levels resembles more to the linear version of the problem with higher number of non-dominated solutions. We observe that the number of non-dominated solutions is not many for the size of BQKP's we had solved yet the problem is still hard to solve since some of the test problems required long cpu times. Supported non-dominated solutions are relatively easier to obtain by using a weighted sum scalarization. When supported non-dominated solutions are regarded as a representation of the entire non-dominated set, averages coverage errors are at acceptable levels for all problem types studied in this work. BQKP's with tri-diagonal quadratic matrices are also solved and reported. Tri-diagonally structured problems are easier to solve and require less computation time.

ÖZETÇE

Çok amaçlı optimizasyon gerçek hayat problemlerini modellemeyi ve çözmeyi amaçlar. İki amaç fonksiyonlu problemlerde de çok amaçlı optimizasyonda olduğu gibi etkin çözümler bulunmaya çalışılır. İki amaç fonksiyonlu doğrusal sırt çantası problemi çok sayıda etkin çözümü bulunan ve zor kabul edilen bir problemdir. Bu problemin karesel versiyonu olan iki amaç fonksiyonlu karesel sırt çantası problemini (İKSP) çalıştık. Bu problem üzerine literatürdeki uygulamalar yetersiz olduğundan etkin çözüm kümesinin yapısının araştırılmasını hedefledik. Etkin çözümler kümesini bulmak için epsilon kısıt yöntemini, destekli etkin çözümleri bulmak için ise bir doğrusal skalarizasyon yöntemi kullandık.

Farklı doluluk değerleri ile rassal yaratılan İKSP'leri CPLEX ile çözüp sonuçları illüstrasyonlar ve kapsama hataları ile raporladık. Doluluk değerleri düşükken doğrusal versiyona yaklaşan İKSP'ler daha çok sayıda etkin çözüme sahiptir. Çözdüğümüz problemler için etkin çözüm sayılarının çok fazla olmadığını gene de yüksek cpu zamanları gerektirdikleri için zor problemler olduklarını belirledik. Destekli etkin çözümlerin ağırlıklı doğrusal skalarizasyon ile bulunmasının daha kolay olduğunu söyleyebiliriz. Bu çözümler tüm çözüm kümesinin bir temsili olarak alındığında ortalama kapsama hatalarının kabul edilebilir seviyede olduğu görülebilir. Bunların yanında, üç köşegenli matrislerle oluşturulmuş İKSP'leri de çözüp raporladık. Bu problemlerin çözümü ise daha kolaydı ve daha az çözüm zamanı gerektirdi.

ACKNOWLEDGMENTS

I want to thank all the people who supported me by any means through this period; my professors especially Serpil Sayın, my family, my friends and my colleagues.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Problems in the optimization theory deal with the optimization of an objective function with respect to a set of constraints or restrictions. The aim here is to find solution which will give the maximum or minimum value of the objective function according to the problem type while satisfying constraints. This solution is called the $optimal solution$ of the problem.

In real life, many aspects are needed to be considered and included in the design and solution process of the problems. In the optimization literature, *multi-objective optimization* addresses this issue. The multi-objective optimization tries to optimize more than one objective function at the same time and finds non-dominated solutions rather than the optimal ones. Optimal solution concept is converted into the non-dominated solution concept in multi-objective problems.

The *knapsack problem* and the *quadratic knapsack problem* are important problems in the optimization literature. The classical *knapsack problem* consists of a linear objective function, a linear capacity constraint and binary variables. The *quadratic knapsack problem* has a quadratic objective rather than a linear one. These problems are well studied and have applications in diverse areas. Both problems have a simple structure yet are hard to solve at the same time.

We have chosen to study a topic that had blended multi-objective optimization and the quadratic knapsack problem in this thesis. *Bi-objective quadratic knapsack problem* (BQKP) is a hard problem which is also applicable to many real life problems. BQKP has its significance since it is non-convex, quadratic and has more than one objective function to consider.

To the best of our knowledge, studies which address this problem directly are very few. Therefore, in this work, we aimed to answer several questions related to BQKP and thus understand the characteristics of BQKP in general.
In this manner, random test problems of different sizes and percentage of fullness of quadratic matrices are generated. The number of non-supported and the supported solutions thereby the number of non-dominated solutions of these test problems are found and reported. Coverage errors are calculated for these two sets of solutions. Moreover, illustrations of the non-dominated set and comparison of the non-dominated sets of bi-objective linear and quadratic knapsack problems are given for small sized problems.
In order to solve BQKP, two fundamental solution methods from the multi-objective optimization literature are incorporated. In addition, a suitable linearization method is applied to BQKP's with large sizes since solution times get unreasonably long.
An extensive literature survey is also given on the subject.
Basic concepts related to our study is explained in the following. In Chapter 2, the background is provided. Then in Chapter 3, the formulation of BQKP and solution techniques applied to the problem are explained. In Chapter 4, we described how these techniques are implemented for BQKP in detail. The results, comments and illustrations are given in Chapter 5. Finally, Chapter 6 gives some insights about further research on BQKP.

Chapter 2

BACKGROUND

This chapter gives basic information on multi-objective optimization including the definitions, solution techniques and applications. Also, the *knapsack problem* and the *quadratic knapsack problem* will be defined in order to fully understand the *bi-objective quadratic knapsack problem* studied in this thesis.

2.1 Multi-objective Optimization (MOO)

Single objective optimization techniques are not adequate to model and solve most real-life problems. In order to get more realistic mathematical programming representations of these problems, more than one objective needs to be considered. Therefore, an important field of optimization theory named the multi-objective (or multi-criteria) optimization (MOO) arose which usually deals with several objectives in conflicting nature.

Most of the optimization problems can be considered with multiple objectives. For instance, the classical examples of combinatorial optimization; the shortest path, minimum spanning tree, assignment, knapsack, traveling salesperson, or set covering problems all have multi-objective versions [72]. These problems are generalized to the multi-objective case since there is the need to consider more than one aspects in many realizations.

A large literature on MOO exists since it has wide range of applications and has been studied extensively by many researchers in the last four decades.

The multi-objective optimization problem (MOP) can be defined with the following formulation,

$$\begin{aligned} &\textit{maximize} \quad (f_1(x), f_2(x), ..., f_p(x)) \\ &\textit{subject to} \quad x \in X. \end{aligned} \tag{2.1}$$

Here $f_i : \Re^n \to \Re$ for $i = 1, ..., p$ are simultaneously maximized, and $f(x) = (f_1(x), f_2(x), ..., f_p(x))^T$ denotes the objective function vector. $\Re^n$ and $\Re^p$ are finite-dimensional Euclidean vector spaces. The *decision (variable) vectors* $x = (x_1, ..., x_n)^T$ belong to the *decision space* $X \subseteq \Re^n$. $Z := f(X)$ is the *objective (or criterion) space.* For $p = 2$, this problem is referred as the *bi-objective problem* (BOP).
Fundamental concepts of MOO are defined in the following subsection.

2.1.1 Basic Concepts of MOO

The fundamental conceptual difference between MOO and the classical single objective optimization is in the notion of optimality. For multi-objective problems the term efficient solution takes the place of the term optimality for single objective problems.

Definition 1. *$x \in X$ is said to be an efficient or Pareto optimal solution if there exists no $x^* \in X$ satisfying $f_i(x^*) \geq f_i(x)$ for all i and $f_i(x^*) > f_i(x)$ for at least one i.*

The objective vector $f(x) = (f_1(x), f_2(x), ..., f_p(x))^T$ is said to be *non-dominated* if x is efficient and *dominated* if x is not efficient (is inefficient). The set of all efficient solutions generates the *efficient set* denoted by X_E, and the image of X_E in the objective space, the *non-dominated set*, is denoted by Z_E. The set of all non-dominated vectors are also referred to as the *efficient frontier.*

Definition 2. *$x \in X$ is said to be a weakly efficient solution if there exists no $x^* \in X$ satisfying $f_i(x^*) > f_i(x)$ for all i.*

Weakly efficient solutions are indeed not desirable. However, some methods may be delivering weakly efficient solutions rather than efficient solutions due to some

technicalities. These solutions can then be eliminated.
When the feasible set of MOP is not convex, for example in the case of multi-objective integer problems, X_E includes both *supported* and *nonsupported efficient solutions.* This characterization of X_E applies to the problem studied in this thesis therefore, these terms are introduced below.

Definition 3. *$x \in X_E$ is called a supported efficient solution if there exists some $\lambda \in \Re^p_>$ where $\Re^p_> = \{\lambda \in \Re^p : \lambda > 0\}$ such that x is an optimal solution of $max_{x \in X} \lambda^T f(x)$ and $z = f(x)$ is called a supported non-dominated point.*

X_{sE} and Z_{sE} denote the set of all supported efficient solutions and the set of all supported non-dominated points, respectively. On the other hand, the set of all non-supported efficient solutions and the set of all non-supported non-dominated points are denoted by X_{nE} and Z_{nE}.
Lastly, concepts of ideal and nadir points will be introduced. These concepts have significant importance and are utilized a lot in the literature.

Definition 4. *The ideal point z^I consists of the individual maximum values of p objective functions, $z^I := (z^I_1, ..., z^I_p)^T$ where $z^I_i := max\{f_i(x) : x \in X\}$, $i = 1, ..., p$.*

The ideal point is usually not feasible to MOP. The ideal point can be calculated easily by just taking the individual maximum values of objective functions over the feasible set, in other words, by solving p single objective optimization problems. It usually serves as a reference point for the MOP.

Definition 5. *The nadir point is defined as $z^N := (z^N_1, ..., z^N_p)^T$ where $z^N_i := min\{f_i(x) : x \in X_E\}$, $i = 1, ..., p$.*

Different than the calculation of the ideal point, the nadir point calculation is not trivial for $p > 2$, therefore it is often estimated. z^I and z^N have importance in the theory of MOO since they constitute bounds for all $z \in Z_E$ such that $z^N \leq z \leq z^I$. This property is often used in designing algorithms for MOP's.

2.1.2 Exact Solution Methods for Multi-objective Problems

Solution methods for multi-objective problems can be separated in three groups as *a priori*, *interactive* and *a posteriori methods*. *A priori methods* assume preference information so the value function $U(f(x))$ is known to the decision maker (DM). Hence, that single value function can be optimized. However, usually the value function is not known a priori or is hard to find. In *interactive methods*, efficient solutions of MOP are presented to DM and according to DM's preferences the optimization process continues or stops when an acceptable solution is found. The drawback of interactive methods is that DM is not always easily reachable and a large number of efficient solutions may not be inferable to the DM. *A posteriori methods* aim to find the efficient frontier and present it to the DM after the solution process. By this way, DM is able to observe all solution alternatives and choose the best one among them. In this thesis, a posteriori methods will be considered. For more information on a priori and interactive methods the reader can refer to Part II, Chapter 4 and 5 of [76].

In addition, solution techniques can be considered according to the solution quality as exact methods or approximation methods and heuristics. Exact methods are elaborated on this thesis since the aim is to find the non-dominated set of the bi-objective quadratic knapsack problem.

It can be claimed that the most important exact methods in terms of applicability and efficiency are the weighted sum, the ϵ-constraint and the Tchebycheff methods. These three methods and another exact solution method that combines the first two methods are explained below.

Weighted Sum Method

In the weighted sum method, the idea is to multiply each objective function with weights and then optimize the weighted sum of p objectives. In this way, the multi-objective optimization problem is converted into a single objective optimization problem. A broad discussion of the method, also the theorems and proofs given below,

can be found in [29]. For more on discussion of the method refer to [76].
Before starting to give information on the method, notations $\Re^p_{\geq}$ and $\Re^p_{>}$ will be used in this section should be defined as $\Re^p_{\geq} = \{y \in \Re^p : y \geq 0\}$ and $\Re^p_{>} = \{y \in \Re^p : y > 0\}$.
The weighted sum problem $P(\lambda)$ can be defined as,

$$\begin{aligned} &\mathit{maximize} \quad \sum_{i=1}^{p} \lambda_i f_i(x) \\ &\mathit{subject\ to} \quad x \in X, \end{aligned} \tag{2.2}$$

where $\lambda_i \in \Re_{\geq}$ for all $i = 1, ..., p$. It is also usually assumed that $\sum_{i=1}^{p} \lambda_i = 1$.
Several theoretical results are given below on finding efficient solutions via weighted sum method.

Theorem 2.1.1. *Let $\hat{x}$ be an optimal solution of the weighted sum problem $P(\lambda)$ where $\lambda \in \Re^p_{\geq}$. Then $\hat{x}$ is weakly efficient.*

Theorem 2.1.2. *Let $\hat{x}$ be an optimal solution of the weighted sum problem $P(\lambda)$ where $\lambda \in \Re^p_{>}$. Then $\hat{x}$ is an efficient solution.*

Theorem 2.1.3. *Let $\hat{x}$ be the unique optimal solution of the weighted sum problem $P(\lambda)$ where $\lambda \in \Re^p_{\geq}$. Then $\hat{x}$ is an efficient solution.*

These theoretical results are not adequate for the characterization of X_E. The next theorem will complete the characterization with a convexity assumption which is a weakness of the weighted sum method. A convex MOP refers to a problem in which all objective functions are convex and the feasible set is convex.

Theorem 2.1.4. *Let the multi-objective optimization problem be convex. If $\hat{x}$ is efficient, then there exists a weighting vector λ where $\lambda_i \in \Re_{\geq}$ for all $i = 1, ..., p$, and $\sum_{i=1}^{p} \lambda_i = 1$, such that $\hat{x}$ is an optimal solution of the weighted sum problem $P(\lambda)$.*

The weighted sum method offers a way to find efficient solutions of MOP easily without changing the structure of the original problem. However, the previous theorem does not cover integer or other non-convex MOP's. In fact, in such cases it

has been shown that the weighted sum method cannot find the entire efficient set X_E since the non-supported solutions which exist in the non-convex regions of the problem are missed.
In order to overcome this weakness in bi-objective problems, two-phase methods can be applied after solving the weighted sum problem $P(\lambda)$ [107]. Two-phases methods find the set of supported solutions by solving a parameterized single objective problem in the first phase. After that, the non-supported solutions are determined with, for example branch-and-bound approaches.

ϵ-Constraint Method

The ϵ-constraint method is a well-known solution procedure for multi-objective optimization problems along with the weighted sum approach. It was first introduced by Haimes *et al.* [113]. In the ϵ-constraint problem, one of the objective functions is selected to be maximized, and the other objectives are taken as inequality constraints. The k^{th} objective ϵ-constraint problem $P_k(\epsilon)$ is,

$$\begin{aligned} &maximize && f_k(x) \\ &subject\ to && f_j(x) \geq \epsilon_j,\ j = 1, ..., p, \quad j \neq k, \\ & && x \in X \end{aligned} \tag{2.3}$$

where $\epsilon_j \in \Re,\ j = 1, ..., p,\ j \neq k$. Since k is arbitrary, we define $\epsilon = (\epsilon_1, \ldots, \epsilon_p)^T \in \Re^p$ as a vector of possible righthand side values.
The following theorems stated here without proofs introduce the relation between efficiency and $P_k(\epsilon)$. A comprehensive discussion of the method and proofs of theorems are given in [105].

Theorem 2.1.5. *Let $\hat{x}$ be an optimal solution of $P_k(\epsilon)$ for some k. Then $\hat{x}$ is weakly efficient.*

Theorem 2.1.6. *Let $\hat{x}$ be a unique optimal solution of $P_k(\epsilon)$ for some k. Then $\hat{x}$ is efficient.*

Theorem 2.1.7. *$\hat{x} \in X$ efficient if and only if there exists $\epsilon \in \Re^p$ such that $\hat{x}$ is an optimal solution to $P_k(\epsilon)$ for all k=1,...,p.*

The strength of the method can be observed from above theorems where no convexity or linearity assumptions are needed for the efficiency of the solution. Unlike the weighted sum method, the entire efficient set including the non-supported solutions can be generated with the ϵ-constraint method by changing ϵ values.
Yet, this method has two problems that must be fixed before implementing. The first one is the difficulty of parameterizing ϵ values and the second is the need to eliminate the weakly efficient solutions found.
Parameterizing all ϵ values so that no efficient solution is missed, can be done effectively for bi-objective problems. However, for more than two objectives it is somewhat challenging and computationally expensive. For instance, a scheme for parameterizing ϵ values in the multi-objective space is given in [66]. Although this algorithm works for three and more objectives, the computational effort required increases dramatically when p is large. A more recent study that utilizes the ϵ-constraint method is given in [55].
Weakly efficient solutions are also found as optimal solutions of $P_k(\epsilon)$ and these solutions need to be eliminated when generating the efficient set. Lexicographic optimization, augmented ϵ-constraint method or two-stage approaches can be used to eliminate these solutions. Lexicographic optimization will be discussed later. The details of the augmented ϵ-constraint method is given in [73].

Other Exact Solution Methods

Two significant solution methods other than the weighted sum and the ϵ-constraint method are explained in this subsection. These are the hybrid and the generalized Tchebycheff Norm method.
The *hybrid method* combines the weighted sum and the ϵ-constraint method by optimizing the weighted sum of all objectives subject to all objectives taken as constraints. Let $\hat{x}$ be an arbitrary feasible point of MOP. Then the *hybrid problem* (HP) can be

stated as,

$$\begin{aligned} &maximize && \sum_{i=1}^{p} \lambda_i f_i(x) \\ &subject\ to && f_i(x) \geq f_i(\hat{x}),\ i = 1, ..., p, \\ & && x \in X \end{aligned} \tag{2.4}$$

where $\lambda \in \Re^p_{\geq}$. All efficient solutions can be obtained with the hybrid method. This result is stated in the following theorem and the proof is given in [29].

Theorem 2.1.8. *Let $\lambda \in \Re^p_{>}$. A feasible solution $\hat{x} \in X$ is an optimal solution of the problem HP if and only if $\hat{x} \in X_E$.*

Finding efficient solutions with this method is computationally more expensive due to the higher number of constraints and the more complex objective function compared to the weighted sum and ϵ-constraint method, which is a disadvantage over these methods.
Lastly, the method for generating the efficient frontier using the generalized Tchebycheff Norm will be described. This method was first introduced by Bowman [9] and it is also referred to as compromise programming [29]. The *generalized Tchebycheff norm* of the p dimensional objective vector $f(x)$ is denoted by $\|f(x)\|$ and defined as $\|f(x)\|_\beta = max_{i=1,...,p}\beta_i|f_i(x)|$ where $\beta \in \Re^p_{>}$. Let z^I be the ideal point as defined before. Then (P_β) can be defined as follows,

$$(P_\beta) \qquad min_{x \in X} \|f(x) - z^I\|_\beta \tag{2.5}$$

According to Bowman [9] all efficient solutions of MOP can be generated by solving (P_β) parameterizing on β.

Theorem 2.1.9. *$\hat{x}$ is efficient only if it is a solution to (P_β) for some $\beta = \hat{\beta}$.*

However, this result is not a complete characterization of efficient solutions since inefficient solutions may be found with this method. A complete characterization is given in [9] under a slightly restrictive assumption of *uniform dominance*.

Definition 6. *The efficient set is uniformly dominant if for every inefficient solution $\hat{x}$ there exits an efficient solution x such that $f_i(\hat{x}) < f_i(x)$ for all i.*

Hence, with the uniform dominance assumption it is accepted that weakly efficient solutions do not exist. The below theorem completes the characterization for uniformly dominant efficient sets.

Theorem 2.1.10. *If the efficient set is uniformly dominant then all solutions to (P_β) are efficient solutions.*

In the absence of uniform dominance assumption, to avoid generation of inefficient solutions the *weighted augmented Tchebycheff program* defined below which yields only the efficient solutions can be applied [96].

$$min_{x\in X}(max_{i=1,\dots,p}\beta_i|f_i(x) - z_i^I| + \rho\sum_{i=1}^{p} f_i(x)), \tag{2.6}$$

where ρ is a sufficiently small positive number.
Another technique for eliminating inefficient solutions is using a *two−phase* approach. In this type of approaches, the generalized Tchebycheff norm is used in the first stage then weakly efficient solutions found in the first stage are eliminated with the second stage optimization problem [90].
For further reading on the theory of MOO, other exact solution methods and an extensive analysis of these methods refer to [76], [29] and [105].

2.1.3 Approximation and Heuristic Methods

Determining the efficient set of a MOP exactly is not an easy task. The number of efficient solutions is quite large; therefore, solving MOP's requires a considerably large computational effort. Thus, approximations, representations and heuristics are studied extensively in MOO literature. Some examples of these methods from the literature that are proposed for different types of problems will be given in this section.

First, the difference between an approximation and a representation should be clarified. Representing an efficient set means finding a subset of the exact efficient set of the problem. For instance, the set of supported efficient solutions of a non-convex MOP constitute a representation of that problem. On the other hand, solutions in an approximation do not have to belong to the exact efficient set; they are estimations of the exact efficient solutions.

Most of the proposed methods deal with linear MOP's. An algorithm for linear bi-objective programs is proposed by Solanki and Cohon in [94]. This algorithm generates a piecewise linear approximation by connecting efficient points found with the weighted problem. The procedure provides an approximate representation of the efficient set. Solanki et al. then introduced the generalization of this method to the multi-objective case [93]. Concepts of multi-objective optimization are well utilized in both methods; however, these methods are not very practical in terms of computation.

A method is proposed for multi-objective combinatorial problems in [58] where the non-dominated frontier is approximated by fitting a hypersurface that passes through the extreme points in the objective space. This method can be implemented easily and gives a good approximation of the efficient set.

The representation method proposed by Sylva and Crema [99] finds well-dispersed subsets of non-dominated solutions for linear, integer and even mixed integer MOP's. The procedure generates the non-dominated solution that maximizes the infinity norm distance to the set dominated by all the previously found solutions. This requires adding binary variables and constraints to the problem at each iteration, making the method inefficient even for moderate size problems.

Martin et al. [70] proposed an approximation method for continuous MOP's which incorporates various concepts from evolutionary algorithms (EA) and stochastic optimization. The common underlying idea behind EA's is to form the next generation of solutions using genetics mechanisms such as mutation, recombination and selection starting with an initial population of solutions. The method in [70] starts with a random sample of the feasible set similar to EA's. Then the non-dominated frontier is approximated with a discrete approximation and a curve is fitted to this approximation using regression. Non-connected non-dominated frontiers of nonlinear problems can also be approximated with this method. Application to well known problems or examples of problems with more than two objectives are not included in the paper; therefore, the quality of the approximation method cannot be stated exactly.

For a survey on approximation methods the reader can refer to [87].

Metaheuristics, which are general purpose heuristic approaches, have been used actively in the last decade for MOO. EA's have been particularly popular. The common property of EA's is their ability to generate multiple solutions in a single run of the algorithm.

Strength-Pareto EA (SPEA) and Non-dominated Sorting Genetic Algorithm (NSGA) introduced in [115] and [95] respectively are among the classical evolutionary multi-objective optimization (EMO) algorithms. SPEA basically combines previous EA's characteristics. In addition, it keeps an external non-dominated set which provides transmission of non-dominated solutions to the next generations. SPEA2 introduced in [27] improved the earlier version of SPEA by fixing some of its weaknesses. The main characteristic of NSGA is the calculation of non-domination levels. NSGAII proposed in [24] over NSGA, is elitist, less expensive in computation and also does not need a user defined sharing parameter. According to the computational results given in [27] on problems of different characteristics, SPEA2 and NSGA2 have similar performances.

EMO algorithms can also be used to address dynamic MOP's as described in [32]. Some new and improved methods proposed in this field are given in [116] and [67]. Recent developments and a review of EMO algorithms can be found in [20], [47], and [21].

2.1.4 Applications of MOO

In this subsection, we will be giving some examples of MOO applications which are spread over many diverse areas. Obviously, we cannot name all of them here; however, this subsection will give an insight to better understand the importance of MOP's.

MOO has applications in finance, energy planning, telecommunication network planning and design, and sustainable development, each explained in [49]. For a bibliography on applications of multi-objective decision making check out [110] and for applications in finance see [97]. Some more examples of MOO applications are on non-financial performance evaluation, optimal capital structure problem, marketing and e-commerce [12].

Bhaskar et al. [104] reviewed MOO applications in chemical engineering and Handl et al. [39] reviewed MOO applications in bio-informatics and computational biology. MOO is also applied to the traffic assignment problem by considering total travel time, air pollution and travel distance as objectives in [102]. Weber and Current looked at the vendor selection problem with a multi-objective approach in [109]. The capital budgeting problem is considered as a MOO problem in [6]. The road network design problem featured in [14] is another application area of MOO. Moreover, MOO is applied to motion planning, trajectory planning and navigation of robot systems in [71], [79] and [52] respectively. Multi-objective system reliability design problem is examined in [100]. Improving energy efficiency in buildings [26], test data generation [64] and environmental investment decision making [42] can be counted as examples of MOO applications to different and diverse areas.

The following applications belong to the family of bi-objective problems. Ehrgott and Ryan [30] modeled robust crew schedules for airlines with bi-objective optimization by minimizing cost and maximizing robustness as objectives which are in conflicting natures. Sayın and Karabatı [89] formulated two-machine flow shop scheduling problem as a bi-objective problem by trying to minimize make-span and sum of completion times simultaneously. The bi-objective resource-constrained project scheduling problem is solved by using a tabu search algorithm in [2] with objectives of make-span minimization and robustness maximization.

EMO algorithms are used in many applications of MOP's which will be exemplified here. Farmani et al. [33] applied EMO techniques to the water distribution network design problem. EMO as well as MOO strategies are applied to intensity modulated radiation therapy (IMRT) dose optimization problem and IMRT planning in [63] and [62] respectively. A survey on the applications of EMO algorithms to economics and finance problems is given in [16]. Meunier et al. [75] proposed a multi-objective genetic algorithm for radio network design problem and Islier [48] used a genetic algorithm approach to the multi-objective facility layout design. Deb discussed EMO algorithms for engineering design and cited applications in [23].

Applications of MOO related to the knapsack problems will be given separately in the next section.

2.2 The Knapsack Problem

The *knapsack problem* (KP) is one of the most studied combinatorial optimization problems in the literature. Despite its *NP-complete* [68] structure, the KP caught the attention of many researchers since it is used in many real life applications such as capital budgeting, cargo loading, and cutting stock [88]. Besides these direct applications, *knapsack problem* is also encountered as a subproblem of numerous complex problems. A comprehensive discussion of various types of knapsack problems, solution methods for these problems and applications can be found in the books by Martello and Toth [88], and Kellerer et al. [38].

In the original version of the *knapsack problem* (KP) also known as the *0-1 (zero one)* or *binary knapsack problem*, the input consists of a knapsack with a certain capacity b and a set of items $j = 1, ..., n$, each of which has a weight and profit denoted by w_j and p_j respectively $(j = 1, ..., n)$. The problem can be stated as follows,

$$\begin{aligned} &maximize \ \sum_{j=1}^{n} p_j x_j \\ &subject\ to \ \sum_{j=1}^{n} w_j x_j \leq b, \\ &x_j \in \{0, 1\}, \qquad j = 1, ..., n. \end{aligned} \tag{2.7}$$

It is usually assumed, without loss of generality, that p_j, w_j and b are positive integers, $\sum_{j=1}^{n} w_j > b$, $w_j \leq b$ for $j = 1, ..., n$. The binary variable $x_j = 1$ means selecting and $x_j = 0$ means excluding the j^{th} item. Moreover, in some cases x_j are taken to be positive integers and we will refer to that problem as the *integer knapsack problem.*
An insight to the mathematical formulation of the problem can be given as the following. Suppose an investment of b dollars will be made and n different investment options are available. If we consider p_j as the expected profit associated with investment option j and w_j as the amount of dollars required to make j^{th} investment, then optimal solution of this problem can be found by solving the corresponding KP.
The *knapsack problem* has a lot of variations each called by special names. Some of these versions will be explained in here. The *multidimensional knapsack problem* (MdKP) is a generalization of the *knapsack problem* which has m constraints instead of one capacity constraint and is known to be *NP-hard.* In the *multiple knapsack problem* a subset of n items are placed into m knapsacks that have different capacities. In another version called the *multiple-choice knapsack problem*, items are divided into classes and only one item has to be selected from each class.
Recent studies on different versions of the *knapsack problem* will be summarized and cited here for the completeness of this thesis.
Various solution techniques are used to solve KP in the literature. Three exact algorithms for solving KP; the branch-and-bound, core and dynamic programming algorithms, are surveyed in [69]. Large sized uncorrelated instances up to $10,000$ variables, where the weights w_j and the profits p_j are uniformly randomly distributed in the

range $[1, R]$, are solved in this paper. These exact algorithms will be briefly explained in the following. The branch-and-bound procedure is based on partitioning the space of all feasible solutions and calculating upper bounds for the solutions in that subset. The idea is to eliminate subset A from the branching tree if the upper bound of A is less than the lower bound of subset B. The core algorithm determines a subset of items as the *core* of the problem by their efficiencies and further explore solutions in the core problem. Lastly, dynamic programming method solves problems by reducing complex problems down into simple subproblems like the branch-and-bound method. KP is treated as a sequential decision making process whose states and transitions between them are determined by the feasibility constraint.

A more recent study proposes a novel global harmony search algorithm, an evolutionary algorithm, and reports its performance on large scale instances up to $1,500$ variables in [117]. Also, the problem of re-optimizing KP is examined by Archetti et al. in [4].

An exact algorithm for MdKP is given in [8] which can solve instances with 10 constraints and 500 variables to optimality. Heuristics are used to solve MdKP commonly since it is *NP-hard* [60], [82], and [34].

Heuristic methods for the *multiple-choice multidimensional knapsack problem* (McdKP) are proposed in [18] and [1] by column generation and by constructing convex hulls, respectively. An exact and a heuristic method is given in [41].

Several examples of *knapsack problem* applications chosen from literature will be stated in here. In addition to those stated earlier, the KP has been used in the area of cryptography [74]. The second stage of the production planning problem faced by a small foundry is modeled with a bounded KP in [13]. The *partially ordered knapsack problem* and its applications to scheduling is discussed in [59]. Mansini and Speranza [84] formulated the problem of selecting assets in a securitization as a MdKP. A variant of McdKP is used to formulate the utility model for resource allocation on computational grids in [106].

The *multi-objective knapsack problem* (MOKP) is an extension of the KP and has applications in transportation planning, packaging and loading, conservation biology, capital budgeting and financial management [19]. The MOKP can be formulated as

$$\begin{aligned} & \textit{maximize} \sum_{j=1}^{n} p_j^i x_j \qquad i = 1, ..., p \\ & \textit{subject to} \sum_{j=1}^{n} w_j x_j \leq b, \\ & \qquad x_j \in \{0,1\}, \qquad j = 1, ..., n. \end{aligned} \tag{2.8}$$

p_j^i are nonnegative integers in this case. For the special case where $i = 2$ this problem is called the *bi-objective knapsack problem* (BOKP) also referred as the *linear bi-objective knapsack problem.*

Solution techniques used to solve BOKP and MOKP, and applications of these problems to real life scenarios are given in the following paragraphs.

The core concept stated above is extended to solve BOKP exactly and approximately in [22] and computational results on different types of instances are also included. For example, uncorrelated instances with 500 variables are reported in the paper. Eben-Chaime [28] presented an algorithm for the construction of a parametric solution for BOKP and solved some small sized instances. Visée et al. [107] introduced two-phases and branch-and-bound procedures to solve BOKP and computational results are presented up to 500 variables. Zhang and Ong [114] proposed a heuristic method for BOKP and included computational results for very large instances up to 50, 000 variables. Kozanidis [61] worked on the multiple-choice bi-objective knapsack problem. The bi-objective max-min knapsack problem is studied in [101] and a report of solutions for large instances with 16, 000 items is given in the paper. BOKP is solved with a tabu search based procedure in [37]. A new solution algorithm for BOKP is presented in [25] based on the use of bound sets which requires less memory than some other dynamic programming approaches.

Some examples of BOKP applications should also be included here. Jenkins [50] modeled the problem of remediation of contaminated lightstation sites with BOKP.

A well-known application of BOKP to the capital budgeting problem, with objectives of maximizing the present value of the accepted projects and minimizing their risk, is examined in [86].

Examples of studies on the MOKP, most of them considering the three objective case, will be stated here. An approximation method is given for the multidimensional MOKP in [31], however no computational testing is included. The same problem is addressed also in [35] with an exact method, heuristics and computational testing. Bazgan et al. [5] solved MOKP with a dynamic programming approach and reported numerical results for the three objective case. A dynamic programming based approach is also used to model integer MOKP in [57].

Lastly, methods belong to the heuristics family are used a lot in the literature to solve different versions of knapsack problems including BOKP's and MOKP's. Wilbaut et al. presented a survey on heuristics applied to a variety of *knapsack problems* [111].

2.2.1 The Quadratic Knapsack Problem

The *quadratic knapsack problem* (QKP), which belongs to the class of nonlinear knapsack problems, is first introduced by Gallo et al. in [36] as maximizing a quadratic objective function subject to a linear knapsack capacity constraint.
QKP is a generalization of the *knapsack problem* when $p_{ij} = 0$ for all $i \neq j$, and it is also a constrained version of the *quadratic* $0-1$ *programming problem.*
As in the linear case, we have n items to pick from, each of which has a nonnegative integer profit and a positive integer weight. The profit matrix $P = \{p_{ij}\}$ is a nonnegative symmetric square matrix of order n. It can be assumed without any loss of generality that profit matrix is symmetric, i.e. $p_{ij} = p_{ji}$ for all $i, j,\ j > i$. This assumption can be made since P can be converted to the symmetric form easily by using $\frac{(P+P^T)}{2}$. The capacity b is a positive integer.

QKP is NP-hard in the strong sense [80] and formulated as follows:

$$\begin{aligned}
&\textit{maximize} \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}x_ix_j \\
&\textit{subject to} \sum_{j=1}^{n} w_jx_j \leq b, \\
&\qquad x_j \in \{0,1\}, \qquad j = 1, ..., n.
\end{aligned} \tag{2.9}$$

QKP is sometimes formulated with a linear part in the quadratic objective function, where c_i is a positive integer, as shown below:

$$\begin{aligned}
&\textit{maximize} \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}x_ix_j + \sum_{i=1}^{n} c_ix_i \\
&\textit{subject to} \sum_{j=1}^{n} w_jx_j \leq b, \\
&\qquad x_j \in \{0,1\}, \qquad j = 1, ..., n.
\end{aligned} \tag{2.10}$$

In our case, the linear terms in the objective of the formulation can be implicitly taken into account in the quadratic part by adding these coefficients to the corresponding diagonal elements of the quadratic matrix P since $x_i = x_i^2$ for $x_i = \{0, 1\}$.

In QKP, different than the KP, an item has its own profit and an additional profit when selected with another item, where p_{jj} is the profit achieved if item j is selected, and for $j > i$, $p_{ij} + p_{ji}$ is the profit achieved if both items i and j are selected. This characterization of QKP can be interpreted as the interdependence of items with each other, which has realizations in applications.

The version we are dealing with is called the supermodular case where all coefficients are integer and nonnegative. Moreover, our objective function is nonseparable. In general, problems with a nonseparable objective function are much harder to solve than those with a separable one. Karush-Kuhn-Tucker conditions get complicated for the nonseparable problem since the objective function of QKP has elements in the form x_ix_j where $i \neq j$.

Many versions of QKP and the nonlinear knapsack problem have been addressed in the literature. These versions include continuous, integer or binary variables, nonseparable or separable functions, convex or non-convex functions and additional constraints such as bounds on the variables [10].

2.2.2 Solution Techniques and Applications of the QKP

Different solution techniques and applications of QKP will be discussed in this subsection.

A lot of work has been done on QKP since the 1980's. QKP has applications in many areas, and hence it is important to develop exact and heuristic algorithms for solving QKP. Techniques of relaxation, linearization, reformulation, Lagrangian relaxation, Lagrangian decomposition, and semi-definite programming are used to compute bounds for the problem. Heuristics, reduction techniques, branch-and-bound algorithms and approximations are also used to solve QKP.

An exact branch-and-bound algorithm for QKP, where upper bounds are computed by considering a Lagrangian relaxation, is introduced by Caprara et al. in [15]. The proposed method can solve QKP's with up to 400 binary variables and this is realized through tight upper bounds found which are typically within 1% of the optimum according to [15].

A recent paper by Wang et al. [108] compares three different linearizations of QKP from the literature with the mixed integer quadratic programming (MIQP) solver of CPLEX and reports their performances up to 800 variables and 75% percentage fullness. In this paper, the quadratic matrix is transformed to a positive semi-definite matrix to ensure that the required convexity conditions were satisfied before starting the optimization process with CPLEX's MIQP solver. Moreover, x_i^2's are directly taken as x_i's in this paper since $x_i = \{0, 1\}$. It has been shown that the quadratic model outperforms linearized models for large instances.

Billionnet and Sourour also applied different linearization techniques to QKP and reported some results in [7]. The quadratic matrix is again transformed to a positive semi-definite matrix with two different preprocessing methods and (MIQP) solver of CPLEX is used in this paper. The *quadratic 0-1 programming problem* is solved with up to 200 variables. With the first preprocessing technique, no instance which has larger than 100 variables could be solved to optimality in three hours cpu time and only 4 instances out of 10 with 100 variables could be solved within the time limit. With the second preprocessing technique which uses semi-definite programming, only one instance out of 10 with 150 variables and 6 instances out of 10 with 120 variables could be solved within three hours cpu time.

Quadri et al. [83] developed a branch-and-bound algorithm coded in C language to solve large scale separable quadratic multi-knapsack problem up to 2000 variables. The upper bound and preprocessing techniques incorporated allow large problems to be solved. Again a branch-and-bound algorithm for the separable integer quadratic knapsack problem with bounds on variables is generated in [11] and results of implementations with up to 100 variables are reported.

Helmberg et al. [40] used a semi-definite programming approach to solve QKP and reported some results for up to 60 variables. Apart from being theoretically interesting, it is concluded that semi-definite approach is probably too expensive to solve pure quadratic knapsack problems to optimality.

Pardalos et al. presented three different algorithms to solve quadratic problems with a knapsack constraint and continuous variables in [78]. Kiwiel [56] solved large scale continuous quadratic knapsack problems where the quadratic matrix only has positive diagonal elements making the objective function strictly convex. Breakpoint searching algorithms are proposed for this separable problem. The continuity, convexity and separability characteristics of the problem made it possible to solve large scale instances.

Hua et al. presented different heuristics for integer QKP with real objective function coefficients including approximate dynamic programming approaches in [44]. Narayan and Patvardhan [77] proposed a quantum evolutionary algorithm (QEA) for solving QKP. Sun et al. proposed an algorithm for the minimization of separable concave knapsack problems with bounded integer variables which is able to solve large-scale problems up to 1200 variables [98].

We will discuss some applications of QKP in the following paragraphs.

The problem of selecting a number of sites for satellite stations such that the global traffic between these stations is maximized subject to a budget constraint in telecommunications is a QKP [112]. Similar models arise when considering the location of airports, railway stations or freight handling terminals [85].

Another application of QKP is the capital budgeting problem where the decision maker is confronted with a set of investments from which he must select a portfolio [65]. Here, the problem includes restrictions requiring that each project be accepted or rejected in its entirety. The decision maker's expected utility function includes as arguments the mean portfolio return and, as a measure of risk, portfolio variance.

The multi-commodity network model is formulated with QKP with bounds on variables [92]. Multi-commodity network flow problems arise when several items share arcs in a capacitated network.

Kellerer and Strusevich offered a fully polynomial approximation scheme for symmetric QKP defined in the presented papers and stated its scheduling applications [54], [53]. Scheduling problems addressed in these papers have the following characteristics. Given a set $N = \{1, 2, ..., n\}$ of jobs to be proceeded without preemption on a single machine; the processing of job $j \in N$ takes p_j time units and positive weight w_j is associated with job j, which indicates its relative importance. It is required to minimize a function $Z(S)$ that depends on the completion times $C_j(S)$.

Also, several graph-theoretic interpretations can be given to QKP and it can be seen that this problem is a generalization of the *clique problem*, which checks whether for a given integer k, a given undirected graph $G = (V,\ E)$ contains a complete subgraph on k nodes. The most famous optimization version of clique is *Max Clique*, and can be solved through a QKP algorithm [15]. Pseudo-polynomial time algorithms for QKP where the underlying graph structure is edge series-parallel is given in [51].

Iasemidis et al. [45], determined predictability of epileptic seizures by using a *quadratic* $0-$

1 *programming* model equivalent to the $k - clique\ problem$. The possibility of prediction of epileptic seizures well in advance of their occurrence is shown in this paper by applying the proposed procedure to epilepsy research for the first time.

Bretthauer and Shetty [10] offered a review of the literature on nonlinear knapsack problems and for a survey on QKP check [80].

Chapter 3

PROBLEM FORMULATION AND SOLUTION TECHNIQUES

The bi-objective quadratic knapsack problem (BQKP) studied in this thesis is stated in this chapter. The lexicographic ϵ-constraint method used to solve the problem and the linearization method incorporated in this formulation are explained. Also, a special weighted sum algorithm which is used to determine the supported efficient solutions of the problem is discussed.

3.1 The Bi-objective Quadratic Knapsack Problem (BQKP)

BQKP can be considered both with and without linear parts in two objectives. We will give the formulation without linear parts here however, the reader should know that the linear parts can be added to the diagonal elements of quadratic matrices and the problem can be handled without considering the linear parts, since as discussed before, $x_i = x_i^2$ for $x_i = \{0, 1\}$.

The formulation of BQKP without the linear parts in objectives is as follows;

$$\begin{aligned} &\mathit{maximize}\ f_1(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}x_ix_j \\ &\mathit{maximize}\ f_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j \\ &\mathit{subject\ to}\ \sum_{j=1}^{n} w_jx_j \leq b, \\ &\qquad x_j \in \{0, 1\}, \qquad j = 1, ..., n. \end{aligned} \tag{3.1}$$

All coefficients are nonnegative integers in these formulations. The profit matrices $P = \{p_{ij}\}$ and $Q = \{q_{ij}\}$ are nonnegative symmetric square matrices of order n. The aim is to determine which k items where $k \leq n$ to choose from these n items that will give non-dominated solutions of the problem.

The properties of the profit matrices stated above are not sufficient for the convexity of the problem. Here, we want to maximize quadratic objectives so we need concave objective functions and therefore negative semi-definite (NSD) quadratic matrices. A real square symmetric matrix is negative semi-definite if and only if all of its eigenvalues are non-positive. This NSD requirement can always be satisfied for the class of problems considered here by modifying P using standard diagonal perturbation techniques [108].

As an example of these techniques, the minimum eigenvalue transformation described in [7] for generating PSD quadratic matrices can be used with a little change in the formulation for the purpose of generating NSD matrices by making all eigenvalues non-positive, which will be explained here.

The perturbed function $q_u(x)$ is defined in the following way

$$q_u(x) = x^T(Q - diag(u))x + u^T x \tag{3.2}$$

where $diag(u)$ is the diagonal matrix obtained from vector u and $u \in \Re$. It can be seen that $q_u(x)$ can also be written as $q(x) + \sum_{i=1}^{n} u_i(x_i - x_i^2)$, and that $q_u(x) = q(x)$ for all $x \in \{0,1\}^n$. The expression $q_u(x)$ is equivalent to our quadratic objective function.

Since, concave objective functions and therefore NSD quadratic matrices are needed for our problem λ_{min} in the formulation given in the referenced paper should be taken as λ_{max}.

Let $\lambda_{max} \in \Re$ be the largest eigenvalue of matrix Q. Here, if at least one term of Q is nonzero then λ_{max} is a real positive number. If we consider the perturbed function

q_u where $u = \lambda_{max}e$ and e is the vector of all ones then we get

$$q_{\lambda_{max}e}(x) = x^T(Q - diag(\lambda_{max}e))x + (\lambda_{max}e)^T x \tag{3.3}$$

Matrix $(Q-diag(\lambda_{max}e))$ is negative semi-definite and then function $q_{\lambda_{max}e}$ is concave. CPLEX itself also transforms the quadratic matrices to NSD matrices before starting the optimization process. After some experimentation, we did not interfere with this step and let CPLEX deal with the NSD requirement on the quadratic matrices.
We have discussed before the *NP* completeness of KP and stated that QKP is *NP* hard in the strong sense. Therefore, it can be concluded that BQKP is a difficult problem.
There are not any particular algorithms intended to solve BQKP in the literature, to the best of our knowledge. We have not encountered any empirical studies that investigate the non-dominated solutions of BQKP. Therefore, understanding the performance of a commercial solver such as CPLEX on BQKP and reporting the related computational results are important which we have addressed in this thesis.

3.2 The Lexicographic ϵ-Constraint Method for Bi-objective Optimization Problems

In this section, the lexicographic variant of the ϵ-constraint method, which is used to generate the non-dominated set of the BQKP, will be described and the formulation of the method for bi-objective optimization problems will be given.
We have discussed the ϵ-constraint method in the second chapter. The lexicographic ϵ-constraint method yields efficient solutions as opposed to weakly efficient ones in the ϵ-constraint approach. Moreover, any efficient solution can be obtained with the lexicographic ϵ-constraint method with a proper choice of ϵ [43]. The theorem given below states this fact and gives the lexicographic ϵ-constraint formulation of a bi-objective optimization problem.

Theorem 3.2.1. *A feasible solution $x^* \in X$ is efficient if and only if it is a solution* *of the problems*

$$\begin{aligned} z_1^* = \mathit{maximize}\ & f_1(x) \\ \mathit{subject\ to}\ & f_2(x) \geq \epsilon_1, \\ & x \in X \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} \mathit{maximize}\ & f_2(x) \\ \mathit{subject\ to}\ & f_1(x) \geq \epsilon_2, \\ & x \in X \end{aligned} \tag{3.5}$$

where $\epsilon_2 = z_1^*$.

The first model (3.4) in the above theorem will be referred as (P_1) and the second model (3.5) will be referred as (P_2) in the following sections.
These two models can be interpreted as described in the following. The first objective is maximized on the feasible set of the bi-objective optimization problem with the second objective bounded below as a constraint. Then, the second objective is maximized over the feasible set of the bi-objective problem plus the first objective as a constraint taking the value obtained from the first stage as a right hand side. In the lexicographic ϵ-constraint method solving these two single objective optimization problems will give an efficient solution of the bi-objective problem. Other efficient solutions can be found by changing the right hand side of the objective constraints.
The need to solve two single objective optimization problems to find one efficient solution is a drawback of the method. Moreover, the problem structure is changed when solving BQKP since a quadratic constraint is added now, which makes the problem dramatically harder to solve.
In the following subsection, we will discuss a special weighted sum algorithm which can find all supported non-dominated solutions of a bi-objective problem with an

effective weight search. Maintaining the structural properties of BQKP is the advantage of this method however, the disadvantage is that non-supported solutions could not be found.

3.3 A Special Weighted Sum Algorithm

A special weighted sum algorithm which is designed to find all supported non-dominated solutions of a bi-objective optimization problem from the literature and its application to BQKP will be explained in this section.
Aneja and Nair [3] introduced an algorithm which finds all supported non-dominated solutions of a bi-objective optimization problem. This algorithm uses the weighted sum method and presents an intelligent way to search the whole weight space.
The algorithmic statement of the method proposed is given below. The validity of the algorithm is also stated with a theorem and its proof in the cited paper. The

Algorithm 1 Aneja and Nair's Weighted Sum Algorithm

```
Step 0: Find $z_1^I$ and $z_2^N$. Record these as $(z_1^1, z_2^1)$ and set $k = 1$.
        Similarly find $z_1^N$ and $z_2^I$.
        If $(z_1^1, z_2^1) = (z_1^N, z_2^I)$, stop.
        Otherwise, record $(z_1^N, z_2^I)$ as $(z_1^2, z_2^2)$ and set $k = k + 1$.
        Define sets $L = \{(1, 2)\}$ and $E = \varnothing$, and go to Step 1.
Step 1: Choose an element $(r, s) \in L$ and set $a_1^{(r,s)} =\mid z_2^s - z_2^r \mid$ and $a_2^{(r,s)} =\mid z_1^s - z_1^r \mid$.
        Solve the weighted problem
```

$$\begin{aligned} & \textit{maximize } a_1^{(r,s)} z_1(x) + a_2^{(r,s)} z_2(x) \\ & \textit{subject to } \ x \in X. \end{aligned} \tag{3.6}$$

```
        Let $x^*$ be the optimal solution to the weighted problem.
        Calculate $z_1(x^*)$ and $z_2(x^*)$.
        If $(z_1(x^*), z_2(x^*))$ is equal either to $(z_1^r, z_2^r)$ or $(z_1^s, z_2^s)$ set $E = E \cup \{(r, s)\}$
        and go to Step 2.
        Otherwise record $(z_1^k, z_2^k)$ such that $z_1^k = z_1(x^*)$ and $z_2^k = z_2(x^*)$ and set
        $k = k + 1$, $L = L \cup \{(r, k), (k, s)\}$ and go to Step 2.
Step 2: Set $L = L - \{(r, s)\}$. If $L = \varnothing$, stop.
        Otherwise go to Step 1.
```

points recorded by the algorithm in set E gives all the supported efficient solutions

of the problem.
The fact that the problem structure stays the same is an advantage of this method. However, only the supported non-dominated solutions of BQKP can be found with this algorithm since the problem is not convex. Moreover, the algorithm will not find non-extreme supported efficient solutions unless it is modified to solve for all alternative optimal solutions in Step 1.
This algorithm is applied to BKQP and a representation of the non-dominated set consisting of all the supported non-dominated solutions is found by this way. The details of the application to our problem is given in the next chapter.

3.4 Linearization

Quadratically constrained quadratic (QCQ) subproblems that arose in the lexicographic ϵ-constraint formulation of BQKP increase the degree of difficulty of the problem. Especially, the quadratic constraint added to the model dramatically affects the performance of the CPLEX solver. As a consequence, to solve the problem for higher number of variables a linearization technique introduced in the literature is incorporated in the lexicographic ϵ-constraint formulation.
The linearization technique given in [17] fits the submodels of our lexicographic ϵ-constraint formulation. Linearization is given for the following problem (P) in the cited paper.

$$\begin{aligned} (P) \qquad \textit{minimize } & f(x) = x^T A x, \\ s.t. \ & Bx \geq b, \\ & x^T C x \geq \alpha, \\ & x \in \{0,1\}^n. \end{aligned}$$

Here, α is a constant, A and C are general $n \times n$ matrices, i.e., $a_{ij}, c_{ij} \in \Re$. B is an $m \times n$ matrix, b is a constant vector, m and n are some integer numbers. So, (P) is a binary QCQ model like our submodels. Although the linear parts in quadratic

objectives can be added easily with the method we have explained before, we applied this linearization method to BQKP's without linear parts in objectives.
Let e be a vector of all ones, i.e., $e = (1, \ldots, 1)^T$. The below formulation is introduced in [17] as the linearization of (P);

$$\begin{aligned}
(P') \qquad & \textit{minimize } g(s,x) = e^T s - M e^T x, \\
& s.t.\ Ax - y - s + Me = 0, \\
& \quad Bx \geq b, \\
& \quad y \leq 2M(e - x), \\
& \quad Cx - z + M'e \geq 0, \\
& \quad e^T z - M' e^T x \geq \alpha, \\
& \quad z \leq 2M'x, \\
& \quad x \in \{0,1\}^n, \\
& \quad y_i,\ s_i,\ z_i \geq 0,
\end{aligned}$$

$$\textit{where } M' = max_i \sum_{j=1}^{n} |c_{ij}| \ \textit{ and } M = max_i \sum_{j=1}^{n} | a_{ij} | .$$

The equivalence of the formulations (P) and (P') is stated in the following theorem.

Theorem 3.4.1. *(P) has an optimal solution x^0 if and only if there exists y^0, s^0, z^0 such that (x^0, y^0, s^0, z^0) is an optimal solution of (P').*

The proof of the above theorem is given in [17].
Our QCQ subproblems in the lexicographic ϵ-constraint formulation of BQKP are in the following form;

$$\begin{aligned}
& \textit{maximize } f_1(x) = x^T P x, \\
& s.t.\ w^T x \leq b, \\
& \quad x^T Q x \geq f_1(x^*), \\
& \quad x \in \{0,1\}^n.
\end{aligned} \tag{3.7}$$

So, if we modify our problem in a suitable way we can apply the linearization (P'). The modified version of our QCQ subproblem is given below;

$$\begin{aligned} \textit{minimize } & f_1(x) = x^T(-P)x, \\ \textit{s.t. } & (-w^T)x \geq -b, \\ & x^TQx \geq f_1(x^*), \\ & x \in \{0,1\}^n, \\ & f_1(x^*) \textit{ is a constant.} \end{aligned} \tag{3.8}$$

It can be noticed that this formulation is equivalent to the one defined as (P). Hence, the resulting linearization equivalent to the above problem is as follows;

$$\begin{aligned} \textit{minimize } & g(s,x) = e^Ts - Me^Tx, \\ \textit{s.t. } & (-P)x - y - s + Me = 0, \\ & (-w^T)x \geq -b, \\ & y \leq 2M(e-x), \\ & Qx - z + M'e \geq 0, \\ & e^Tz - M'e^Tx \geq \alpha, \\ & z \leq 2M'x, \\ & x \in \{0,1\}^n, \\ & y_i,\ s_i,\ z_i \geq 0, \end{aligned} \tag{3.9}$$

$$\textit{where } M' = max_i \sum_{j=1}^{n} |q_{ij}| \ \textit{ and } M = max_i \sum_{j=1}^{n} |\, p_{ij} \,| \,.$$

We used the above formulation to linearize our QCQ submodels. With this linearization, a binary QCQ subproblem with n variables and two constraints (one knapsack and one quadratic constraint) is turned into a mixed binary linear problem with $4n$ variables and $4n+2$ constraints.

The insight behind this linearization can be stated as in the following. Newly added

variables and constraints in the linearization take the place of quadratic terms in the QCQ formulation. For example, from the third constraint we see that for every i, where $x_i = 1$, we need to have $y_i = 0$; for every i, where $x_i = 0$, the value of y_i does not depend on this constraint but is upper bounded by the value of $2M$. The opposite of this relationship between x_i's and y_i's exists for x_i's and z_i's. With these additional variables and constraints the minimization of the objective function defined in this linearization provides the optimal solution for the QCQ subproblem defined above.
In conclusion, the lexicographic ϵ-constraint formulation of BQKP is linearized with this approach and we were able to solve problems with higher number of variables using CPLEX.

Chapter 4

GENERATING THE NON-DOMINATED SOLUTION SET OF BQKP

Two different approaches both incorporating the lexicographic ϵ-constraint method in their core, are used to generate the entire non-dominated solution set of BQKP in this thesis. The first approach involves implementation of lexicographic ϵ-constraint method directly to the problem. In the second approach, a linearization technique for QCP's is applied to the lexicographic ϵ-constraint submodels. In addition, supported non-dominated solutions are generated with the special weighted sum algorithm discussed in the previous chapter.
Implementations of these methods will be explained in this chapter.

4.1 Quadratically Constrained Quadratic Model

The lexicographic ϵ-constraint method is coded in CPLEX to generate the non-dominated set of BQKP. The aim here is to determine and to report the performance of CPLEX on solving BQKP when the problem is directly submitted to the commercial solver. Therefore, extant settings on CPLEX are not changed. This code will be referred as the LexECM code.
The LexECM randomly generates BQKP's with user determined number of variables and percentage of fullness of quadratic matrices. The lexicographic ϵ-constraint method is applied in the code as described in the previous chapter. In the code, all efficient solutions are found in a while loop which starts with z_1^I, the ideal z_1 value and $\epsilon_1 = 0$ as the righthand side of the first model and iterates until ideal z_2 is found. At first z_2^I, the ideal z_2 is calculated to be given as the end parameter to the while loop.

The algorithmic statement of LexECM is given below.

Algorithm 2 LexECM

```
$\epsilon_2 \leftarrow 0$
Solve $(P_2)$ with $\epsilon_2$
$z_2^I \leftarrow z_2^*$
$\epsilon_1 \leftarrow 0$
while $\epsilon_1 \leq z_2^I$ do
    Solve $(P_1)$ with $\epsilon_1$
    $\epsilon_2 \leftarrow z_1^*$
    Solve $(P_2)$ with $\epsilon_2$
    $\epsilon_1 \leftarrow z_1^* + 1$
    Keep $(z_1^*, z_2^*)$
end while
```

We will state two models in the while loop. (P_1) is in the form given below.

$$\begin{aligned} z_1^* = \mathit{maximize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}x_ix_j \\ \mathit{subject\ to} \quad & \sum_{j=1}^{n} w_jx_j \leq b, \\ & \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j \geq \epsilon_1, \\ & x_j \in \{0,1\}, \qquad j = 1, ..., n. \end{aligned} \tag{4.1}$$

The second model below takes the optimal value $z_1(x^*)$ of the first model and uses it in the righthand side of the quadratic constraint. This model is named (P_2).

$$\begin{aligned} \mathit{maximize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_ix_j \\ \mathit{subject\ to} \quad & \sum_{j=1}^{n} w_jx_j \leq b, \\ & \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}x_ix_j \geq \epsilon_2, \\ & x_j \in \{0,1\}, \qquad j = 1, ..., n. \end{aligned} \tag{4.2}$$

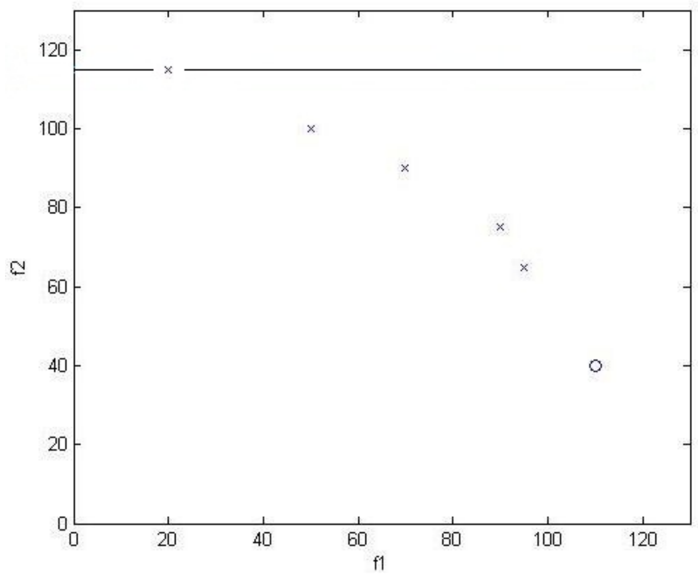

Figure 4.1: LexECM algorithm steps

LexECM algorithm will be explained with the help of Figure 4.1.

Before starting the while loop, ideal z_2 value is determined by solving (P_2) with $\epsilon_2 = 0$. This value is shown with the straight line in Figure 4.1 and is used as the stopping level for ϵ_1 in the loop since this it sets a boundary for the non-dominated set.

The loop in the first iteration starts with $\epsilon_1 = 0$ and solving (P_1) will give the ideal z_1 value. The optimal solution x^* of (P_1) is used in determining ϵ_2 for (P_2) as $\epsilon_2 = f_1(x^*)$. This way the nadir value of z_2 where z_1 is equal to $f_1(x^*)$ is calculated. This point is marked with the ring shape in Figure 4.1.

Then, in the second iteration the optimal value z_2 of the second model in the previous iteration is used in the righthand side of the quadratic constraint. Doing so lets us find a new non-dominated solution, the closest one to the ring shape shown in Figure 4.1, which has a worse z_1 and a better z_2 value than the previously found one.

The procedure continues by obtaining a non-dominated solution at each iteration closer to the level marked with the straight line and ends when (z_1^N, z_2^I) is found.

It has been observed that the quadratic constraint in the formulation of the lexicographic ϵ-constraint method for BQKP, increases the cpu times seriously. The time required to solve even a small sized problem gets unreasonably long. Therefore, LexECM is operated only on small sized problems. Computational results of LexECM are reported in the results chapter.

We did not find solving only small sized BQKP's adequate for this thesis. For this reason, we searched the literature for appropriate linearization methods that can be applied to QCQ subproblems in LexECM. Related work is explained in Chapter 3.4 and the implementation is explained in the following section.

Assume that the number of non-dominated solutions of one specific BQKP is α then the minimum number of single optimization problems CPLEX needs to solve to generate all non-dominated solutions is 2α. In addition, these 2α number of problems required to be solved are quadratically constrained quadratic (QCQ) models. From this, the computational challenge on finding the non-dominated solution set of the BQKP can be inferred. Therefore, a linearization method is also applied to the lexicographic ϵ-constraint code which will be explained in the next chapter.

4.2 Linearized LexECM

The LexECM code explained in the previous section is linearized with the method described in Chapter 3.4. The algorithmic structure of LexECM, which incorporates lexicographic ϵ-constraint method, is the same except the subproblem definitions (P_1) and (P_2) in it.

We will state the linearized subproblems in the while loop. (P_1) is in the form given below.

$$\begin{aligned}
minimize\ & g(s,x) = e^T s - Me^T x, \\
s.t.\ & (-P)x - y - s + Me = 0, \\
& (-w^T)x \geq -b, \\
& y \leq 2M(e-x), \\
& Qx - z + M'e \geq 0, \\
& e^T z - M'e^T x \geq \epsilon_1, \\
& z \leq 2M'x, \\
& x \in \{0,1\}^n, \\
& y_i,\ s_i,\ z_i \geq 0. \qquad (4.3)
\end{aligned}$$

Here, M' and M is defined as $M' = max_i \sum_{j=1}^{n} \mid q_{ij} \mid$ and $M = max_i \sum_{j=1}^{n} \mid p_{ij} \mid$. The second subproblem is named again as (P_2) and is stated below.

$$\begin{aligned}
minimize\ & g(s,x) = e^T s - M'e^T x, \\
s.t.\ & (-Q)x - y - s + M'e = 0, \\
& (-w^T)x \geq -b, \\
& y \leq 2M'(e-x), \\
& Px - z + Me \geq 0, \\
& e^T z - Me^T x \geq z_1^*, \\
& z \leq 2Mx, \\
& x \in \{0,1\}^n, \\
& y_i,\ s_i,\ z_i \geq 0. \qquad (4.4)
\end{aligned}$$

The problem of determining the ϵ values is addressed in the same way again. The non-dominated z_2^* value found in the previous iteration is used as the right hand side to the first model. This means searching for a smaller z_1 value, which gives the next non-dominated solution.

4.3 Finding Supported Non-dominated Solutions

In this subsection, the aforesaid algorithm by Aneja and Nair will be explained in the way we implemented it.

In our implementation, first (z_1^1, z_2^1) ideal and (z_1^2, z_2^2) nadir points are calculated with weights $(1,0)^T$ and $(0,1)^T$. If these points are equal, the algorithm stops with only one non-dominated solution. If these points are not equal, they are recorded in a structure array named *list* together which means that they constitute the first element of the *list* as a pair.

Then, the iteration through the *list* starts. The size of *list* gets bigger dynamically while the algorithm works. $w_1^{(1,2)} = |z_2^1 - z_2^2|$ and $w_2^{(1,2)} = |z_1^1 - z_1^2|$ are calculated from the first pair of supported non-dominated solutions in the *list*. BQKP is optimized by using the weights w_1 and w_2 in the weighted sum formulation for the first and the second objective functions, respectively.

If a different solution other than the parents is found, then it is recorded with its parents to the *list*. Here, two new elements are added to the structure array *list*, the new solution with one parent and the other.

If the new solution found in some iteration is equal to one of its parents, then no addition to the *list* is made. This procedure continues until all elements of the *list* is checked.

Eventually, BQKP is optimized with all the different weights calculated from the pair of solutions in the *list*. The distinct elements of the *list* gives the set of all supported non-dominated solutions.

The algorithmic expression of this procedure is given in Algorithm 3.

It is known that Aneja and Nair's algorithm may not find non-extreme supported solutions unless (4.5) is solved for all alternative optimal solutions. In this work we do not find alternative solutions of (4.5) and we may be missing non-extreme solutions. Nonetheless, since these solutions are rare in random experiments we use the term

Algorithm 3 A Special Weighted Sum Algorithm

```
Step 0: Calculate ideal and nadir points. Record these two solutions to the weight
        array list if different from each other, else stop.
Step 1: Choose a pair of solutions $(r, s) \in list$. Calculate weights $w_1^{(r,s)} = \mid z_2^s - z_2^r \mid$
        and $w_2^{(r,s)} = \mid z_1^s - z_1^r \mid$.
        Solve the weighted problem

            $maximize\ w_1^{(r,s)} z_1(x) + w_2^{(r,s)} z_2(x)$
            $subject\ to\ \ x \in X.$                                (4.5)

        Let $x^*$ be the optimal solution to the weighted problem.
        Calculate $z_1(x^*)$ and $z_2(x^*)$.
        If $(z_1(x^*), z_2(x^*))$ is not equal to an element in list record this pair
        with its parents to the list.
Step 2: If list is not empty go to Step 1. If list is empty, stop.
```

supported solution here [81].

The properties of test problems, the results and comments on them will be included in the next chapter.

Chapter 5

RESULTS

In this chapter, implementation characteristics and results of the study will be detailed. Summary tables of performance indicators such as the number of non-dominated solutions, cpu times will be given. Illustrations and coverage errors will also be included.

5.1 Implementation Characteristics

We will state the characteristics of randomly generated BQKP's in this section.
Elements of quadratic matrices; p_{ij} and q_{ij}, and all of the coefficients; elements of the constraint matrix w_j, are randomly generated nonnegative integers in the interval $[1, 100]$ in this study. The righthand side of the knapsack constraint is equal to the closest integer value of $b = \frac{\sum_{j=1}^{n} w_j}{2}$, which makes the knapsack constraint harder to satisfy.
N is the number of variables and Pct is the percentage of the nonzero elements in quadratic matrices. Pct takes the values 25, 50, 75 and 100 in random problems. For each problem size N and Pct value, the percentage fullness of quadratic matrices, thirty randomly generated problems are solved.
The LexECM code on CPLEX, which solves quadratically constrained quadratic sub-models, does not perform well. Sizes of the problems solved with LexECM are small because CPLEX can only handle up to 30 variables when the quadratic constraint is involved. Therefore, only problems of $N = 10, 15, 20$ with four different Pct values and $N = 25$ with $Pct = 25, 50$ are solved with this code. The addition of the quadratic constraint here significantly affects the difficulty of the problem and gives rise to this situation. The cpu times are dramatically increased, therefore we stopped

the tests for LexECM for more than $N = 25$ variables.

Larger problems are solved with the linearized LexECM code. These include problems of size $N = 50, 60, 70$ with four different Pct values and $N = 80$ with $Pct = 25, 50$. It is observed that the memory limit determined the size of the problems we have solved in this category.

The number of supported non-dominated solutions of these set of bigger sized problems is also found with the special weighted sum algorithm explained in the previous chapters.

All implementations of the LexECM, the linearized LexECM, the special weighted sum algorithm and the linearization of it are done in C using CPLEX Callable Library version 12.3 [46]. All tests are completed under a 4.00 GB memory limit and the time limit of queued jobs on the cluster we used for these implementations unless stated otherwise.

CPLEX's mixed integer quadratic problem (MIQP) solver is used in these applications. This routine is designed mainly to solve linearly constrained quadratic binary problems where the quadratic matrix is PSD (NSD for maximization problems). Our method of generating quadratic matrices does not guarantee these matrices to be NSD. CPLEX converts these matrices into the required NSD form before starting the MIQP solver.

The results of the tests are given in the next section.

5.2 Test Results and Comments

Performance indicators of the problems; the number of non-dominated, supported and non supported non-dominated solutions, and cpu times will be reported in this section. The analysis of these results will also be made.

The average, minimum, and maximum cpu times of the quadratically constrained quadratic problems solved with the LexECM code are given in the summary Table 5.1 for $N = 10, 15, 20$ with four different Pct values and $N = 25$ with $Pct = 25, 50$. The results are of 30 randomly generated problems for each problem type.

According to the results in Table 5.1, minimum and maximum cpu times of the

Table 5.1: Cpu Times (secs) of Quadratic LexECM

N/Pct		25			50	
	Avr.	Min.	Max.	Avr.	Min.	Max.
10	59	12	133	44	8	88
15	66	8	177	104	6	890
20	979	113	2,863	2,936	36	34,721
25	40,853	572	383,998	374,280	401	6,609,925

N/Pct		75			100	
	Avr.	Min.	Max.	Avr.	Min.	Max.
10	33	9	95	456	2	59
15	126	2	912	137	2	411
20	4,180	22	46,257	26,135	49	250,116

same sized problems differ noticeably. However, this is not surprising since the test problems are randomly generated. The maximum cpu time observed for the problem set $N = 25$ and $Pct = 50$ is unreasonably high. Problems of larger sizes could not be solved with quadratic LexECM because the time limit in the queue is reached on the cluster which is used for the implementations. Also, it was irrational to wait for this long solution times.

Table 5.2 shows the average, minimum and the maximum number of non-dominated solutions of the same problems solved with the quadratic LexECM.

The fact that few number of non-dominated solutions is found for these problem sets is noticed from Table 5.2. This can be expected since the problem sizes are small. Moreover, it can be said that two objectives in these cases are not in highly conflicting nature, and this could be an explanation of this situation. The minimum number of non-dominated solutions is equal to one for ten of fourteen problem sets of different types. This also indicates the nonconflicting nature of some of the randomly generated problems.

Problems with smaller Pct levels have more non-dominated solutions on average than bigger size of problems with larger Pct levels. The percentage fullness values of the

Table 5.2: # of Non-dominated Solutions of Quadratic LexECM

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
10	3.97	1	8	3.67	1	7
15	6.77	2	14	5.00	2	16
20	8.57	3	15	6.53	1	14
25	11.60	5	22	6.70	1	22
N/Pct	75			100		
	Avr.	Min.	Max.	Avr.	Min.	Max.
10	3.07	1	7	2.70	1	6
15	4.47	1	19	4.10	1	9
20	4.63	1	9	3.97	1	11

quadratic matrices may have an impact on the conflict characteristics of quadratic objectives.

The summary in Table 5.3 shows the average, minimum and maximum cpu times for randomly generated 30 problem sets of size $N = 50, 60, 70$ with four different Pct values and $N = 80$ with $Pct = 25, 50$. These problems are solved with the linearized version of the code LexECM. Problems with $Pct = 75, 100$ for $N = 80$ could not be solved. The main reason for this is reaching the 4.00 GB memory limit of the queue.

If we interpret cpu times of the linearized LexECM, we observe that the average solution times of larger problems are less than the cpu times of smaller problems solved with quadratic version of LexECM. Moreover, none of the maximum solution times of these problems is close to the solution time of the problem set of size $N = 25$, $Pct = 50$. Therefore, we can conclude that the linearization we applied to the QCQ submodels dramatically reduced the cpu times required to solve these problems. Also, from our tests with the weighted sum algorithm, we can state that the elimination of the quadratic constraint in the LexECM is the main reason behind this reduced cpu times. Larger quadratic problems can be solved with the weighted sum algorithm since a quadratic constraint is not involved as in the LexECM.

Another reason which affects cpu times is the number of non-dominated solutions.

Table 5.3: Cpu Times (secs) of the Linearized LexECM

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	1,054	192	3,817	11,362	110	236,964
60	2,534	251	22,357	88,911	745	1,580,073
70	19,874	1,480	124,504	54,252	245	946,137
80	130,335	797	931,633	178,568	20	2,724,783
N/Pct	75			100		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	79,503	31	1,565,712	5,957	18	31,443
60	75,846	195	672,989	119,680	52	833,409
70	168,292	45	1,226,883	167,720	25	1,766,202

When the number of non-dominated solutions is large for the same sized problems, cpu time required gets larger usually. This can easily be deducted since existence of larger number of non-dominated solutions mean that the algorithm has to iterate many times.

It is also evident that the difficulty of the specific random problem has an impact on cpu times.

Figure 5.1 shows cpu time distributions of four biggest problem sets $N = 70$ with $Pct = 75, 100$ and $N = 80$ with $Pct = 25, 50$. It can be observed that cpu times are below $500,000$ for most of the random problems. Most probably structural properties of these problems or queuing issues in the cluster cause these high outlier cpu times.

The number of random problems in different cpu time intervals for four biggest problem sets $N = 70$ with $Pct = 75, 100$ and $N = 80$ with $Pct = 25, 50$ are given in Figure 5.2. 40 random problems in 160 are solved in less than $5,000$ cpu seconds and 11 of them required more than $500,000$ cpu seconds in the cluster.

The details on the number of non-dominated solutions; average, minimum, and maximum, for the problems solved with the linearized LexECM are given in Table 5.4. The average number of non-dominated solutions gets bigger when the size N of the problem gets larger. Moreover, for higher Pct values of same sized problems the

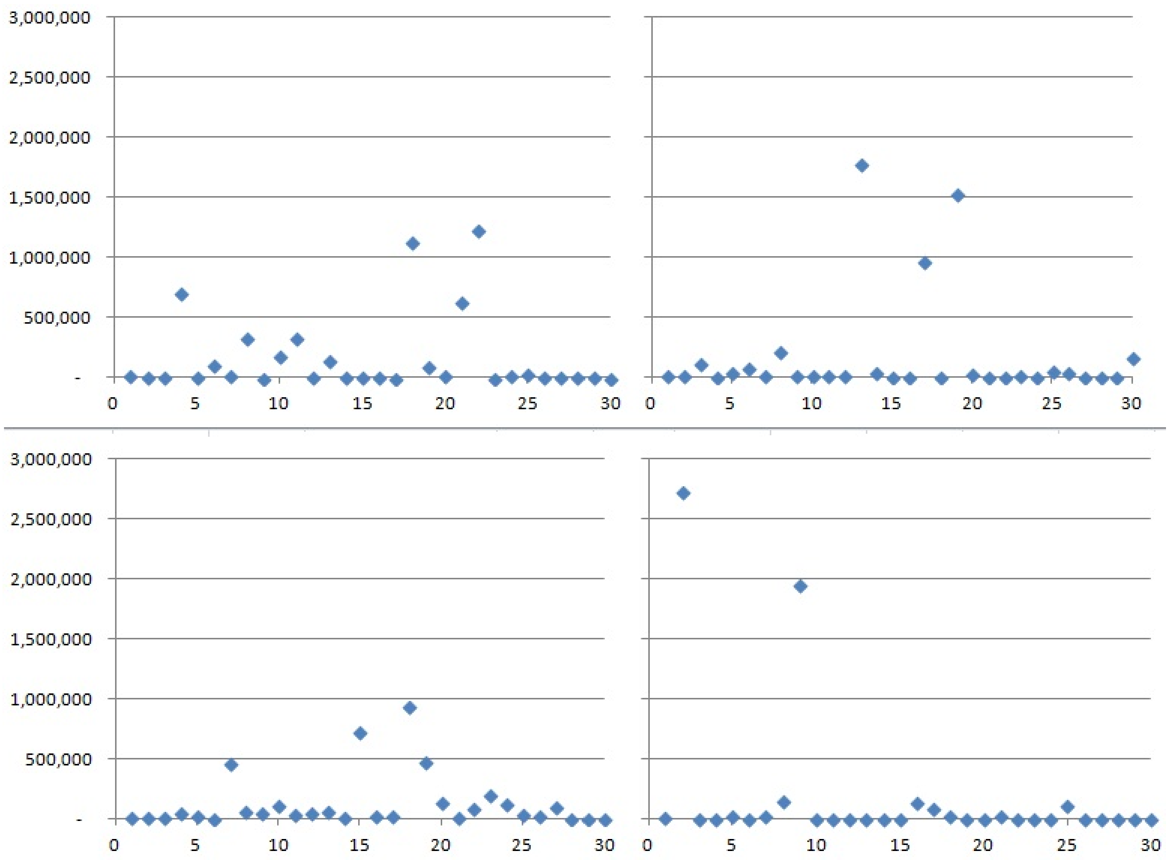

Figure 5.1: Cpu time plots of N=70 with Pct=75,100 and N=80 with Pct=25,50

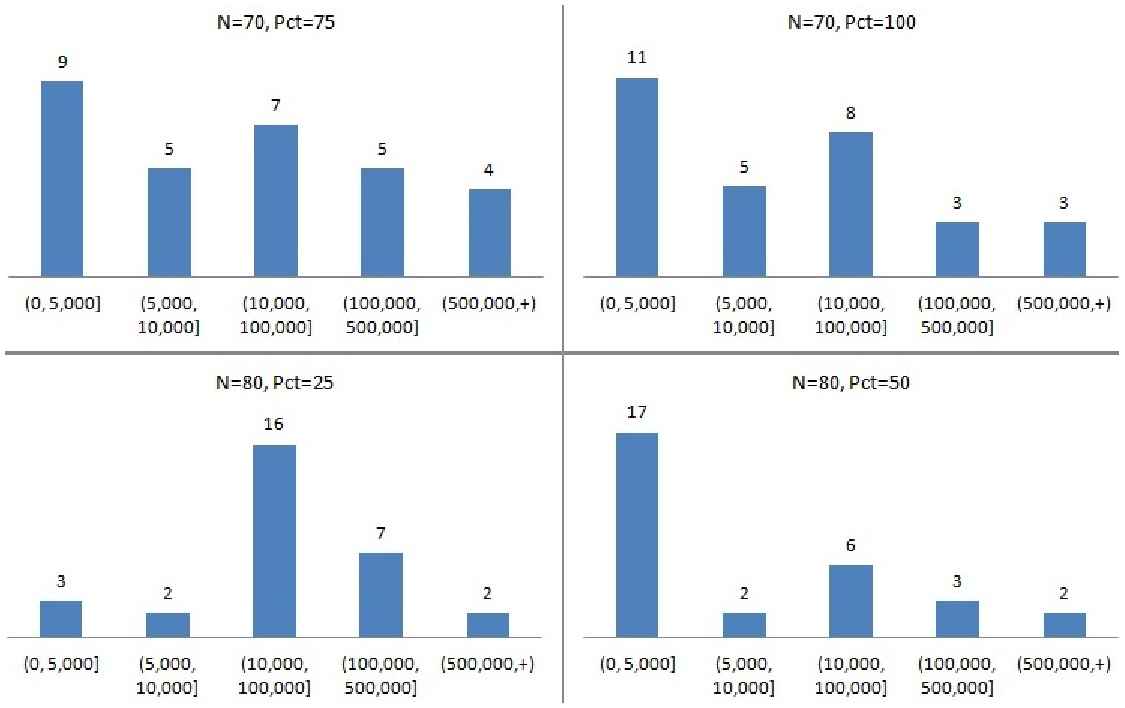

Figure 5.2: Count of random problems in cpu time intervals

average number of non-dominated solutions is decreased.

The summary in Table 5.5 shows the average, minimum and maximum cpu times for the same randomly generated 30 problem sets of size $N = 50, 60$ with four different Pct values and $N = 70$ with $Pct = 25, 50, 75$. These problems are solved again with the Special Weighted Sum Algorithm described in the previous chapters in order to generate the supported non-dominated solution set of BQKP.

It should be clarified that test problems bigger than $N = 70$ and $Pct = 75$ could not be solved with the Special Weighted Sum Algorithm since the memory limit on the cluster is exceeded. The summary in Table 5.6 shows the average, minimum and maximum cpu times for the same randomly generated 30 problem sets of size $N = 50, 60, 70$ with four different Pct values and $N = 80$ with $Pct = 25, 50$. These problems are solved with the linearized version of the Special Weighted Sum Algorithm. Therefore, the cpu times of these instances are low compared to the quadratic version of the algorithm.

Table 5.4: # of Non-dominated Solutions of the Linearized LexECM

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	20.87	10	54	15.53	5	32
60	24.27	9	39	15.77	6	27
70	30.67	16	59	14.6	5	32
80	37.63	12	63	20.57	6	48
N/Pct	75			100		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	10.73	2	28	9.4	2	24
60	11.7	2	32	13.7	1	27
70	13.6	3	29	12.13	2	25

Table 5.5: Cpu Times (secs) of the Special Weighted Sum Algorithm

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	437	15	3,026	5,463	52	79,970
60	3,798	96	28,339	89,037	172	1,868,775
70	89,608	2,409	793,437	92,500	1,208	490,037
N/Pct	75			100		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	18,973	13	322,046	4,243	69	25,774
60	59,400	225	643,393	75,416	492	521,041
70	299,170	363	2,289,104	-	-	-

Solution times are strongly related to the number of iterations the algorithm incorporates until finding all solutions. Moreover, the minimum and the maximum cpu times differ dramatically for all tests in general. Mainly, these random hard instances prevented us from solving large instances. If a cpu time limit is incorporated then bigger sized problems can be solved with these algorithms.

The average, minimum and maximum number of supported non-dominated solutions found are given in Table 5.7. The average number of supported non-dominated solutions is less than half of the average number of non-dominated solutions for all types

Table 5.6: Cpu Times (secs) of the Linearized Special Weighted Sum Algorithm

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	39	3	276	440	5	8,714
60	126	7	758	2,229	3	43,389
70	1,176	52	8,731	1,807	20	16,801
80	6,535	97	48,549	25,095	11	288,172
N/Pct	**75**			**100**		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	2,821	1	67,803	672	1	5,131
60	1,163	4	13,207	3,351	7	30,351
70	8,481	35	77,095	13,802	18	157,065

of test problems. We examined how well these supported non-dominated solutions represent the entire non-dominated set in the next section by using quality metrics of representations.

Additional testing is done regarding the cpu times of the Linearized Special Weighted

Table 5.7: # of Supported Non-dominated Solutions

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	6.63	4	10	5.63	3	11
60	7.37	5	12	6.07	2	10
70	8.83	5	12	5.50	3	9
80	8.47	5	14	6.07	2	11
N/Pct	**75**			**100**		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	4.57	2	9	4.13	2	7
60	4.47	2	7	5.10	1	9
70	5.23	3	11	4.77	2	7

Sum Algorithm (WSA). We wanted to observe how the solution times will differ when these problems are solved on a personal computer (PC) not on the cluster with possible queuing issues. The average, minimum and maximum cpu times for the same randomly generated 30 problem sets of size $N = 50, 60$ with four different Pct values are showed in Table 5.8. Average solution times on the personal computer are less than the average solution times on the cluster for all of the tested problem types. Before, we had discussed the possibility of queueing issues on the cluster we worked on. It can be said that these results justifies this discussion.

In quadratic problems the structure of quadratic matrices determines the charac-

Table 5.8: Cpu Times (secs) of the Linearized Special WSA - on PC

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	11	1	76	178	1	3,909
60	53	3	352	1,070	1	21,585
N/Pct	75			100		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	1,505	0	36,816	86	1	462
60	605	2	6,422	2,220	3	19,607

Table 5.9: Cpu Times (secs) of the Tri-diagonal Linearized LexECM - on PC

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	4	0.44	12.14	8	3.31	16.68
60	6	0.03	13.14	15	5.20	40.53
70	9	2.24	21.21	28	11.90	78.60
80	11	0.02	22.25	37	12.67	103.02
N/Pct	75			100		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	8	2.72	23.90	5	1.69	17.06
60	12	4.47	36.78	14	1.94	32.82
70	20	7.51	50.17	10	3.49	18.08

teristics of the problem. In order to observe this effect tri-diagonal symmetric random matrices are generated. A $tri-diagonal\ matrix$ has nonzero elements only on the

Table 5.10: # of Non-dominated Solutions of the Tri-Diagonal Problems

N/Pct	25			50		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	33	9	72	40	19	69
60	40	1	67	57	25	96
70	96	1	94	79	43	131
80	56	1	94	92	51	172
N/Pct	75			100		
	Avr.	Min.	Max.	Avr.	Min.	Max.
50	32	14	68	17	7	35
60	37	17	80	24	9	46
70	50	26	83	25	9	40

main diagonal, the first diagonal below, and the first diagonal above the main diagonal.

Problems with these matrices are solved with the Linearized LexECM Algorithm on the personal computer. The cpu time results are shown in Table 5.9. The solution times are dramatically reduced. The structural matrices made BQKP easier to solve compared with the nonstructural version.

The number of non-dominated solutions of these problem sets are showed in Table 5.10. BQKP with symmetric tri-diagonal quadratic matrices have higher number of non-dominated solutions compared with the nonstructural matrices version. The reason behind this may be the fact that the problem resembles the linear version more with tri-diagonal matrices.

5.3 Coverage Error Calculations

We obtained the representation set for BQKP, which consists of all the supported non-dominated solutions, with the special weighted sum algorithm. The analysis of the quality of the representation set should be made. For this purpose, we calculated coverage errors between these two sets; the entire non-dominated solution set and the set of supported non-dominated solutions.

The quality of the representation set can be measured in different ways. As proposed in [91], three dimensions are needed to be checked when determining the quality of the discrete representation of a set. These dimensions are coverage, uniformity and cardinality. The coverage dimension controls whether or not, all of the elements of the original set is well represented. The uniformity requires the elements of the representation to be distributed uniformly, in the sense that the elements of the representation does not form clusters. Lastly, the number of elements of the representation set should be reasonable and this is expressed with the cardinality dimension.
Two different coverage error calculations from different papers are described in the following. Actually, the insight behind these methods is close in a sense. The second method is chosen to be applied to our problem.
First method is given in the paper cited above. Before we give the coverage error definition in [91], we need some definitions. These two definitions are given below.

Definition 7. *Let $C > 0$ be a real number. Let $D \subseteq Z$ be a discrete set. D is called a d_C-representation of Z if for any $z \in Z$, there exists $y \in D$ such that $d(z, y) \leq C$.*

Given a d_C-representation D of a set Z, the coverage error is defined in [91] as

$$C = max_{z \in Z} min_{x \in D} d(z, x). \tag{5.1}$$

In our case, the set Z is the non-dominated solutions set and the representation set D is the set of all supported non-dominated solutions.
The coverage error is used to compare different representations, usually. The method described above is more appropriate for such use. Therefore, another coverage error calculation is applied in this thesis.
We incorporated two different coverage error calculations d_1 and d_2, and the ratio of these values, all defined in [103]. The interpretation here is; the lower this ratio, the higher the quality of the representation.
To be able to define $d_1(Z_{sE}, Z_E)$ and $d_2(Z_{sE}, Z_E)$, let Z_E be the set of non-dominated solutions and Z_{sE} be the set of supported non-dominated solutions. Let $x \in Z_E$ and

$y \in Z_{sE}$. $d(x,y)$ is defined as below.

$$d(x,y) = \sum_{i=1}^{p} w_i |f_i(x) - f_i(y)| \tag{5.2}$$

Here, w_i is the weight related to criterion f_i, and p is the number of criteria.

$$w_i = \frac{1}{\Delta_i} \ with \ \Delta_i = max_{x \in Z_E} f_i(x) - min_{x \in Z_E} f_i(x) \tag{5.3}$$

$d'(Z_{sE}, x)$ the distance between $x \in Z_E$ and the closest solution $y \in Z_{sE}$ is given as such;

$$d'(Z_{sE}, x) = min_{y \in Z_{sE}} d(y, x). \tag{5.4}$$

Finally, with the help of above definitions formulations of $d_1(Z_{sE}, Z_E)$ and $d_2(Z_{sE}, Z_E)$ can be given.

$$d_1(Z_{sE}, Z_E) = \frac{1}{|Z_E|} \sum_{x \in Z_E} d'(Z_{sE}, x) \tag{5.5}$$

$$d_2(Z_{sE}, Z_E) = max_{x \in Z_E} d'(Z_{sE}, x) \tag{5.6}$$

$d_1(Z_{sE}, Z_E)$ represents the average distance between Z_{sE} and Z_E. $d_2(Z_{sE}, Z_E)$ represents the worst case distance between Z_{sE} and E. Therefore, it is clear that lower the ratio of $d_2(Z_{sE}, Z_E)$ and $d_1(Z_{sE}, Z_E)$ higher the quality of the representation set Z_{sE}.

$$Ratio = \frac{d_2(Z_{sE}, Z_E)}{d_1(Z_{sE}, Z_E)} \tag{5.7}$$

Table 5.11 shows d_1, d_2 and *Ratio* values for the supported efficient set and the entire efficient set of our test problems. Averages of these three indicator values are close for all problem types. d_1 the average distance between the supported efficient set and the efficient set and d_2 the worst case distance between the supported efficient set

Table 5.11: Coverage Results

N/Pct	25			50		
	d_1	d_2	*Ratio*	d_1	d_2	*Ratio*
50	0.12	0.30	2.76	0.14	0.35	2.89
60	0.10	0.28	2.87	0.12	0.34	2.99
70	0.08	0.23	3.01	0.13	0.35	3.01
80	0.09	0.24	2.83	0.13	0.35	2.74

N/Pct	75			100		
	d_1	d_2	*Ratio*	d_1	d_2	*Ratio*
50	0.11	0.31	2.61	0.13	0.37	2.69
60	0.17	0.42	2.80	0.14	0.38	2.94
70	0.12	0.32	2.72	0.17	0.43	2.81

and the efficient set, take highest average values 0.17 and 0.43, respectively, for the random problem set of size $N = 70$ and percentage of fullness $Pct = 100$. d_1 and d_2 take the minimum average values 0.08 and 0.23, respectively, for the random problem set of size $N = 70$ and percentage of fullness $Pct = 25$. However, when we consider the ratio of d_1 and d_2 for these two problem sets, the one with the worst d_1 and d_2 values have the smallest coverage ratio value $Ratio = 2.81$ on average.

These three coverage indicator values d_1, d_2 and *Ratio* should be interpreted by considering what features are expected primarily from the representation set. In this case, d_1 values are low which indicates that on average Z_{sE} represents Z_E well. On the other hand, d_2 values are not very low which may cause large errors.

5.4 Comparison of Non-dominated Sets

The number of non-dominated solutions in our calculations is small. There are two probable reasons for this situation. The first reason is the fact that the size of the random problems are not so big. Secondly, the nonconflicting nature of quadratic objectives might cause this situation.

We tried to examine this outcome by enumerating all feasible solutions for small sized bi-objective linear and quadratic knapsack problems. The illustration of these solu-

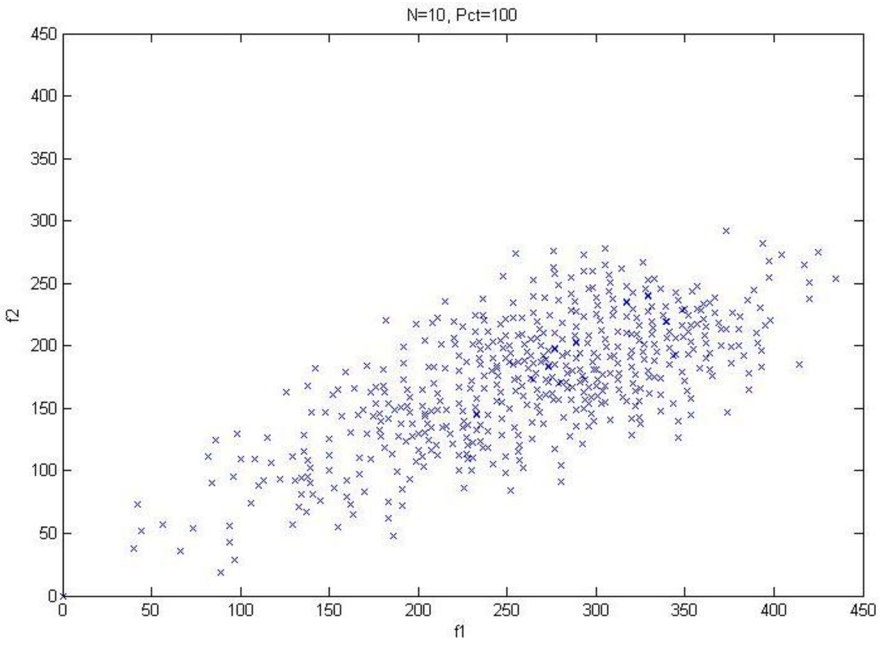

Figure 5.3: Bi-objective Linear Knapsack Problem Pct=100

tions is observed to see how the feasible solutions are distributed in the solution space. In this study, we are also interested in how the non-dominated set of our problem BQKP looks like. The insight behind the non-supported and supported non-dominated solutions will be clarified by observing plots of the non-dominated set. Therefore, we have included some illustrations of test problems in this section.
In order to see the difference between the appearance of the feasible solution sets of the bi-objective linear and the quadratic knapsack problems, we enumerate and plot the feasible solutions of these problems in the MATLAB environment. The distribution of the feasible solutions in the solution space can be observed by these plots. Moreover, comparison between the plots of bi-objective linear and quadratic knapsack problems can be made.
The feasible solution set of a bi-objective linear knapsack problem with $N = 10$ and $Pct = 100$ can be seen in Figure 5.3.

The distribution of the feasible solution set of the same bi-objective linear knapsack problem with $N = 10$ and decreased percentage of fullness values $Pct = 50$ and

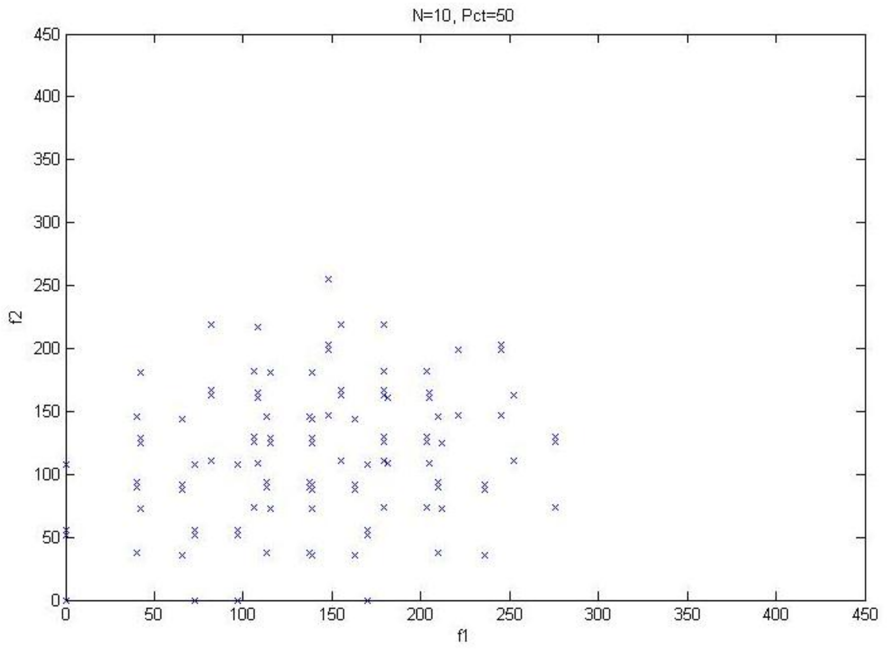

Figure 5.4: Bi-objective Linear Knapsack Problem Pct=50

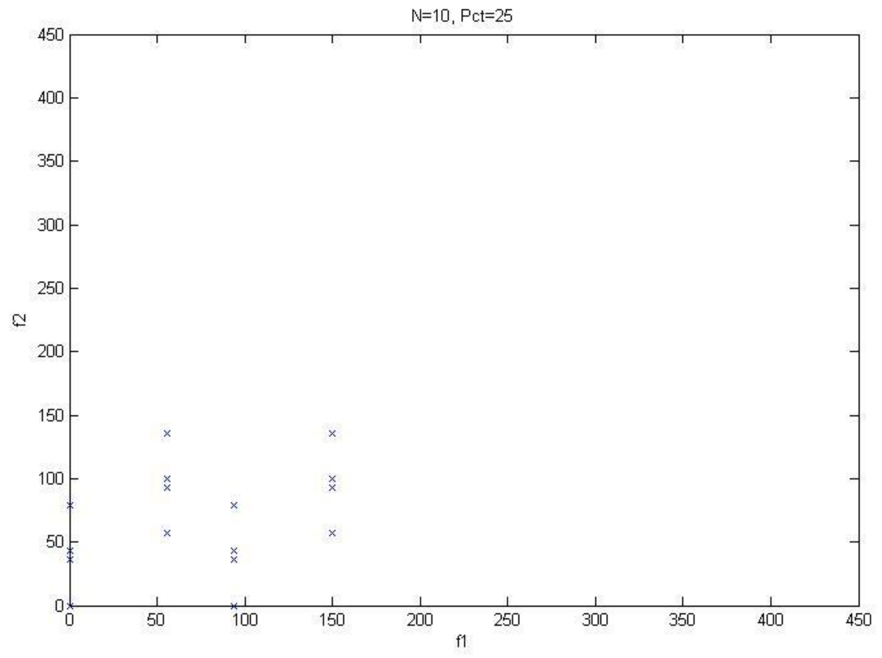

Figure 5.5: Bi-objective Linear Knapsack Problem Pct=25

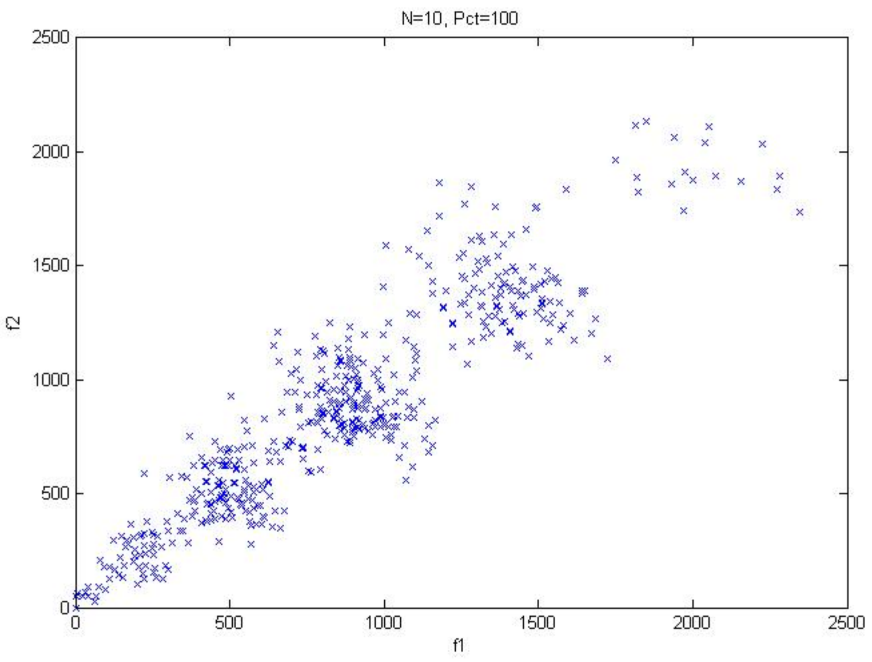

Figure 5.6: BQKP Pct=100

$Pct = 25$ are given in Figure 5.4 and Figure 5.5, respectively. From these figures, we can observe that the feasible solution set expands when the percentage of fullness is decreased.

BQKP's with $N = 10$ and percentage fullness of $Pct = 100$, $Pct = 50$ and $Pct = 25$ are illustrated in Figures 5.6, 5.7 and 5.8, respectively. It should be noted that these BQKP's have the same knapsack constraint as the bi-objective linear knapsack problems included above.

Comparing with the linear form of the problem for the same Pct values, the feasible solution set and the non-dominated frontier of the quadratic version are more compact. It can also be seen from these figures that the feasible solution set expands when the percentage of fullness is decreased, which means that the problem approaches the linear form.

The plots of the non-dominated sets of our test problems will be given here. The non-dominated set of the random BQKP with $N = 80$ and percentage fullness of $Pct = 50$

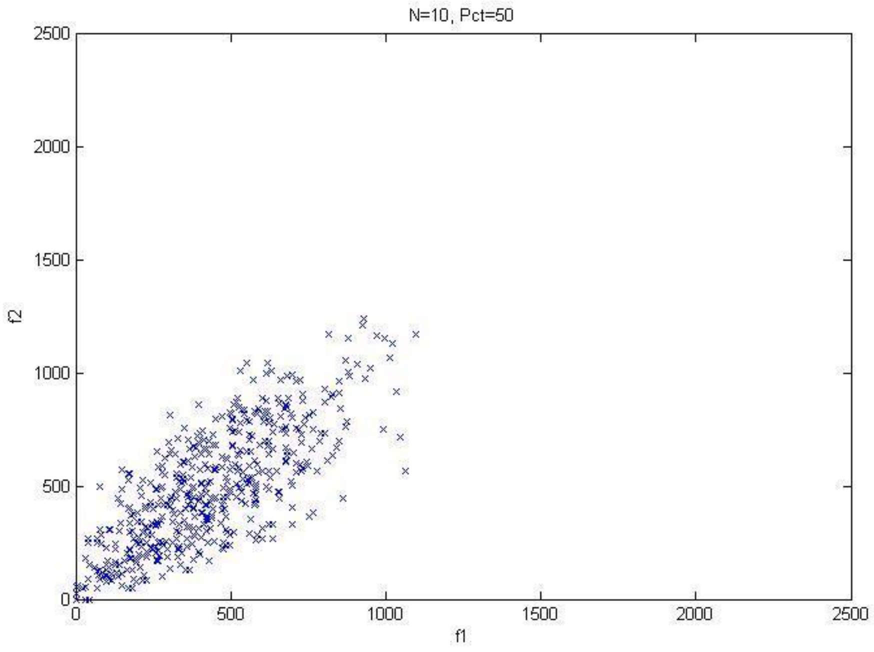

Figure 5.7: BQKP Pct=50

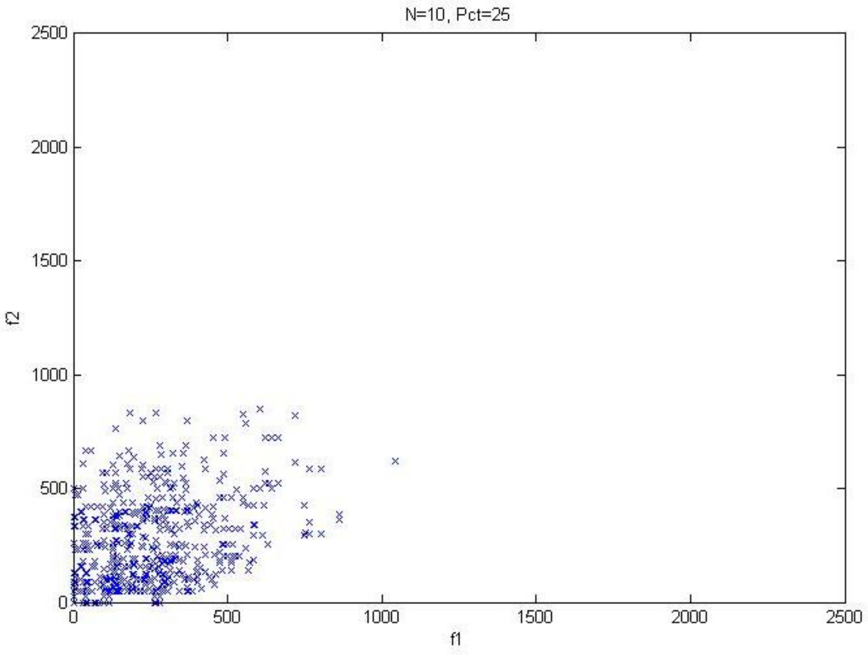

Figure 5.8: BQKP Pct=25

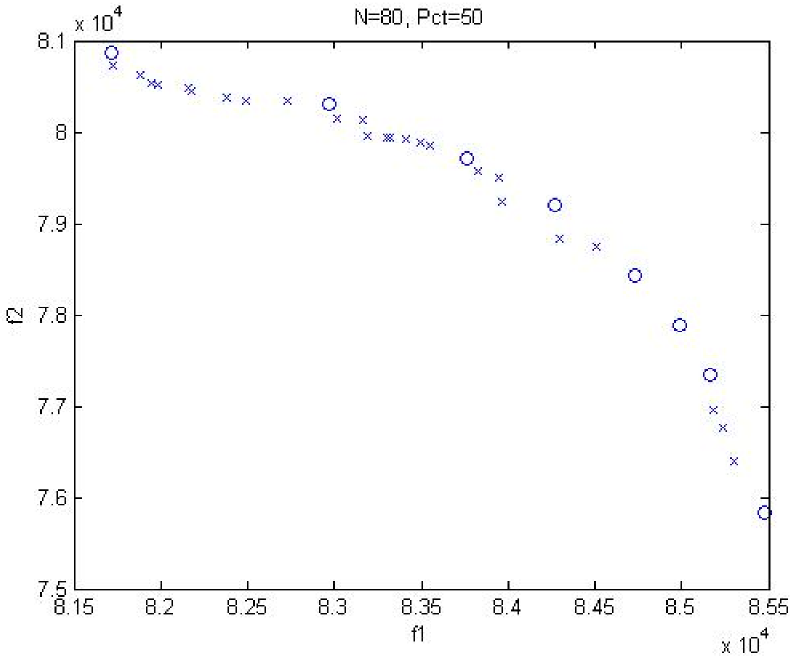

Figure 5.9: Non-dominated Set of N=80, Pct=50

can be observed in Figure 5.9. The rings represent the supported non-dominated solutions and the crosses represent the non-supported non-dominated solutions. This problem has a total number of 33 non-dominated solutions and 8 of them belong to the supported non-dominated set.

The non-supported non-dominated solutions cannot be found with linear weights. This characteristic of non-supported solutions can be realized from Figure 5.9. The ideal and the nadir solution pairs are the ones at the bottom and the top of the plot. The coverage values for this random problem are $d_1 = 0.09$, $d_2 = 0.2$ and $Ratio = 2.39$. As we have pointed out before, d_1 and d_2 represent the average and the worst case distances between the supported non-dominated set and the entire non-dominated set, respectively.

Figure 5.10 shows the non-dominated set of the random BQKP of size $N = 80$ and percentage fullness of $Pct = 25$. This problem has a total number of 48 non-dominated solutions and 9 of them are supported non-dominated solutions. The coverage values for this random problem are $d_1 = 0.09$, $d_2 = 0.26$ and $Ratio = 2.89$.

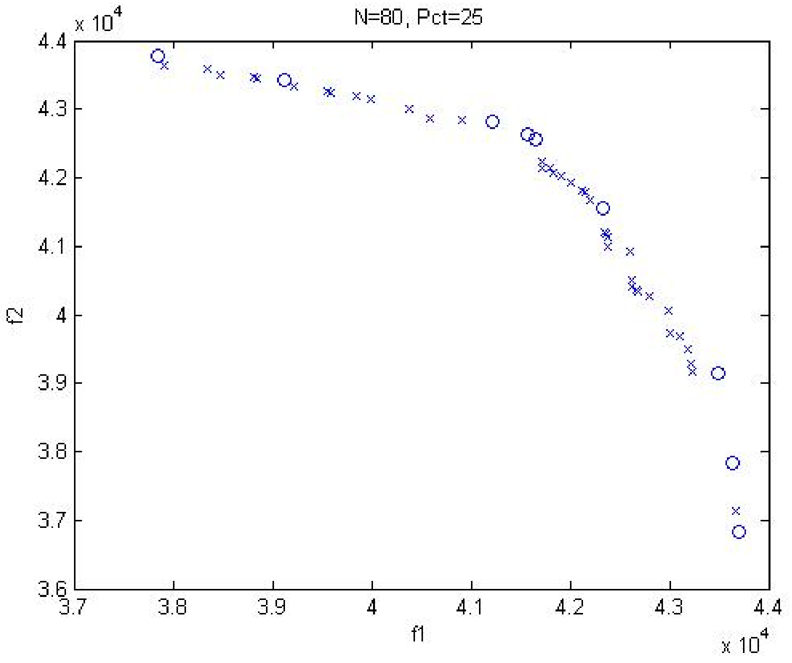

Figure 5.10: Non-dominated Set of N=80, Pct=25

It can be observed that non-supported solutions are heaped up in some parts of the non-dominated frontier.

The last non-dominated frontier illustration we will include again belongs to a random BQKP of size $N = 80$ and percentage fullness of $Pct = 50$. This problem has a total number of 15 non-dominated solutions and only 3 of them are supported non-dominated solutions. The coverage values for this random problem are $d_1 = 0.23$, $d_2 = 0.58$ and $Ratio = 2.51$.

According to these last three plots, the first example's non-dominated frontier can be best represented with the supported non-dominated set since the coverage ratio is the minimum for this problem. d_2, the worst case distance between the supported non-dominated set and the entire non-dominated set is the maximum for the last problem. The fewer number of supported non-dominated solutions might cause this situation.

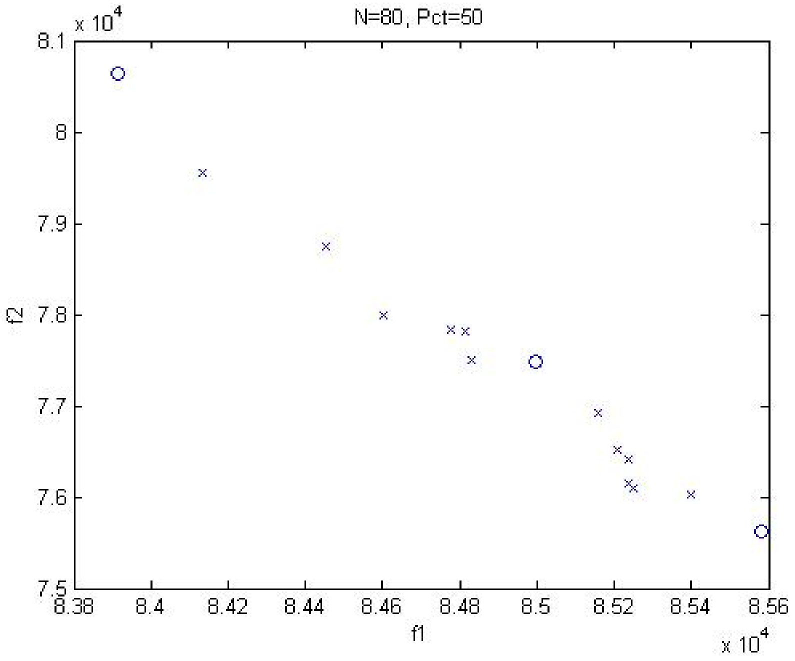

Figure 5.11: Non-dominated Set of N=80, Pct=50

Chapter 6

CONCLUSIONS

Bi-objective quadratic knapsack problem is studied in this thesis. The entire non-dominated solution set and the supported non-dominated solutions are found by incorporating different exact solution methods in the literature. Results and coverage errors of random problems are reported and some illustrations are given on the subject.

BQKP is a challenging problem to solve in general. It requires a considerable effort and time when trying to solve the problem with the commercial solver CPLEX. Moreover, large sized instances can not be solved. We will state some research areas that can be improved or developed concerning BQKP in this section.

Further research improvements on this topic may include studies on solving large instances of BQKP in a more efficient way with less solution times. In order to solve large sized instances of BQKP, we proposed two methods in the following paragraphs.

Exact solution methods are used in this thesis. Approximations or heuristics specifically designed for BQKP may be developed for further research. Since BQKP is a computationally challenging problem despite its simple structure, studying on developing approximations or heuristics for BQKP is reasonable.

More can be done on the exact solution methods side. Developing a specific and efficient algorithm to solve BQKP could be the next step on this study. Some structural properties of BQKP and the characteristics of the random problems generated can be used in this algorithm. In this manner, BQKP can be solved in a more effective way.

Another idea which can be implemented to widen the discussions on this thesis may be the comparison with the literature. Some known random test problems given in the literature can be solved and thus the performance of the algorithms in this thesis can be reported and compared.

Moreover, multi-objective versions of QKP with three or more objective functions can be addressed in detail as further research. Again, specifically designed exact algorithms for this problem can be proposed or approximations can be generated.

In conclusion, BQKP is a significant problem in the multi-objective literature. Its core problems KP and QKP are well studied in their fields. These problems will continue to appeal to researchers in the optimization discussions for the forthcoming years.

BIBLIOGRAPHY

[1] Md Mostofa Akbar, M. Sohel Rahman, M. Kaykobad, E.G. Manning, and G.C. Shoja. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers & Operations Research*, 33(5):1259 – 1273, 2006.

[2] M.A. Al-Fawzan and Mohamed Haouari. A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96(2):175 – 187, 2005.

[3] Y. P. Aneja and K. P. K. Nair. Bicriteria transportation problem. *Management Science*, 25(1):pp. 73–78, 1979.

[4] Claudia Archetti, Luca Bertazzi, and M. Grazia Speranza. Reoptimizing the 0-1 knapsack problem. *Discrete Applied Mathematics*, 158(17):1879 – 1887, 2010.

[5] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. Solving efficiently the 0-1 multi-objective knapsack problem. *Computers & Operations Research*, 36(1):260 – 279, 2009. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

[6] Krish Bhaskar. A multiple objective approach to capital budgeting. *Accounting and Business Research*, 10(37):25–46, 1979.

[7] Alain Billionnet and Sourour Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109:55–68, 2007. 10.1007/s10107-005-0637-9.

[8] Sylvain Boussier, Michel Vasquez, Yannick Vimont, Said Hanafi, and Philippe Michelon. A multi-level search strategy for the 0-1 multidimensional knapsack problem. *Discrete Applied Mathematics*, 158(2):97 – 109, 2010.

[9] V. J. Bowman. On the relationship of the tchebycheff norm and the efficient frontier of multiple-criteria objectives. 130:76–85, 1976.

[10] Kurt M Bretthauer and Bala Shetty. The nonlinear knapsack problem - algorithms and applications. *European Journal of Operational Research*, 138(3):459 – 472, 2002.

[11] Kurt M. Bretthauer, Bala Shetty, and Siddhartha Syam. A projection method for the integer quadratic knapsack problem. *The Journal of the Operational Research Society*, 47(3):pp. 457–462, 1996.

[12] P. M. Pardalos C. Zopounidis, editor. *Handbook of Multicriteria Analysis.* Applied Optimization. Springer-Verlag, Berlin, 2010.

[13] Victor C.B. Camargo, Leandro Mattiolli, and Franklina M.B. Toledo. A knapsack problem as a tool to solve the production planning problem in small foundries. *Computers & Operations Research*, 39(1):86 – 92, 2012. Special Issue on Knapsack Problems and Applications.

[14] Giulio Cantarella and Antonino Vitetta. The multi-criteria road network design problem in an urban area. *Transportation*, 33:567–588, 2006. 10.1007/s11116-006-7908-z.

[15] Pisinger D. Toth P. Caprara, A. Exact solution of the quadratic knapsack problem. pages 125–137, 1999.

[16] M.G. Castillo Tapia and C.A.C. Coello. Applications of multi-objective evolutionary algorithms in economics and finance: A survey. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 532 –539, sept. 2007.

[17] Wanpracha Chaovalitwongse, Panos M Pardalos, and Oleg A Prokopyev. A new linearization technique for multi-quadratic 0-1 programming problems. *Operations Research Letters*, 32(6):517 – 522, 2004.

[18] N. Cherfi and M. Hifi. A column generation method for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, 46:51–73, 2010. 10.1007/s10589-008-9184-7.

[19] Altannar Chinchuluun and Panos Pardalos. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154:29–50, 2007. 10.1007/s10479-007-0186-0.

[20] C.A. Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *Computational Intelligence Magazine, IEEE*, 1(1):28 – 36, feb. 2006.

[21] Carlos Coello Coello. Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. *Frontiers of Computer Science in China*, 3:18–30, 2009. 10.1007/s11704-009-0005-7.

[22] Carlos Gomes da Silva, Joo Clmaco, and Jos Rui Figueira. Core problems in bi-criteria -knapsack problems. *Computers & Operations Research*, 35(7):2292 – 2306, 2008. Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization.

[23] K. Deb. Evolutionary algorithms for multicriterion optimization in engineering design. In *In Proc. of Evolutionary Algorithms in Engineering and Computer Science, EUROGEN'99*, 1999.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182 –197, apr 2002.

[25] Charles Delort and Olivier Spanjaard. Using bound sets in multiobjective optimization: Application to the biobjective binary knapsack problem. 6049:253–265, 2010.

[26] Christina Diakaki, Evangelos Grigoroudis, and Dionyssia Kolokotsa. Towards a multi-objective optimization approach for improving energy efficiency in buildings. *Energy and Buildings*, 40(9):1747 – 1754, 2008.

[27] M. Laumanns E. Zitzler and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization.

[28] Moshe Eben-Chaime. Parametric solution for linear bicriteria knapsack models. *Management Science*, 42(11):pp. 1565–1575, 1996.

[29] M. Ehrgott. *Multicriteria Optimization*. Springer-Verlag, Berlin, second edition, 2005.

[30] Matthias Ehrgott and David M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of Multi-Criteria Decision Analysis*, 11(3):139–150, 2002.

[31] Thomas Erlebach, Hans Kellerer, and Ulrich Pferschy. Approximating multiobjective knapsack problems. *Management Science*, 48(12):pp. 1603–1612, 2002.

[32] M. Farina, K. Deb, and P. Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *Evolutionary Computation, IEEE Transactions on*, 8(5):425 – 442, oct. 2004.

[33] R. Farmani, D. A. Savic, and G. A. Walters. Evolutionary multi-objective optimization in water distribution network design. *Engineering Optimization*, 37(2):167–183, 2005.

[34] Krzysztof Fleszar and Khalil S. Hindi. Fast, effective heuristics for the 0-1 multidimensional knapsack problem. *Computers & Operations Research*, 36(5):1602 – 1607, 2009. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

[35] Kostas Florios, George Mavrotas, and Danae Diakoulaki. Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms. *European Journal of Operational Research*, 203(1):14 – 21, 2010.

[36] P. L. Hammer G. Gallo and B. Simone. Quadratic knapsack problems.

[37] Xavier Gandibleux and Arnaud Freville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objectives case. *Journal of Heuristics*, 6:361–383, 2000. 10.1023/A:1009682532542.

[38] D. Pisinger H. Kellerer, U. Pferschy. *Knapsack Problems.* Springer, Berlin, Germany, 2004.

[39] Julia Handl, Douglas B. Kell, and Joshua Knowles. Multiobjective optimization in bioinformatics and computational biology. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(2):279–292, April 2007.

[40] C. Helmberg, F. Rendl, and R. Weismantel. A semidefinite programming approach to the quadratic knapsack problem. *Journal of Combinatorial Optimization*, 4:197–215, 2000. 10.1023/A:1009898604624.

[41] Mhand Hifi and Lei Wu. An equivalent model for exactly solving the multiple-choise multidimensional knapsack problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 3(3), 2012.

[42] Andrew J. Higgins, Stefan Hajkowicz, and Elisabeth Bui. A multi-objective model for environmental investment decision making. *Computers & Operations*

Research, 35(1):253 – 266, 2008. Part Special Issue: Applications of OR in Finance.

[43] Christian Roed Pedersen Horst W. Hamacher and Stefan. Finding representative systems for discrete bicriterion optimization problems. *Operations Research Letters*, 35(3):336 – 344, 2007.

[44] Zhongsheng Hua, Bin Zhang, and Liang Liang. An approximate dynamic programming approach to convex quadratic knapsack problems. *Computers &; Operations Research*, 33(3):660 – 673, 2006.

[45] L.D. Iasemidis, P. Pardalos, J.C. Sackellares, and D.-S. Shiau. Quadratic binary programming and dynamical system approach to determine the predictability of epileptic seizures. *Journal of Combinatorial Optimization*, 5:9–26, 2001. 10.1023/A:1009877331765.

[46] ILOG. Ilog cplex 12.3 reference manual. *ILOG CPLEX Division.*

[47] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2419 –2426, june 2008.

[48] A. A. Islier. A genetic algorithm approach for multiple criteria facility layout design. *International Journal of Production Research*, 36(6):1549–1569, 1998.

[49] S. Greco J. Figueira and M. Ehrgott. *Multiple Criteria Decision Analysis: State of the Art Surveys.* Springer, New York, 2005.

[50] Larry Jenkins. A bicriteria knapsack program for planning remediation of contaminated lightstation sites. *European Journal of Operational Research*, 140(2):427 – 433, 2002.

[51] David J. Rader Jr. and Gerhard J. Woeginger. The quadratic 0-1 knapsack problem with series-parallel support. *Operations Research Letters*, 30(3):159 – 166, 2002.

[52] Dong-Oh Kang, Sung-Hun Kim, Heyoung Lee, and Zeungnam Bien. Multi-objective navigation of a guide mobile robot for the visually impaired based on intention inference of obstacles. *Autonomous Robots*, 10:213–230, 2001. 10.1023/A:1008990105090.

[53] Hans Kellerer and Vitaly Strusevich. The symmetric quadratic knapsack problem: approximation and scheduling applications. *4OR: A Quarterly Journal of Operations Research*, pages 1–51. 10.1007/s10288-011-0180-x.

[54] Hans Kellerer and Vitaly Strusevich. Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. *Algorithmica*, 57:769–795, 2010. 10.1007/s00453-008-9248-1.

[55] Sayn S. Kirlik, G. A new algorithm for generating all non-dominated solutions for multiobjective discrete optimization problems. *Working paper*, 2012.

[56] Krzysztof Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming*, 112:473–491, 2008. 10.1007/s10107-006-0050-z.

[57] Kathrin Klamroth and Margaret M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics (NRL)*, 47(1):57–76, 2000.

[58] Murat Koksalan and Banu Lokman. Approximating the nondominated frontiers of multi-objective combinatorial optimization problems. *Naval Research Logistics (NRL)*, 56(2):191–198, 2009.

[59] Stavros G. Kolliopoulos and George Steiner. Partially ordered knapsack and applications to scheduling. *Discrete Applied Mathematics*, 155(8):889 – 897, 2007.

[60] Min Kong, Peng Tian, and Yucheng Kao. A new ant colony optimization algorithm for the multidimensional knapsack problem. *Computers & Operations Research*, 35(8):2672 – 2683, 2008. Queues in Practice.

[61] George Kozanidis. Solving the linear multiple choice knapsack problem with two objectives: profit and equity. *Computational Optimization and Applications*, 43:261–294, 2009. 10.1007/s10589-007-9140-y.

[62] Daniel L. McShan Kyung-Wook Jee and Benedick A. Fraass. Lexicographic ordering: intuitive multicriteria optimization for imrt. *Physics in Medicine and Biology*, 52(7):1845–1861, 2007.

[63] Michael Lahanas, Eduard Schreibmann, Natasa Milickovic, and Dimos Baltas. Intensity modulated beam radiation therapy dose optimization with multiobjective evolutionary algorithms. 2632:70–70, 2003.

[64] Kiran Lakhotia, Mark Harman, and Phil McMinn. A multi-objective approach to search-based test data generation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1098–1105, New York, NY, USA, 2007. ACM.

[65] D. J. Laughhunn. Quadratic binary programming with application to capital-budgeting problems. *Operations Research*, 18(3):pp. 454–461, 1970.

[66] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932 – 942, 2006.

[67] Pekka J. Korhonen Julian Molina Lothar Thiele, Kaisa Miettinen. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17:411–436, 2009.

[68] D. S. Johnson M. R. Garey. *Computes and Intractibility: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[69] Silvano Martello, David Pisinger, and Paolo Toth. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*, 123(2):325 – 332, 2000.

[70] Jacinto Martn, Concha Bielza, and David Ros Insua. Approximating nondominated sets in continuous multiobjective optimization problems. *Naval Research Logistics (NRL)*, 52(5):469–480, 2005.

[71] Ellips Masehian and Davoud Sedighizadeh. Multi-objective robot motion planning using a particle swarm optimization model. *Journal of Zhejiang University - Science C*, 11:607–619, 2010. 10.1631/jzus.C0910525.

[72] Xavier Gandibleux Matthias Ehrgott. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22:425–460, 2000. 10.1007/s002910000046.

[73] George Mavrotas. Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455 – 465, 2009.

[74] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *Information Theory, IEEE Transactions on*, 24(5):525 – 530, sep 1978.

[75] H. Meunier, E.-G. Talbi, and P. Reininger. A multiobjective genetic algorithm for radio network optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 317 –324 vol.1, 2000.

[76] K. Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers, Dordrecht, 1999.

[77] A. Narayan and C. Patvardhan. A novel quantum evolutionary algorithm for quadratic knapsack problem. pages 1388 –1392, oct. 2009.

[78] Panos M. Pardalos, Yinyu Ye, and Chi-Geun Han. Algorithms for the solution of quadratic knapsack problems. *Linear Algebra and its Applications*, 152(0):69 – 91, 1991.

[79] E. Pires, J. Machado, and P. de Moura Oliveira. Robot trajectory planning using multi-objective genetic algorithm optimization. 3102:615–626, 2004.

[80] David Pisinger. The quadratic knapsack problem - a survey. *Discrete Applied Mathematics*, 155(5):623 – 648, 2007.

[81] Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2):509 – 533, 2008.

[82] Jakob Puchinger, Günther R. Raidl, and Ulrich Pferschy. The multidimensional knapsack problem: Structure and algorithms. *INFORMS J. on Computing*, 22(2):250–265, April 2010.

[83] D. Quadri, E. Soutif, and P. Tolla. Exact solution method to solve large scale integer quadratic multidimensional knapsack problems. *Journal of Combinatorial Optimization*, 17:157–167, 2009. 10.1007/s10878-007-9105-1.

[84] Mansini R., Speranza M.G., and Correspondence. A multidimensional knapsack model for asset-backed securitization. *Journal of the Operational Research Society*, 53(8), 2002.

[85] J. M. W. Rhys. A selection problem of shared fixed costs and network flows. *Management Science*, 17(3):pp. 200–207, 1970.

[86] Meir J. Rosenblatt and Zilla Sinuany-Stern. Generating the discrete efficient frontier to the capital budgeting problem. *Operations Research*, 37(3):pp. 384–394, 1989.

[87] S. Ruzika and M. M. Wiecek. Approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications*, 126:473–501, 2005. 10.1007/s10957-005-5494-4.

[88] P. Toth S. Martello. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, England, 1990.

[89] Serpil Sayin and Selcuk Karabati. Theory and methodology a bicriteria approach to the two-machine flow shop scheduling problem. *European Journal of Operational Research*, 113(2):435 – 449, 1999.

[90] Serpil Sayin and Panos Kouvelis. The multiobjective discrete optimization problem: A weighted min-max two-stage optimization approach and a bicriteria algorithm. *Management Science*, 51(10):pp. 1572–1581, 2005.

[91] Serpil Sayn. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming*, 87(3):543–560, 2000.

[92] Bala Shetty and R. Muthukrishnan. A parallel projection for the multicommodity network model. *The Journal of the Operational Research Society*, 41(9):pp. 837–842, 1990.

[93] Rajendra S. Solanki, Perry A. Appino, and Jared L. Cohon. Approximating the noninferior set in multiobjective linear programming problems. *European Journal of Operational Research*, 68(3):356 – 373, 1993.

[94] Rajendra S. Solanki and Jared L. Cohon. Approximating the noninferior set in linear biobjective programs using multiparametric decomposition. *European Journal of Operational Research*, 41(3):355 – 366, 1989.

[95] N. Srinivas and Kalyanmoy Deb. Muiltiobjective function optimization using nondominated sorting genetic algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1995.

[96] R. E. Steuer. Multiple criteria optimization: Theory, computation and application. *Wiley, New York*, 1986.

[97] Ralph E Steuer and Paul Na. Multiple criteria decision making combined with finance: A categorized bibliographic study. *European Journal of Operational Research*, 150(3):496 – 515, 2003. Financial Modelling.

[98] X. Sun, F. Wang, and D. Li. Exact algorithm for concave knapsack problems: Linear underestimation and partition method. *Journal of Global Optimization*, 33:15–30, 2005. 10.1007/s10898-005-1678-6.

[99] John Sylva and Alejandro Crema. A method for finding well-dispersed subsets of non-dominated vectors for multiple objective mixed integer linear programs. *European Journal of Operational Research*, 180(3):1011 – 1027, 2007.

[100] Heidi A. Taboada, Fatema Baheranwala, David W. Coit, and Naruemon Wattanapongsakorn. Practical solutions for multi-objective optimization: An application to system reliability design problems. *Reliability Engineering & System Safety*, 92(3):314 – 322, 2007.

[101] Fumiaki Taniguchi, Takeo Yamada, and Seiji Kataoka. A virtual pegging approach to the max - min optimization of the bi-criteria knapsack problem. *International Journal of Computer Mathematics*, 86(5):779–793, 2009.

[102] Gwo Hshiung Tzeng and Chien-Ho Chen. Multiobjective decision making for traffic assignment. *Engineering Management, IEEE Transactions on*, 40(2):180 –187, may 1993.

[103] E.L. Ulungu, J. Teghem, P.H. Fortemps, and D. Tuyttens. Mosa method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

[104] A. K. Ray V. Bhaskar, S.K. Gupta. Applications of multiobjective optimization in chemical engineering. *Reviews in Chemical Engineering*, 16(1):1–54, 2000.

[105] Y. Haimes V. Chankong. *Multiobjective Decision Making Theory and Methodology.* Elsevier Science, New York, 1983.

[106] Daniel C. Vanderster, Nikitas J. Dimopoulos, Rafael Parra-Hernandez, and Randall J. Sobie. Resource allocation on computational grids using a utility model and the knapsack problem. *Future Generation Computer Systems*, 25(1):35 – 50, 2009.

[107] M. Visee, J. Teghem, M. Pirlot, and E.L. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998. 10.1023/A:1008258310679.

[108] Haibo Wang, Gary Kochenberger, and Fred Glover. A computational study on the quadratic knapsack problem with multiple constraints. *Computers & Operations Research*, 39(1):3 – 11, 2012. Special Issue on Knapsack Problems and Applications.

[109] Charles A. Weber and John R. Current. A multiobjective approach to vendor selection. *European Journal of Operational Research*, 68(2):173 – 184, 1993.

[110] D. J. White. A bibliography on the applications of mathematical programming multiple-objective methods. *The Journal of the Operational Research Society*, 41(8):pp. 669–691, 1990.

[111] Christophe Wilbaut, Said Hanafi, and Said Salhi. A survey of effective heuristics and their application to a variety of knapsack problems. *IMA Journal of Management Mathematics*, 19(3):227–244, 2008.

[112] C. Witzgall. Mathematical methods of site selection for electronic message system (ems). *Technical Report, NBS Internal Report*, 1975.

[113] L. Lasdon Y. Haimes and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. 1:296–297, 1971.

[114] Cai Wen Zhang and Hoon Liong Ong. Solving the biobjective zero-one knapsack problem by an efficient lp-based heuristic. *European Journal of Operational Research*, 159(3):545 – 557, 2004.

[115] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257 –271, nov 1999.

[116] Eckart Zitzler and Johannes Bader. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19:45–76, 2011.

[117] Dexuan Zou, Liqun Gao, Steven Li, and Jianhua Wu. Solving 0-1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing*, 11(2):1556 – 1564, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence.