QUADRATIC BALANCED OPTIMIZATION PROBLEMS

by

Sara Taghipour

B.Sc. (Mathematics), University of Tehran, 2009

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
In the Department of Mathematics
Faculty of Science

SIMON FRASER UNIVERSITY
Fall 2011

APPROVAL

Name:	Sara Taghipour
Degree:	Master of Science
Title of Thesis:	Quadratic Balanced Optimization Problems
Examining Committee:	Dr. Zhaosang Lu Chair
	Dr. Abraham Punnen, Professor Senior Supervisor
	Dr. Tamon Stephen, Assistant Professor Supervisor
	Dr. Snezana Mitrovic-Minic, Adjunct Professor SFU Examiner
Date Approved:	December 6, 2011

Partial Copyright Licence

Abstract

We introduce the Quadratic Balanced Optimization Problem (QBOP) and study its complexity. QBOP is NP-hard even for very special cases such as Quadratic Balanced Knapsack Problem (QBKP) and Quadratic Balanced Assignment Problem (QBAP). Several general purpose algorithms are proposed and tested on the special cases of QBKP and QBAP. Polynomial solvable special cases are also identified.

To my parents for their endless love and support

"The essence of mathematics is its freedom."

George Cantor

Acknowledgments

My special thanks to my senior supervisor Dr. Abraham Punnen, for guiding me during my graduate studies and specifically this thesis. I am really grateful for his endless support, patience and useful advices. I should also thank Dr. Tamon Stephen and Dr. Zhaosong Lu for the lessons I learned from them in Operations Research. Also, I would like to express my gratitude to Dr. Randall Pyke for being perceptive and supportive as a TA coordinator and a good friend during my graduate studies. A sincere thank you to Dr. Matt DeVos. I was lucky to meet Matt as a brilliant teacher and an amazing friend.

I also appreciate kindness of Bishnu Bhattacharyya, Annie Zhang and my other friends for their support during this thesis.

Last but not least a great thank you from the bottom of my heart to my parents, Fereshteh Fareghi and Ebrahim Taghipour for their heart warming support during these years.

Contents

List of Tables

Chapter 1

Introduction

1.1 Combinatorial Optimization Problem

Let $E = \{1, ..., m\}$ be a finite set, F be a family of subsets of E, and $f : F \to R$. Then, a *Combinatorial Optimization Problem* (COP) is formulated as follows:

$$\begin{aligned} \text{Minimize} \quad & f(S) \\ \text{Subject to} \quad & S \in F. \end{aligned}$$

Here f is called the objective function and elements of F are called feasible solutions. Depending on the nature of f and the structure of E and F we obtain various special cases of COP which are discussed extensively in the literature. In most cases of well studied COP, a cost c_e is prescribed for each $e \in E$ and the objective function f often depends on c_e.

When $f(S) = \sum_{e \in S} c_e$, the resulting COP is called a *min-sum problem.* A closely related objective function is $f(S) = \dfrac{\sum_{e \in S} c_e}{|S|}$. Here, $f(S)$ indicates the average cost of an element in S. If $|S^1| = |S^2|$ for all $S^1, S^2 \in F$, the minsum problem and the average cost minimization problem are equivalent. When $f(S) = \sum_{e \in S} \{\max\{c_e : e \in S\} - c_e\}$ the COP reduces to the *Minimum Deviation Problem* [26] and when $f(S) = \sum_{e \in S} \left(c_e - \dfrac{\sum c_e}{|S|} \right)^2$ it becomes the *Minimum Variance Problem* [32]. All of the objective functions discussed above contains '$\sum$' in some form or other. There are objective functions for COP studied in the literature that depend on the maximum value of elements of a feasible solution. The resulting problems are called *Bottleneck Problems* [18]. Bottleneck problems are closely related to the objective

function we study in this thesis. Hence, first we give a brief literature review on bottleneck problems.

1.2 Linear Bottleneck Problem

In COP, if we choose $f(S) = \max_{e \in S} c_e$, we get the *Linear Bottleneck Problem* (LBP). Depending on the structure of F, special cases of LBP are known under various names. For instance, if E is the edge set of a graph G and F is the family of all spanning trees of G where c_e is the prescribed cost for edge $e \in E$, LBP reduces to the *Bottleneck Spanning Tree Problem* [11]. Camerini [11] presented an algorithm of complexity $O(m)$ to solve the bottleneck spanning tree problem. Punnen and Nair [49] proposed an algorithm to solve the bottleneck spanning tree problem with an additional linear constraint on a graph with n nodes and m edges. This algorithm runs in $O(m + n \log n)$ time. The best known algorithm for solving the bottleneck spanning tree problem on an undirected graph is of complexity of $O(m)$ [21] whereas, for a directed graph the most efficient known algorithm has complexity of $O(\min(m+n \log n, m \log^* n))$ where $\log^* n$ is the iterative logarithm of n [21]. Geetha and Nair [24] considered a generalized version of the spanning tree problem where edge costs are random variables and the objective is to find a spectrum of optimal spanning trees satisfying a certain chance constraint whose right-hand side is also treated as a decision variable. The most efficient method suggested in this paper, makes use of the efficient extreme points of the convex hull of the mappings of all the spanning trees in a bicriteria spanning tree problem. Ishi and Shiode [29] considered a stochastic version of the bottleneck spanning tree problem with edge costs considered as independent random variables, which is a generalization of stochastic bottleneck spanning tree problem already introduced in [28]. They presented a polynomial algorithm to find an optimal spanning tree. Hideki et al. [31] studied this problem where each cost attached to the edge in a given graph is represented as a fuzzy random variable. The goal in this case is to find an optimal spanning tree that maximizes the degree of possibility or necessity under some chance constraint. After transforming the problem into the deterministic form, they present a polynomial time algorithm.

When $E = \{1, ..., m\}$, $w_1, w_2, ..., w_m$, c are given numbers and $F = \{S : \sum_{j \in S} w_j \geq c, S \subseteq E\}$, LBP reduces to the *Bottleneck Knapsack Problem* [27]. Shioura and Shigeno [53] studied the tree center problems of finding a subtree minimizing the maximum distance from any vertex. They showed that the problem is related to the bottleneck knapsack

problem and presented a linear-time algorithm for the tree center problem by using the bottleneck knapsack problem solution. Further, they showed that the bottleneck knapsack problem can be solved in linear time [27].

Let P_n be the family of all permutations of $N = \{1, ..., n\}$. If we choose $F = \{(i, \pi(i)) : i \in N, \pi \in P_n\}$, $E = \{(i, j) \in N \times N\}$ and c_{ij} be the cost of $(i, j) \in E$. LBP reduces to the *Bottleneck Assignment Problem* [52]. Ravindran and Ramaswami [52] treated bottleneck assignment problem as a class of permutation problems and solved it by defining neighborhoods in the space of permutations and designed critical solutions in this space which results in global solutions. The bottleneck assignment problem can be also described in terms of perfect matching of a bipartite graph [48]. Derigs and Zimmermann [16] presented an augmenting path method to solve bottleneck assignment problem. Garbow and Tarjan [21] derived an algorithm with complexity of $O(m\sqrt{n \log n})$ and, Punnen and Nair [48] developed an algorithm with complexity of $O(n^{1.5}\sqrt{m})$ for solving the bottleneck assignment problem. Here the underlying bipartite graph has $O(n)$ nodes and m edges. Armstrong and Jin solved bottleneck assignment problem by applying strong spanning trees [5]. For an $n \times n$ linear bottleneck assignment Pferschy [43] obtained explicit upper and lower bounds where n is fixed and the distribution of the edges is uniform. Later, Pferschy [57] presented computational results on LBP. Yechiali [59] studied a stochastic version of the bottleneck assignment problem. Spivey [54] developed a method to find all of the asymptotic moments of a random bottleneck assignment problem where costs are chosen from a variety of continuous distributions. Burkard et al. [6] presented a comprehensive survey on the bottleneck assignment problem.

When $G = (V, E)$ is a complete graph, F is the family of all hamiltonian cycles in G and c_e is the cost of each $e \in E$, LBP reduces to the *Bottleneck Traveling Salesman Problem.* The bottleneck traveling salesman problem was introduced by Gilmore and Gomory [25]. Garfinkel and Gilbert [22] provided a branch-and-bound based exact algorithm to solve the problem. Experimental results with this algorithm on a randomly generated data set with size smaller than 100 vertices were also presented. Later, Carpento et al. [13] presented computational results on a branch-and-bound algorithm on problems of size up to 200 vertices. Philips et al. [44] showed that the bottleneck traveling salesman problem on a Halin graph can be solved in linear time. Parker and Rardin [42] provided a 2-approximation algorithm for the problem and showed this is the best possible performance bound a polynomial time algorithm can achieve unless P=NP. Ramakrishnan et al. [51] , and LaRusic and Punnen

[35] reported experimental results on large scale problems with heuristic algorithms. Other related works on bottleneck traveling salesman problem include [58, 30].

1.2.1 The Threshold Algorithm

Recall that c_e is the cost of the element $e \in E$. Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $c_e : e \in E$. We call $R(k)$ a *restricted set* and define it as $R(k) = \{e \in E : c_e > z_k\}$. Let $S_k = \{S \in F : S \cap R(k) = \emptyset\}$. Then, S_k includes all feasible solutions of LBP with objective function value not greater than z_k. Note that $S_1 \subseteq S_2 \subseteq ... \subseteq S_p$ and $S_p = F$

The following theorem is well known and is the basis of threshold algorithm for bottleneck problems. See for example [18, 50].

Theorem 1. *For $1 \leq k < p$, if k is the largest index such that $S_k = \emptyset$, then any $S \in S_{k+1}$ is an optimal solution to LBP whenever $F \neq \emptyset$. Also, if there exists $q : S_q = \emptyset$ then $S_k = \emptyset$ for all $k \leq q$*

Let us now discuss a general purpose algorithm for LBP, called *the threshold algorithm.* This is based on a duality theorem introduced by Edmonds and Fulkerson [18]. The algorithm solves a sequence of linear feasibility problems, (LFP(k)) described below:

"Given an index k , $1 \leq k \leq p$, is $S_k = \emptyset$?"

Note that the optimal objective function value of LBP is one of the values $z_1, z_2, ..., z_p$. The threshold algorithm [18] performs a binary search over these candidate objective values by using LFP(k). Depending on the answer of LFP(k) the size of the search space is reduced. The algorithm terminates when the search space becomes singleton in which case the optimal objective function value is identified. A formal description of the algorithm is given in Algorithm 1.1.

Since the algorithm reduces the size of the search space by half in each iteration, the algorithm terminates in $O(\log m)$ iterations. If $O(\phi(m))$ is the complexity of LFP(k) (i.e. complexity of testing if $S_k = \emptyset$ or not) then the complexity of the threshold algorithm is $O(\phi(m) \log m)$.

Algorithm 1.1: The Threshold Algorithm for LBP

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $c_e : e \in E$
$l = 1$; $u = p$;
while $u - l > 0$ **do**
  $k = \left\lfloor \frac{l+u}{2} \right\rfloor$;
  if $S_k = \emptyset$ **then**
    $l = k + 1$;
  else
    $u = k$
  end if
end while
Return z_l and any $S \in S_l$

1.3 Linear Balanced Optimization Problem

Let c_e be a prescribed cost associated with each element e of E. Define

$$f(S) = \max_{e \in S}\{c_e\} - \min_{e \in S}\{c_e\}$$

Then, COP reduces to the *Linear Balanced Optimization Problem* (LBOP). Consider the example of a school which wants to distribute students of each grade in a certain number of classes in a way that the students of the same class have the same level of GPA as much as possible. In other words, the gap between maximum GPA and minimum GPA in a class is minimized. Generally, balanced optimization model can be used in equitable distribution of resources. For further applications of BOP and related problems we refer to [39] and [17].

The *Linear Balanced Optimization Problem* (LBOP) was introduced by Martello et al. [39] and an application of the model in scheduling an international tour for a travel agency was pointed out. They suggested a general algorithm, called the double threshold algorithm to solve the problem. They also studied the special case of balanced assignment problem. Duin and Volgenant[17] showed that LBOP can be solved as a sequence of bottleneck problems and obtained an improved algorithm. In fact the algorithm of [17] is more general and could solve other related combinatorial problems. Punnen and Nair [49] studied LBOP with an additional linear constraint and Punnen and Aneja [47] considered the lexicographic version of the problem. Ahuja studied the balanced linear programming problem and solved it by parametric simplex method [3]. A special case of balanced linear programming problem

was studied by Cappanera and Scutella [12]. Tigan et al. [56] investigated the monotone balanced optimization problem. Other special cases of LBOP include *Balanced Spanning Tree Problem*, *Balanced Knapsack Problem*, and *Balanced Traveling Salesman Problem*. Camerini et al. [10] considered balanced spanning tree problem and proposed an $O(m^2)$ algorithm where m is the number of edges in the underlying graph. Later, Galil and Schieber [20] improved the algorithm presented in [10]. This algorithm is of complexity $O(m \log n)$ where m is the number of edges and n is the number of vertices in the graph under consideration. LaRusic and Punnen [35] studied balanced traveling salesman problem, which is NP-hard, and provided efficient heuristic algorithms to solve the problem. The nozzle guide vane assembly problem can be formulated as a balanced TSP [35]. LBOP in context of *Resource Allocation Problem* was studied by Zeitlin [60]. Nemoto [41] presented an algorithm to minimize the gap between maximum and minimum weights in an ideal of size k in a partially ordered set. Katoh and Iwano [33] and Dai et al. [14] developed efficient algorithms for the set of feasible solutions forms all cuts in a graph G. Epstein found balanced cut which minimizes the maximum range of edge lengths in time $O(m + n^2 \log n)$ [19].

1.3.1 The Feasibility Problem and The Double Threshold Algorithm

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of c_e for $e \in E$ and S^0 be an optimal solution to LBOP. For any two real numbers γ and δ, where $\gamma < \delta$ consider $R_{(\gamma,\delta)} = \{e : c_e < \gamma \text{ or } c_e > \delta\}$ and $F_{(\gamma,\delta)} = \{S : S \cap R_{(\gamma,\delta)} = \emptyset\}$. Let the indices l and u be chosen such that $z_u = \max\{c_e : e \in S^0\}$ and $z_l = \min\{c_e : e \in S^0\}$ where S^0 is the optimal solution to BOP. Hence, whenever $\delta - \gamma < z_u - z_l$ and $S^0 \in F_{(z_l,z_u)}$, we have $F_{(\gamma,\beta)} = \emptyset$
The feasibility problem considered here is closely related to LFP(k) considered for bottleneck problem. Here, we want to test if $F_{(\gamma,\beta)} = \emptyset$ or not. We represent feasibility problem as FP(γ, δ) which is a 'yes' or 'no' question. The algorithm initially selects lower threshold = upper threshold = z_1. In the general step, if the answer to feasibility question is 'No', then upper threshold is increased and if the answer is 'Yes', the lower threshold is increased. The algorithm outputs the best solution generated in the search process. For details of this algorithm we refer to [39]. Algorithm 1.2 gives the formal description of the double threshold algorithm.

Theorem 2 ([39]). *The double threshold algorithm solves the linear balanced optimization*

Algorithm 1.2: The Double Threshold Algorithm (DT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $c_e : e \in E$
$l = 1$; $u = 1$
Opt-Sol= $\emptyset$; Obj-Val= ∞
while $l \leq p$ and $u \leq p$ **do**
  if $F_{(z_l,z_u)} \neq \emptyset$ **then**
    Choose $S \in F_{(z_l,z_u)}$
    $z_r = \min\{c_e : e \in S \times S\}$
    $z_t = \max\{c_e : e \in S \times S\}$
    if $z_t - z_r = 0$ **then**
      return S
    end if
    if $z_t - z_r <$ Obj-Val **then**
      Obj-Val= $z_t - z_r$
      Opt-Sol= S
    end if
    $l = r + 1$
  else
    $u = u + 1$
  end if
end while
Return Opt-Sol and Obj-Val

problem in $O(m\phi(m))$ time, where $O(\phi(m))$ is the complexity of the feasibility problem FP(γ, δ) .

The primary goal of this thesis is to study the quadratic version of Balanced Optimization Problems which is defined as follows:

$$QBOP: \quad \text{Minimize} \quad \{\max_{(i,j)\in S\times S} q_{ij} - \min_{(i,j)\in S\times S} q_{ij}\}$$
$$\text{Subject to} \quad S \in F.$$

where $E = \{1, 2, ..., m\}$ and q_{ij} is the prescribed cost associated with the ordered pair $(i, j) \in E \times E$.

Subsequent chapters include our contribution to QBOP. We showed that

(1) Developed 4 general algorithms to solve QBOP.

(2) The algorithms developed can be used as exact algorithms or heuristics, depending on the way an associated feasibility problem is solved.

(3) Some polynomially solvable special cases of QBOP are identified.

(4) Special cases of QBOP when feasible solutions satisfy a knapsack constraint and when feasible solutions are perfect matchings of a bipartite graph are investigated.

(5) Experimental results with the algorithms developed are also included.

The thesis organized as follows: In Chapter 2, (QBOP) was introduced. Four exact algorithms including Double Threshold Algorithm (DT), Improved Double Threshold Algorithm (IDT), Modified Double Threshold Algorithm(MDT) and Iterative Bottleneck in addition to two polynomially solvable cases of QBOP are presented.

Chapter 3 deals with Quadratic Balanced Knapsack Problem(QBKP) which is an instance of QBOP. Polynomially solvable cases of QBKP are investigated. The algorithms introduced in Chapter 2 were applied to QBKP. In addition to these exact algorithms, a heuristic method was developed to solve this problem. The computational results of the algorithms and the heuristic method were provided in 4 tables.

In Chapter 4, the Quadratic Balanced Assignment Problem(QBAP) was introduced. We considered this problem on both complete and sparse bipartite graphs. All four exact algorithms in Chapter 2 were used for this problem on a sparse bipartite graph case. Computational results of the experiments on these algorithms are provided in 4 tables. Further, polynomially solvable cases of QBAP are presented.

Conclusion remarks are given in Chapter 5.

Chapter 2

Quadratic Balanced Optimization Problem

Recall that the *Quadratic Balanced Optimization Problem*(QBOP) is to:

$$\begin{array}{ll} \text{Minimize} & \{\max_{(i,j)\in S\times S} q_{ij} - \min_{(i,j)\in S\times S} q_{ij}\} \\ \text{Subject to} & S \in F. \end{array}$$

where $E = \{1, 2, ..., m\}$ and q_{ij} is the prescribed cost associated with the ordered pair $(i, j) \in E \times E$. QBOP is yet another example of a COP.

Depending on the structure of E and F, we get several special cases of QBOP. For instance, if E is the edge set of a graph G and F is the family of all spanning trees of G, we have the special case of *Quadratic Balanced Spanning Tree Problem*. Let $a_1, a_2, ..., a_n$, c be given numbers and $F = \{S : \sum_{j\in S} a_j \geq c, S \subseteq E\}$. Then, QBOP reduces to the *Quadratic Balanced Knapsack Problem*. Let P_n be the family of all permutations of $N = \{1, ..., n\}$ and $F = \{(i, \pi(i)) : i \in N : \pi \in P_n\}$ and $E = \{(i, j) \in N \times N\}$ we get the *Quadratic Balanced Assignment Problem*. To the best of our knowledge, QBOP is not studied in literature. A closely related problem, called quadratic bottleneck problem has been studied by various researchers [55, 50]. Let us first discuss the quadratic bottleneck problem.

2.1 Quadratic Bottleneck Problem

QBOP is closely related to the quadratic bottleneck problems [50] given by two types.

A quadratic bottleneck problem of type 1 (QBP1) is defined as

$$QBP1: \quad \begin{array}{ll} \text{Minimize} & \max\{q_{ij} : (i,j) \in S \times S\} \\ \text{Subject to} & S \in F \end{array}$$

and the quadratic bottleneck problem of type 2 (QBP2) is defined as:

$$QBP2: \quad \begin{array}{ll} \text{Maximize} & \min\{q_{ij} : (i,j) \in S \times S\} \\ \text{Subject to} & S \in F \end{array}$$

The problem QBP1 was investigated by Punnen and Zhang [50] and they proposed a general purpose algorithm to solve the problem. QBP2 can be modified to QBP1 by converting its objective function to $\min\{\max -q_{ij} : (i,j) \in S \times S\}$. Similarly QBP2 can be formulated as QBP1. In this sense both QBP1 and QBP2 are equivalent. QBP1 is known to be NP-hard [50]. Since it can be reduced to QBP2, QBP2 is also NP-hard and all the algorithms for QBP1 in [50] can be modified to obtain algorithms for QBP2, without reducing QBP2 to QBP1.

The Quadratic bottleneck assignment problem with feasible solutions as perfect matchings of a bipartite graph was introduced by Steinberg [55] to solve a backboard wiring problem. Burkard and Fincke [7] studied asymptotic properties of the quadratic bottleneck assignment problems. Kellerer and Wirsching [34] used the quadratic bottleneck assignment model to solve bandwidth minimization problem of matrices and graphs. Zhang et al. [61] studied a special case of quadratic bottleneck spanning tree problem and presented some efficient algorithms to solve it. Punnen and Zhang [50] studied the general quadratic bottleneck problem. They presented a weak duality theorem and general purpose algorithms to solve it. The algorithm was illustrated using the special case of quadratic spanning trees. Zhang and Punnen [62] studied the quadratic bottleneck knapsack problem.

Note that if $q_{ij} \geq 0$ for all $(i,j) \in E \times E$ and $q_{ii} = 0$ for all $i \in E$ then QBOP reduces to QBP1. However, in general the QBOP is different from QBP1 and QBP2 and specialized algorithms are required to solve QBOP.

To motivate the study of the QBOP model, consider an example of a construction company that wants to build a residential complex in a certain distance from facilities. The distance between building i and facility j is given by d_{ij}. Their goal is to build these

buildings of the complex in a way that all of the buildings have almost the same distance from the facilities and the gap between the farthest and the closest distance from facilities is minimized.

2.2 The Double Threshold Algorithm for QBOP

Recall that the linear balanced optimization problem can be solved by the double threshold algorithm [39] as a sequence of feasibility problems. We use the same philosophy for solving QBOP. However, the nature of the feasibility problem is quite different for QBOP. Punnen and Zhang [50] considered the quadratic feasibility problem and used it to solve the quadratic bottleneck problems. We use a variation of this quadratic feasibility problem to develop our double threshold algorithm.

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct q_{ij} values. For any real numbers α and β with $\alpha \leq \beta$ define: $E(\alpha, \beta) = \{(i,j) \in E \times E : q_{ij} > \alpha \text{ or } q_{ij} < \beta\}$ and $F(\alpha, \beta) = \{S : (S \times S) \cap E(\alpha, \beta) = \emptyset\}$. Let S^* be an optimal solution to QBOP and let $w_1 = \max\{q_{ij} : (i,j) \in S^* \times S^*\}$ and $w_2 = \min\{q_{ij} : (i,j) \in S^* \times S^*\}$. Then, $F(\alpha, \beta) = \emptyset$ for any α and β such that $\beta - \alpha < w_1 - w_2$ and $S^* \in F(w_2, w_1)$. Furthermore, if $F(\alpha, \beta) = \emptyset$ then $F(\gamma, \delta) = \emptyset$ for $\alpha \leq \gamma \leq \delta \leq \beta$.

The double threshold algorithm primarily uses a feasibility oracle that tests if $F(z_l, z_u) \neq \emptyset$ or not. For appropriate choices of l and u, if the answer is 'No' the upper threshold z_u is increased to z_{u+1}. If the feasibility oracle answers 'Yes' then we have a feasible solution S with QBOP objective function value $\leq z_u - z_l$. Choose r and t such that $z_r = \min\{q_{ij} : (i,j) \in S \times S\}$ and $z_t = \max\{q_{ij} : (i,j) \in S \times S\}$. The lower threshold is updated to z_{r+1} and the best solution and the best objective function value so far is updated. Note that although binary search takes less time than sequential search, since the objective function of QBOP is not monotonic, we can not use binary search for finding the optimal solution of QBOP.

A formal description of the double threshold algorithm is summarized in Algorithm 2.1. Note that $p = O(m^2)$. Let $O(\phi(m))$ be the complexity of the feasibility oracle i.e. to determine if $F(\alpha, \beta) \neq \emptyset$ or not and if it is nonempty the oracle retrieves a solution $S \in F(\alpha, \beta)$.

Theorem 3. *The double threshold algorithm computes an optimal solution to QBOP in* $O(m^2\phi(m))$ *time.*

Algorithm 2.1: The Double Threshold Algorithm (DT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$
$l = 1$; $u = 1$
Opt_Sol $= \emptyset$; Obj_Val $= \infty$
while $l \leq p$ and $u \leq p$ **do**
  if $F_{(z_l, z_u)} \neq \emptyset$ **then**
    Choose $S \in F_{(z_l, z_u)}$
    $z_r = \min\{q_{ij} : (i, j) \in S \times S\}$
    $z_t = \max\{q_{ij} : (i, j) \in S \times S\}$
    if $z_t - z_r = 0$ **then**
      return S
    end if
    if $z_t - z_r <$ Obj-Val **then**
      Obj_Val$= z_t - z_r$
      Opt_Sol $= S$
    end if
    $l = r + 1$
  else
    $u = u + 1$
  end if
end while
Return Opt_Sol and Obj_Val

Proof. The validity of the algorithm follows from discussions presented earlier. In each iteration, the algorithm either increases the lower threshold or increases the upper threshold. Thus, the total number of iterations is bounded above by $2p$. Since the dominating complexity in each iteration is $O(\phi(m))$ and $p = O(m^2)$ the result follows. □

2.2.1 Feasibility test

The crucial step in Algorithm 2.1 is the feasibility oracle to test if $F(\alpha, \beta) = \emptyset$ or not and if $F(\alpha, \beta) \neq \emptyset$ produce an $S \in F(\alpha, \beta)$. This can be achieved in several ways by appropriate modifications of the results given in [50] for the case of quadratic bottleneck problems.

Let $D = (d_{ij})$ be an $m \times m$ matrix given by

$$d_{ij} = \begin{cases} M & \text{if } (i,j) \in E(\alpha, \beta) \\ 0 & \text{otherwise.} \end{cases}$$

Consider the *Quadratic Sum Problem*(QSP):

$$\begin{aligned} \text{Minimize} \quad & \sum_{(e,f) \in S \times S} d_{ef} \\ \text{Subject to} \quad & S \in F. \end{aligned}$$

Then, $F(\alpha, \beta) \neq \emptyset$ if and only if the optimal objective function value of QSP is zero. Further, if $F(\alpha, \beta) \neq \emptyset$ then an optimal solution to QSP belongs to $F(\alpha, \beta)$.

Another way to test if $F(\alpha, \beta) = \emptyset$ or not is by using a linear combinatorial optimization problem with conflict pairs (LCOPC) [61]:

The feasibility check in this case is simply to check if there is an $S \in F$ that satisfies the conflict pair constraints [61] where the conflict set is $S(\alpha, \beta) = \{\{i, j\} : (i, j) \in E(\alpha, \beta)\}$.

Let $Conv(F)$ be the convex hull of incidence vectors of $S \in F$. Note that an n-vector $(x_1, ..., x_m)$ is an incidence vector of solution S if $x_j = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise.} \end{cases}$

Then, the feasible solutions of the LCOPC with conflict set $S(\alpha, \beta)$ can be written as [61],

$$x \in Conv(F)$$
$$x_i + x_j \leq 1 \quad if\{i,j\} \in E_{(\alpha,\beta)}$$

Depending on the structure of F, the feasibility version of this LCOPC can be solved by different algorithms. We will discuss this in more details in the next two chapters. For details on LCOPC and its various special cases, we refer to[61] and [15].

2.2.2 The Improved Double Threshold Algorithm for QBOP

The Improved Double Threshold (IDT) algorithm helps to restrict the search interval for optimal solution. This algorithm first solves QBP1 and QBP2. For any feasible solution S, let $Z_1(S) = \max\{q_{ij} : (i,j) \in S \times S\}$, $Z_2(S) = \min\{q_{ij} : (i,j) \in S \times S\}$ and $Z(S) = Z_1(S) - Z_2(S)$. Let S^1 be the optimal solution to QBP1 , U^* be its optimal objective function value and S^2 be the optimal solution to QBP2 and L^* be its optimal objective function value.

Theorem 4. *For any feasible solution S, $Z_2(S) \leq L^*$. Further, for any solution with $Z_1(S) = U^*$, if $Z_2(S) < Z_2(S^1)$, then $Z(S) > Z(S^1)$*

Proof. The inequality $Z_2(S) \leq L^*$ follows from the definition of L^*. Also, for any feasible solution S, $Z_1(S) \geq Z_1(S^1)$. Thus, if $Z_2(S) < Z_2(S^1)$ we have $Z(S) = Z_1(S) - Z_2(S) > Z_1(S^1) - Z_2(S^1) = Z(S^1)$ and the proof is complete. □

Using Theorem 4, we can reset the starting lower threshold value as $Z_2(S^1)$ and the lower threshold is not required to increase beyond L^*. This could reduce the search interval considerably. The double threshold algorithm incorporating this enhancement is called the improved double threshold algorithm. A formal description of the algorithm is presented in Algorithm 2.2.

2.2.3 The Modified Double Threshold Algorithm for QBOP

Let us now discuss how to improve the average performance of the double threshold algorithm and the improved double threshold algorithm. The improvement is achieved using
1) sufficient condition for optimality and

Algorithm 2.2: The Improved Double Threshold Algorithm (IDT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$
Let Opt_Sol be the optimal solution to QBOP and Obj_Val the optimal objective function value to QBOP
Let S^1 be the optimal solution to QBP1 and U^* be the optimal objective function value of that, l be such that $z_l = min\{q_{ij} : (i, j) \in S^1 \times S^1\}$. We set l and u be such that $(z_u \leftarrow U^*)$
Let L^* be such that z_{L^*} be the optimal objective function value of QBP2
Opt_Sol $\leftarrow S^1$
Obj_Val $= z_u - z_l$
$l \leftarrow l + 1$
while $(l \leq u)$ and $(l \leq L^*)$ and $(u \leq p)$ **do**
  if $F_{(z_l,z_u)} \neq \emptyset$ **then**
    Choose $S \in F_{(z_l,z_u)}$
    $z_r = \min\{q_{ij} : (i, j) \in S \times S\}$
    $z_t = \max\{q_{ij} : (i, j) \in S \times S\}$
    if $z_t - z_r <$ Obj_Val **then**
      Obj-Val$= z_t - z_r$ Opt_Sol $= S$
    end if
    $l = r + 1$
  else
    $u \leftarrow u + 1$
  end if
end while
Return Opt_Sol and Obj_Val

2) a sufficient condition that allows increments of upper and lower threshold values by larger amounts.

The conditions however does not seem to affect the worst case complexity of the algorithm.

Let $F^* = \{S_1, S_2, ..., S_p\}$ be the set of all solutions generated by the double threshold algorithm while completing the sequential search. We assume that S_i is generated before S_j for $i < j$. For $1 \leq t \leq p$ let, $\alpha_t = max\{q_{ij} : (i,j) \in S_t \times S_t\}$ and $\beta_t = min\{q_{ij} : (i,j) \in S_t \times S_t\}$. Thus, $\alpha_1 < \alpha_2 < ... < \alpha_p$ and $\beta_1 < \beta_2 < ... < \beta_p$. For $k = 1, ..., p$. Choose $t(k)$ such that $\alpha_{t(k)} - \beta_{t(k)} = min\{\alpha_i - \beta_i : 1 \leq i \leq k\}$.

Let $F_k^* = \{S^1, S^2, ..., S^k\}$. Note that $S^{t(p)}$ is an optimal solution to QBOP since $\alpha_{t(p)} - \beta_{t(p)} = min\{\alpha_i - \beta_i : 1 \leq i \leq p\}$. Now, let $F^0 \subset F$ such that $F^0 = \{S_i \in F^* : \alpha_i - \beta_i = \alpha_{t(p)} - \beta_{t(p)}\}$. Thus, any $S \in F^0$ is an optimal solution to QBOP. Let β be a real number such that $\beta \geq \max\{\beta_i : S_i \in F^0\}$. We choose $\beta = \min\{q_{ij} : (i,j) \in S^2 \times S^2\}$ to obtain the best possible bound.

Theorem 5. *For any $1 \leq k \leq p$ if $\alpha_{t(k)} - \beta_{t(k)} + \beta \leq \alpha_k$ then $S_{t(k)} \in F^0$.*

Proof. Suppose $S_{t(k)} \notin F^0$. Then there is an $S_i \in F^0$ and $i > k$ and, $\alpha_i - \beta_i < \alpha_{t(k)} - \beta_{t(k)}$. Then $\alpha_i < \beta_i + \alpha_{t(k)} - \beta_{t(k)} \leq \beta + \alpha_{t(k)} - \beta_{t(k)} \leq \alpha_k$. This implies $i \leq k$ which is a contradiction. □

Theorem 5 provides a termination criterion that could reduce the number of iterations for the double threshold algorithm.

Theorem 6. *If $S_{t(k)}$ does not belong to F^0 then there is an $S_q \in F^0$ for $q > k$ such that $\beta_q > \alpha_k - \alpha_{t(k)} + \beta_{t(k)}$.*

Proof. On contrary, suppose $\beta_q \leq \alpha_k - \alpha_{t(k)} + \beta_{t(k)} \leq \alpha_q - \alpha_{t(k)} + \beta_{t(k)}$ then $\alpha_q - \beta_q \geq \alpha_{t(k)} + \beta_{t(k)}$ then $S^{t(k)} \in F^0$ which is a contradiction. □

Theorem 6 could be used to update the lower threshold by larger quantity whenever the condition holds.

Results similar to Theorem 5 and 6 were used by Ahuja [2], Martins[40], Punnen [45] and Punnen and Nair [49] in solving other combinatorial optimization problems. Incorporating the conditions presented in Theorems 5 and 6 into the improved double threshold algorithm, we get the modified double threshold algorithm to solve QBOP. A formal description of the modified double threshold algorithm is given in Algorithm 2.3.

Algorithm 2.3: The Modified Double Threshold Algorithm (MDT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$
Let Opt_Sol be the optimal solution to QBOP and Obj_Val the optimal objective function value to QBOP
Let S^1 be the optimal solution to QBP1 and U^* be the optimal objective function value of that, l be such that $z_l = min\{q_{ij} : (i, j) \in S^1 \times S^1\}$. We set l and u be such that $(z_u \leftarrow U^*)$ and L^* be such that z_{L^*} is the optimal objective function value of QBP2
Opt_Sol $\leftarrow S^1$
Obj_Val $= z_u - z_l$
$l \leftarrow l + 1$
while $(l \leq u)$ and $(l \leq L^*)$ and $(u \leq p)$ **do**
  if $F_{(z_l,z_u)} \neq \emptyset$ **then**
    Choose $S \in F_{(z_l,z_u)}$
    $z_r = \min\{q_{ij} : (i, j) \in S \times S\}$
    $z_t = \max\{q_{ij} : (i, j) \in S \times S\}$
    if $z_t - z_r <$ Obj-Val **then**
      Obj_Val $= z_t - z_r$
      Opt_Sol $= S$
    end if
    if Obj_Val $+L^* \leq z_r$ **then**
      return Obj_Val and Opt_Sol
    end if
    $l = r + 1$
    choose smallest $q \geq l$ such that $z_q \geq z_u -$Obj_Val
    $l \leftarrow q$
  else
    $u \leftarrow u + 1$
  end if
end while
Return Opt_Sol and Obj_Val

2.2.4 The Iterative bottleneck algorithm for QBOP

Let us now discuss two additional algorithms to solve QBOP. In both cases, we solve a sequence of quadratic bottleneck problems of the type QBP1 or QBP2 instead of a quadratic feasibility problem. The worst case complexities of these algorithms, in general, are higher than that of the DT algorithm and its variations, the average performance is expected to be better. Recall that

$$\begin{aligned} QBP1: \quad & \text{Minimize} \quad \max\{q_{ij} : (i,j) \in S \times S\} \\ & \text{Subject to} \quad S \in F \end{aligned}$$

$$\begin{aligned} QBP2: \quad & \text{Maximize} \quad \min\{q_{ij} : (i,j) \in S \times S\} \\ & \text{Subject to} \quad S \in F \end{aligned}$$

Thus, QBP1 minimizes $Z_1(Q,S)$ while QBP2 maximizes $Z_2(Q,S)$. To develop our first iterative bottleneck algorithm, we consider a generalization of QBOP where lower and upper threshold restrictions are imposed on feasible solutions. Let α and β be real numbers such that $\alpha \leq \beta$. Consider the problem

$$\begin{aligned} \text{QBOP(Q,}\alpha,\beta\text{):= Minimize} \quad & Z_1(Q,S) - Z_2(Q,S) \\ \text{Subject to} \quad & S \in F \\ & Z_2(Q,S) \geq \alpha \\ & Z_1(Q,S) \leq \beta \end{aligned}$$

Consider the cost matrix Q' defined by

$$q'_{ij} = \begin{cases} M & \text{if } q_{ij} < \alpha \text{ or } q_{ij} > \beta \\ q_{ij} & \text{otherwise.} \end{cases}$$

where M is a large number.

Theorem 7. *Let S^0 be an optimal solution to QBP1 with cost matrix Q' and q be the index such that $z_q = Z_2(Q,S^0)$.*
(i) If $Z_1(Q,S^0) = M$ then QBOP(Q,α,β) is infeasible.
(ii) If $Z_1(Q,S^0) < M$ and $Z_1(Q,S^0) = Z_2(Q,S^0)$ then S^0 is an optimal solution to QBOP(Q,α,β).

(iii) If conditions (i) and (ii) above are not satisfied, then either S^0 is an optimal solution to QBOP(Q,α, β) or there exists an optimal solution to QBOP(Q,α, β) which is optimal to QBOP(Q,γ, β) where $\gamma = z_{q+1}$.

Proof. The proof of (i) and (ii) are straightforward. Let us now prove (iii).

Let $F = \{S \in F(\alpha, \beta) : Z_2(Q, S) \leq Z_2(Q, S^0)\}$. By definition of F

$$Z_2(Q, S^0) \geq Z_2(Q, S) \text{ for all } S \in F. \tag{2.1}$$

By optimality of S^0 to QBP1 with cost matrix Q' and condition (i) of the theorem is not satisfied, we have

$$Z_1(Q, S^0) \leq Z_1(Q, S) \text{ for all } S \in F. \tag{2.2}$$

Multiply inequality (2.1) by -1 and adding to inequality (2.2) we have QBOP$(Q, S^0) \leq$ QBOP(Q, S) for all $S \in F$. Thus either S^0 is an optimal solution to QBOP(Q,α, β) or there exists an optimal solution S to QBOP(Q,α, β) satisfying $Z_2(Q, S) > Z_2(Q, S^0)$ and the result follows. □

In view of Theorem 7, we can solve QBOP as a sequence of QBP1 problems. In each iteration, the algorithm maintains an upper threshold β and a lower threshold α and construct a modified cost matrix Q' which depends on the values of α and β. Then using an optimal solution to QBP1 with cost matrix Q', the lower threshold is updated until infeasibility with respect to the threshold values is identified or optimality of an intermediate solution is identified using condition (ii) of Theorem 7. The algorithm compares the solutions generated by the QBP1 solver and outputs the overall best solution with respect to the QBOP objective function. The resulting algorithm is called the *type 1 iterative bottleneck algorithm* (IB1-algorithm) and its formal description is given in Algorithm 2.4.

Recall that QBP2 can be reformulated as QBP1 or the algorithms for QBP1 [50] can be modified to solve QBP2 directly. Consider the cost matrix $\tilde{Q}$ defined by

$$\tilde{q}_{ij} = \begin{cases} -M & \text{if } q_{ij} > \beta \text{ or } q_{ij} < \alpha \\ q_{ij} & \text{otherwise.} \end{cases}$$

where M is a large number.

Theorem 8. *Let S^0 be an optimal solution to QBP2 with cost matrix $\tilde{Q}$ and r be the index such that $z_r = Z_1(Q, S^0)$.*
(i) If $Z_2(\tilde{Q}, S^0) = -M$ then QBOP(Q,α,β) is infeasible.
(ii) If $Z_2(\tilde{Q}, S^0) > -M$ and $Z_1(Q, S^0) = Z_2(Q, S^0)$ then S^0 is an optimal solution to QBOP(Q,α,β).
(iii) If conditions (i) and (ii) are not satisfied, then either S^0 is an optimal solution to QBOP(Q,α,β) or there exists an optimal solution to QBOP(Q,α,β) which is also optimal to QBOP(Q,α,γ) where $\gamma = z_{r+1}$.

The proof of this theorem can be constructed by appropriate modifications in the proof of Theorem 7 and hence is omitted. In view of Theorem 8, we can solve QBOP as a sequence of QBP2 problems. In each iteration, the algorithm maintains a lower threshold α and an upper threshold β and construct a modified cost matrix $\tilde{Q}$. Using an optimal solution to QBP2 with cost matrix $\tilde{Q}$, the upper threshold is updated and the process is continued until infeasibility with respect to the threshold values is identified. The algorithm compares the solutions generated by the QBP2 solver in each iteration and outputs the overall best solution. The resulting algorithm is called the *type 2 iterative bottleneck algorithm* (IB2-algorithm) and its formal description is given in Algorithm 2.5.

Algorithms 2.4 and 2.5 can be viewed as extension of an algorithm by Duin and Volgenant [17] for a generalization of the LBOP.

2.3 Polynomially solvable cases

Let us now consider some special cases of QBOP that can be solved in polynomial time.
Case 1: Sum Case
We first consider the *decomposable cost matrix* where $q_{ij} = a_i + b_j$ for given $a_i \geq 0$ and $b_j \geq 0$ where $i, j \in E$. Note that

$$\begin{aligned} Z(Q,S) &= \max\{q_{ij} : (i,j) \in S \times S\} - \min\{q_{ij} : (i,j) \in S \times S\} \\ &= \max\{a_i + b_j : (i,j) \in S \times S\} - \min\{a_i + b_j : (i,j) \in S \times S\} \\ &= \max\{a_i : i \in S\} + \max\{b_i : i \in S\} - \min\{a_i : i \in S\} - \min\{b_i : i \in S\} \quad (2.3) \\ &= \max\{a_i : i \in S\} + \max\{-a_i : i \in S\} + \max\{b_i : i \in S\} - \min\{b_i : i \in S\} \quad (2.4) \end{aligned}$$

Algorithm 2.4: The type1 Iterative Bottleneck Algorithm(IB1)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$

Let Q be the cost matrix , Opt_Sol be the optimal solution of QBOP and Obj_Val be the objective function value of QBOP in each iteration.

$Q' \leftarrow Q$; $Obj\_Val \leftarrow \infty$; $Opt\_Sol \leftarrow \emptyset$, $M \leftarrow 1 + z_p$; $Z_0 = z_p$;

while $(Obj\_Val \neq 0)$ and $Z_0 \neq M$ **do**

Solve QBP1 with cost matrix Q'. Let S be the resulting solution.

$Z_0 \leftarrow \max\{q'_{ij} : (i, j) \in S \times S\}$

if $Z_0 < M$ **then**

$Z_1 \leftarrow \max\{q_{ij} : (i, j) \in S \times S\}$

$Z_2 \leftarrow \min\{q_{ij} : (i, j) \in S \times S\}$

if $Z_1 - Z_2 < Obj - Val$ **then**

$Opt\_Sol \leftarrow S$

$Obj\_Val \leftarrow Z_1 - Z_2$

end if

Modify costs:

$$q'_{ij} = \begin{cases} q_{ij} & \text{if } Z_2 < q_{ij} < z_p \\ M & \text{Otherwise} \end{cases}$$

end if

end while

Return Opt_Sol and Obj_Val

Algorithm 2.5: The type2 Iterative Bottleneck Algorithm(IB2)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$

Let Q be the cost matrix , Opt_Sol be the optimal solution of QBOP and Obj_Val be the objective function value of QBOP in each iteration.

$\tilde{Q} \leftarrow Q$; $Obj\_Val \leftarrow \infty$; $Opt\_Sol \leftarrow \emptyset$, $M \leftarrow 1 + z_p$; $Z_0 = z_1$;

while $(Obj\_Val \neq 0)$ and $Z_0 \neq -M$ **do**

Solve QBP2 with cost matrix $\tilde{Q}$. Let S be the resulting solution.

$Z_0 \leftarrow \min\{\tilde{q}_{ij} : (i, j) \in S \times S\}$

if $Z_0 > -M$ **then**

$Z_1 \leftarrow \max\{q_{ij} : (i, j) \in S \times S\}$

$Z_2 \leftarrow \min\{q_{ij} : (i, j) \in S \times S\}$

if $Z_1 - Z_2 < Obj - Val$ **then**

$Opt\_Sol \leftarrow S$

$Obj\_Val \leftarrow Z_1 - Z_2$

end if

Modify costs:

$$\tilde{q}_{ij} = \begin{cases} q_{ij} & \text{if } z_1 < q_{ij} < Z_1 \\ -M & \text{Otherwise} \end{cases}$$

end if

end while

Return Opt_Sol and Obj_Val

Let w_i be a prescribed weight of element $i \in E$ and $g : F \to \Re$. Duin and Volgenant [17] showed that combinatorial optimization problems of the type

COP(g): Minimize $\max\{w_i : i \in S\} + g(S)$
Subject to
$S \in F$.

can be solved in $O(m\zeta(m))$ where $\zeta(m)$ is the complexity of minimizing $g(S)$ over F. Note that

$$Z(Q, S) = \max\{a_i : i \in S\} + g(S) \tag{2.5}$$

where $g(S) = \max\{-a_i : i \in S\} + g_1(S)$ and $g_1(S) = \max\{b_i : i \in S\} - \min\{b_i : i \in S\}$. But minimizing $g_1(S)$ over F is precisely the LBOP [39]. Thus recursively applying the results of Duin and Volgenant [17], BCOP with a decomposable cost matrix can be solved in $O(m^2\eta(m))$ time where $\eta(m)$ is the complexity of an LBOP.

When $a_i = b_i$, then $Z(Q, S) = 2[\max\{a_i : i \in S\} - \min\{a_i : i \in S\}]$. Thus, the problem reduces to LBOP.

Note that when feasible solutions are spanning trees of a graph on n nods and m edges, LBOP can be solved in $O(m \log n)$ time [61] and hence the resulting QBOP with decomposable cost function can be solved in $O(m^3 \log n)$ time.

Case 2: Product Case

In this case, we assume $q_{ij} = a_i.b_j$ where $a_i \geq 0$ and $b_j \geq 0$ for any $i \in S$ and for any $j \in S$. Then, we have:

$$\begin{aligned} Z(Q, S) &= \max\{q_{ij} : (i, j) \in S \times S\} - \min\{q_{ij} : (i, j) \in S \times S\} \\ &= \max\{a_i.b_j : (i, j) \in S \times S\} - \min\{a_i.b_j : (i, j) \in S \times S\} \\ &= \max\{a_i : i \in S\}.\max\{b_i : i \in S\} - \min\{a_i : i \in S\}.\min\{b_i : i \in S\} \end{aligned} \tag{2.6}$$

Let $g(S) = \max_{i \in S} a_i.\max_{i \in S} b_i$, by simple modification in the algorithm for solving max+sum combinatorial optimization problem [17] to multiplicative combinatorial optimization problem, $g(S)$ can be solved as a series of bottleneck problems in polynomial time and hence, QBOP with special structure of $q_{ij} = a_i.b_j$ can be solved in polynomial time.

Chapter 3

Quadratic Balanced Knapsack Problem(QBKP)

Let $E = \{1, ..., m\}$ and F be a family of subsets of E. For any $x = (x_1, ..., x_m) \in \{0,1\}^n$ let $S(x) = \{j : x_j = 1\}$. Then, QBKP can be defined as

$$\begin{aligned} \text{Minimize} \quad & \{\max_{(i,j)\in S(x)\times S(x)} q_{ij} - \min_{(i,j)\in S(x)\times S(x)} q_{ij}\} \\ \text{Subject to} \quad & \sum_{i\in S(x)} a_i x_i \geq c \\ & x_i \in \{0,1\} \quad \forall i \in E \end{aligned}$$

QBKP is NP-hard since if $q_{ij} \geq 0$ for all $(i,j) \in E \times E$ and $q_{ii} = 0$ for all $i \in E$ then QBKP reduces to *Quadratic Bottleneck Knapsack Problem* which is known to be NP-hard in [62].

Note that QBKP is a special case of QBOP discussed in the previous chapter. Thus, all the results discussed in Chapter 2 are applied here. In particular, the algorithms developed in Chapter 2 can be used to solve QBKP. The focus of this chapter is to explore how the general algorithms can be simplified by exploiting the special structure of QBKP.

3.1 The Double Threshold Algorithm for QBKP

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct q_{ij} values. For any real numbers α and β where $z_1 \leq \alpha \leq \beta \leq z_p$ let $E(\alpha, \beta) = \{(i,j) \in E \times E : q_{ij} > \alpha \text{ or } q_{ij} < \beta\}$

and $F(\alpha, \beta) = \{S \in F : (S \times S) \cap E(\alpha, \beta) = \emptyset\}$.

Consider the Maximum Weight Independent set Problem (MWIP)

$$\begin{aligned} \text{Maximize} \quad & \sum_{j=1}^{m} a_j x_j \\ \text{Subject to} \quad & x_i + x_j \leq 1 \quad \forall (i,j) \in E(\alpha, \beta) \quad (1) \\ & x_j \in \{0, 1\} \end{aligned}$$

Let $x^* = (x_1^*, x_2^*, ..., x_m^*)$ be an optimal solution to the MWIP and z^* be its optimal objective function value.

Theorem 9. *$F(\alpha, \beta) \neq \emptyset$ if and only if $z^* \geq c$.*

Proof. If $z^* \geq c$ then, x^* is a feasible solution to QBKP and $S(x^*) \in F(\alpha, \beta)$.

Conversely, suppose $F(\alpha, \beta) \neq \emptyset$. Choose $S \in F(\alpha, \beta)$ and let x^s be the incidence vector of s. Then x^s satisfies constraint (1). Since $S \in F$, $\sum_{i=1}^{m} a_i x_i^s \geq c$. But $z^* \geq \sum_{i=1}^{m} a_i x_i^s$. Hence, $z^* \geq c$ □

Thus, if we can solve the MWIP in polynomial time, then the feasibility problem associated with QBKP can be solved in polynomial time. Using the feasibility test discussed above, let us restate our double threshold algorithm for QBKP.

Theorem 10. *QBKP can be solved in $O(m^2 \psi(m))$ time, where $\psi(m)$ is the complexity of the MWIP.*

The structure of the matrix Q is such that certain pairs $(i, j) \in E \times E$ are explicitly prohibited. i.e. no feasible solution is allowed to have both i and j together for such explicitly prohibited pairs. Consider the graph $G = (V, A)$ with node set E and an edge $(i, j) \in A$ if and only if (i, j) is not explicitly prohibited. Then, if G is a bipartite graph, the MWIP can be solved in polynomial time. Also, any subgraph of a bipartite graph is also bipartite; Thus, the double threshold algorithm runs in polynomial time and hence this special case of QBKP can be solved in polynomial time.

Let us now give an integer linear program formulation of QBKP.

Theorem 11. *The following problem which is called QBKP-IP is equivalent to QBKP.*

Algorithm 3.1: The Double Threshold Algorithm (DT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$
Opt_Sol $= \emptyset$
Obj_Val $= \infty$
$l = u = 1$
while $l \leq p$ and $u \leq p$ **do**
  Solve Maximum Weight Independent set Problem(MWIP)
  Let z^* be the optimal objective value and x^* the optimal solution for MWIP
  if $z^* > c$ **then**
    $z_r = \min\{q_{ij} : (i, j) \in S(x^*) \times S(x^*)\}$
    $z_t = \max\{q_{ij} : (i, j) \in S(x^*) \times S(x^*)\}$
    if $z_t - z_r <$ Obj-Val **then**
      Obj_Val $= z_t - z_r$
      Opt_Sol $= S(x^*)$
    end if
    $l = r + 1$
  else
    $u = u + 1$
  end if
end while
Return Opt_Sol and Obj_Val

$$
\begin{aligned}
\min \quad & y - z \\
\text{Subject to} \quad & \sum_{j=1}^{m} a_j x_j \geq c \quad (1) \\
& q_{ij}\alpha_{ij} \leq y + M(1 - x_i) + M(1 - x_j) \quad (2) \\
& z \leq q_{ij}\alpha_{ij} + M(1 - x_i) + M(1 - x_j) \quad (3) \\
& \alpha_{ij} \leq x_i \quad (4) \\
& \alpha_{ij} \leq x_j \quad (5) \\
& x_i + x_j - 1 \leq \alpha_{ij} \quad (6) \\
& x_i \in \{0, 1\} \\
& \alpha_{ij} \in \{0, 1\} \\
& M \geq 0
\end{aligned}
$$

Proof. Let $(\alpha^*_{ij}, x^*_i, y^*, z^*)$ be the optimal solution to QBKP-IP, $y^* - z^*$ be the corresponding optimal objective function value and M be a large positive number. Constraints (4) , (5) and (6) force the value of α_{ij} to be equal to the value of $x_i x_j$ for the set: $\{(i, j) : (i, j) \in S(x) \times S(x)\}$. By constraint (1), x^* is also feasible for QBKP. Constraint (2) forces $y = \max\{q_{ij} : (i, j) \in S(x) \times S(x)\}$ and constraint (3) forces $z = \min\{q_{ij} : (i, j) \in S(x) \times S(x)\}$. By feasibility of $(\alpha^*_{ij}, x^*_i, y^*, z^*)$, we have $y^* - z^* \leq \min\{\max_{(i,j)\in S(x^*)\times S(x^*)} q_{ij} - \min_{(i,j)\in S(x^*)\times S(x^*)} q_{ij}\}$.

Now, consider x^0 to be the optimal solution and $z(x^0)$ to be the optimal objective function value of QBKP. The knapsack constraint is common in both QBKP and QBKP-IP, set $y^0 = \max\{q_{ij} : (i, j) \in S(x) \times S(x)\}$ and $z^0 = \min\{q_{ij} : (i, j) \in S(x) \times S(x)\}$. Constraints $(2), (3)$ and (4) are also satisfied by x^0 since the value of $x_i x_j$ results the value of α_{ij}. Since x^* is optimal to QBKP-IP, we have $z(x^0) = y^0 - z^0 \leq y^* - z^*$ (i). Also, $z(x^*) = y^* - z^* \leq z(x^0) = y^0 - z^0$ (ii) since x^0 is optimal to QBKP. From (i) and (ii), $y^* - z^* = y^0 - z^0$ and the result follows. □

3.1.1 The Improved Double Threshold Algorithm for QBKP

The Improved Double Threshold (IDT) algorithm helps to bound the search interval for optimal solution. This algorithm first solves Quadratic Bottleneck Knapsack Problem (QKP1

and QKP2). QKP1 and QKP2 are the instances of QBP1 and QBP2 as introduced in chapter 2.

$$
\begin{aligned}
QKP1: \quad \text{Minimize} \quad & \{\max_{(i,j)\in S(x)\times S(x)} q_{ij}\} \\
\text{Subject to} \quad & \sum_{j}^{m} a_j x_j \geq c \\
& x_j \in \{0,1\}
\end{aligned}
$$

$$
\begin{aligned}
QKP2: \quad \text{Maximize} \quad & \{\min_{(i,j)\in S(x)\times S(x)} q_{ij}\} \\
\text{Subject to} \quad & \sum_{j}^{m} a_j x_j \geq c \\
& x_j \in \{0,1\}
\end{aligned}
$$

The problem QKP1 was investigated by Zhang and Punnen [62] and they proposed exact algorithms and heuristic methods to solve this problem. QKP2 can be modified to QKP1 by converting its objective function to $\min\{\max -q_{ij} : (i,j) \in S(x) \times S(x)\}$. Similarly, QKP2 can be formulated as QKP1. In this sense both QKP1 and QKP2 are equivalent. QKP1 is known to be NP-hard[62] and since it can be converted to QKP2, QKP2 is also NP-hard.

QKP1 is solved by threshold algorithm much faster than QBKP. Since QKP1 has a monotonic objective function, we can use binary search for solving it [62] while we have to use linear search to solve QBKP by double threshold algorithm since QBKP objective function is not monotonic.

For any feasible solution $S(x)$, let $Z_1(S(x)) = \max\{q_{ij} : (i,j) \in S(x)\times S(x)\}$, $Z_2(S(x)) = \min\{q_{ij} : (i,j) \in S(x)\times S(x)\}$ and $Z(S(x)) = Z_1(S(x)) - Z_2(S(x))$. Let $S^1(x)$ be the optimal solution to QKP1 , U^* be its optimal objective function value and $S^2(x)$ be the optimal solution to QKP2 and L^* be its optimal objective function value.

Theorem 12. *For any feasible solution $S(x)$, $Z_2(S(x)) \leq L^*$. Further, for any solution with $Z_1(S(x)) = U^*$, if $Z_2(S(x)) < Z_2(S^1(x))$, then $Z(S(x)) > Z(S^1(x))$*

Proof. The inequality $Z_2(S(x)) \leq L^*$ follows from the definition of L^*. Also, for any feasible solution $S(x)$, $Z_1(S(x)) \geq Z_1(S^1(x))$. Thus, if $Z_2(S(x)) < Z_2(S^1(x))$ we have $Z(S(x)) = Z_1(S(x)) - Z_2(S(x)) > Z_1(S^1(x)) - Z_2(S^1(x)) = Z(S^1(x))$ and the proof is complete. □

Using Theorem 12, we can reset the starting lower threshold value as $Z_2(S^1(x))$ and the lower threshold is not required to increase beyond L^*. This could reduce the search interval considerably. The double threshold algorithm incorporating this enhancement is called the improved double threshold algorithm. A formal description of the algorithm is presented in Algorithm 3.2.

Algorithm 3.2: The Improved Double Threshold Algorithm (IDT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$
Let Opt_Sol be the optimal solution to QBKP and Obj_Val the optimal objective function value to QBKP
Let $S^1(x)$ be the optimal solution to QKP1 and U^* be the optimal objective function value of that, l be such that $z_l = min\{q_{ij} : (i, j) \in S^1(x) \times S^1(x)\}$. We set l and u be such that $(z_u \leftarrow U^*)$
Let L^* be such that z_{L^*} be the optimal objective function value of QKP2
Opt_Sol $\leftarrow S^1(x)$
Obj_Val $= z_u - z_l$
$l \leftarrow l + 1$
while $(l \leq u)$ and $(l \leq L^*)$ and $(u \leq p)$ **do**
  Solve Maximum Weight Independent set Problem(MWIP)
  Let z^* be the optimal objective value and x^* the optimal solution for MWIP
  if $z^* > c$ **then**
    $z_r = \min\{q_{ij} : (i, j) \in S(x^*) \times S(x^*)\}$
    $z_t = \max\{q_{ij} : (i, j) \in S(x^*) \times S(x^*)\}$
    if $z_t - z_r <$ Obj-Val **then**
      Obj_Val$= z_t - z_r$
      Opt_Sol$= S(x^*)$
    end if
    $l = r + 1$
  else
    $u \leftarrow u + 1$
  end if
end while
Return Opt_Sol and Obj_Val

3.1.2 The Modified Double Threshold Algorithm for QBKP

Let us now discuss how to improve the average performance of the double threshold algorithm and the improved double threshold algorithm. The improvement is achieved using

1) sufficient condition for optimality and
2) a sufficient condition that allows increments of upper and lower threshold values by larger amounts.

The conditions however does not seem to affect the worst case complexity of the algorithm.

Let $F^*(x) = \{S(x^1), ..., S(x^p)\}$ be the set of all solutions generated by the double threshold algorithm while completing the sequential search. We assume that $S(x^i)$ is generated before $S(x^j)$ for $i < j$. For $1 \leq t \leq p$ let, $\alpha_t = max\{q_{ij} : (i,j) \in S(x^t) \times S(x^t)\}$ and $\beta_t = \min\{q_{ij} : (i,j) \in S(x^t) \times S(x^t)\}$. Thus, $\alpha_1 < \alpha_2 < ... < \alpha_p$ and $\beta_1 < \beta_2 < ... < \beta_p$. For $k = 1, ..., p$. Choose $t(k)$ such that $\alpha_{t(k)} - \beta_{t(k)} = min\{\alpha_i - \beta_i : 1 \leq i \leq k\}$.

Let $F_k^*(x) = \{S^1(x), S^2(x), ..., S^k(x)\}$. Note that $S^{t(p)}(x)$ is an optimal solution to QBKP since $\alpha_{t(p)} - \beta_{t(p)} = \min\{\alpha_i - \beta_i : 1 \leq i \leq p\}$. Now, let $F^0(x) \subset F(x)$ such that $F^0(x) = \{S_i(x) \in F^*(x) : \alpha_i - \beta_i = \alpha_{t(p)} - \beta_{t(p)}\}$. Thus, any $S(x) \in F^0$ is an optimal solution to QBKP. Let γ be a real number such that $\gamma \geq \max\{\beta_i : S_i(x) \in F^0(x)\}$. We choose $\gamma = \min\{q_{ij} : (i,j) \in S^2(x) \times S^2(x)\}$ to obtain the best possible bound.

Theorem 13. *For any $1 \leq k \leq p$ if $\alpha_{t(k)} - \beta_{t(k)} + \gamma \leq \alpha_k$ then $S(x^{t(k)}) \in F^0(x)$.*

Proof. Suppose $S(x^{t(k)}) \in F^0(x)$ then there is an $S(x^i) \in F^0(x)$ and $i > k$ such that $\alpha_i - \beta_i < \alpha_{t(k)} - \beta_{t(k)}$. Then $\alpha_i < \beta_i + \alpha_{t(k)} - \beta_{t(k)} \leq \gamma + \alpha_{t(k)} - \beta_{t(k)} \leq \alpha_k$ this implies $i \leq k$ which is a contradiction. □

Theorem 13 adds a termination criterion to IDT algorithm which reduces the number of iterations.

Theorem 14. *If $S(x)$ does not belong to $F^0(x)$ then there is an $S(x^q) \in F^0$ for $q > k$ such that $\beta_q > \alpha_k - \alpha_{t(k)} + \beta_{t(k)}$.*

Proof. On contrary suppose $\beta_q \leq \alpha_k - \alpha_{t(k)} + \beta_{t(k)} \leq \alpha_q - \alpha_{t(k)} + \beta_{t(k)}$ then $\alpha_q - \beta_q \geq \alpha_{t(k)} + \beta_{t(k)}$ then $S(x^{t(k)}) \in F^0$ which is a contradiction. □

Theorem 14 could be used to update the lower threshold by larger quantity whenever the condition holds.

Results similar to Theorem 13 and 14 were used by Ahuja [2], Martins[40], Punnen [45] and Punnen and Nair [49] in solving other combinatorial optimization problems. Incorporating the conditions presented in Theorems 13 and 14 into the improved double threshold

algorithm, we get the modified double threshold algorithm to solve QBKP. A formal description of the modified double threshold algorithm is given in Algorithm 3.3.

3.1.3 The Iterative bottleneck knapsack algorithm for QBKP

Let us now discuss two additional algorithms to solve QBKP. In both cases, we solve a sequence of quadratic bottleneck knapsack problems (QKP1 or QKP2) instead of MWIP problem. The worst case complexities of these algorithms , in general, are higher than that of the DT algorithm and its variations, the average performance is expected to be better. Recall that

$$\begin{aligned} QKP1: \quad \text{Minimize} \quad & \{\max_{(i,j)\in S(x)\times S(x)} q_{ij}\} \\ \text{Subject to} \quad & \sum_{j}^{m} a_j x_j \geq c \\ & x_j \in \{0,1\} \end{aligned}$$

$$\begin{aligned} QKP2: \quad \text{Maximize} \quad & \{\min_{(i,j)\in S(x)\times S(x)} q_{ij}\} \\ \text{Subject to} \quad & \sum_{j}^{m} a_j x_j \geq c \\ & x_j \in \{0,1\} \end{aligned}$$

QKP1 minimizes $Z_1(Q, S(x))$ while QKP2 maximizes $Z_2(Q, S(x))$.

To develop our first iterative bottleneck knapsack algorithm, we consider a generalization of QBKP where lower and upper threshold restrictions are imposed on feasible solutions.

Algorithm 3.3: The Modified Double Threshold Algorithm (MDT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$
Let Opt_Sol be the optimal solution to QBKP and Obj_Val the optimal objective function value to QBKP
Let $S^1(x)$ be the optimal solution to QKP1 and U^* be the optimal objective function value of that, l be such that $z_l = min\{q_{ij} : (i, j) \in S^1(x) \times S^1(x)\}$. We set l and u be such that $(z_u \leftarrow U^*)$
Let L^* be such that z_{L^*} be the optimal objective function value of QKP2
Opt_Sol $\leftarrow S^1(x)$
Obj_Val $= z_u - z_l$
$l \leftarrow l + 1$
while $(l \leq u)$ and $(l \leq L^*)$ and $(u \leq p)$ **do**
  Solve Maximum Weight Independent set Problem(MWIP)
  Let z^* be the optimal objective value and x^* the optimal solution for MWIP
  if $z^* > c$ **then**
    $z_r = \min\{q_{ij} : (i, j) \in S(x^*) \times S(x^*)\}$
    $z_t = \max\{q_{ij} : (i, j) \in S(x^*) \times S(x^*)\}$
    if $z_t - z_r <$ Obj-Val **then**
      Obj_Val= $z_t - z_r$
      Opt_Sol= $S(x^*)$
    end if
    if Obj_Val $+L^* \leq z_r$ **then**
      return Obj_Val and Opt_Sol
    end if
    $l = r + 1$
    choose smallest $q \geq l$ such that $z_q \geq z_u -$ Obj_val
    $l \leftarrow q$
  else
    $u \leftarrow u + 1$
  end if
end while
Return Opt_Sol and Obj_Val

Let α and β be real numbers such that $\alpha \leq \beta$. Consider the problem

$$\begin{aligned} \text{QBKP(Q,}\alpha,\beta\text{):= Minimize} \quad & Z_1(Q,S(x)) - Z_2(Q,S(x)) \\ \text{Subject to} \quad & \sum_{i \in S(x)} a_i x_i \geq c \\ & Z_2(Q,S(x)) \geq \alpha \\ & Z_1(Q,S(x)) \leq \beta \\ & x_i = 0,1 \quad \forall i \in E \end{aligned}$$

Consider the cost matrix Q' defined by

$$q'_{ij} = \begin{cases} M & \text{if } q_{ij} < \alpha \text{ or } q_{ij} > \beta \\ q_{ij} & \text{otherwise.} \end{cases}$$

where M is a large number.

Theorem 15. *Let $S(x^0)$ be an optimal solution to QKP1 with cost matrix Q' and q be the index such that $z_q = Z_2(Q,S(x^0))$.*
(i) If $Z_1(Q,S(x^0)) = M$ then QBKP(Q,α,β) is infeasible.
(ii) If $Z_1(Q,S(x^0)) < M$ and $Z_1(Q,S(x^0)) = Z_2(Q,S(x^0))$ then $S(x^0)$ is an optimal solution to QBKP(Q,α,β).
(iii) If conditions (i) and (ii) above are not satisfied, then either $S(x^0)$ is an optimal solution to QBKP(Q,α,β) or there exists an optimal solution to QBKP(Q,α,β) which is optimal to QBKP(Q,γ,β) where $\gamma = z_{q+1}$.

Proof. The proof of (i) and (ii) are straightforward. Let us now prove (iii).

Let $F = \{S(x) \in F(\alpha,\beta) : Z_2(Q,S(x)) \leq Z_2(Q,S(x^0))\}$. By definition of F

$$Z_2(Q,S(x^0)) \geq Z_2(Q,S(x)) \text{ for all } S(x) \in F. \tag{3.1}$$

By optimality of $S(x^0)$ to QKP1 with cost matrix Q' and condition (i) of the theorem is not satisfied, we have

$$Z_1(Q,S(x^0)) \leq Z_1(Q,S(x)) \text{ for all } S(x) \in F. \tag{3.2}$$

Multiply inequality (3.1) by -1 and adding to inequality (3.2) we have $\text{QBKP}(Q,S(x^0)) \leq$

QBKP$(Q, S(x))$ for all $S(x) \in F$. Thus either $S(x^0)$ is an optimal solution to QBKP(Q,α, β) or there exists an optimal solution $S(x)$ to QBKP(Q,α, β) satisfying $Z_2(Q, S(x)) > Z_2(Q, S(x^0))$ and the result follows. □

In view of Theorem 15, we can solve QBKP as a sequence of QKP1 problems. In each iteration, the algorithm maintains an upper threshold β and a lower threshold α and construct a modified cost matrix Q' which depends on the values of α and β. Then using an optimal solution to QBP1 with cost matrix Q', the lower threshold is updated until infeasibility with respect to the threshold values is identified or optimality of an intermediate solution is identified using condition (ii) of Theorem 15. The algorithm compares the solutions generated by the QKP1 solver and outputs the overall best solution with respect to the QBKP objective function. The resulting algorithm is called the *type 1 iterative bottleneck knapsack algorithm* (IBK1-algorithm) and its formal description is given in Algorithm 3.4.

QKP2 can be reformulated as QKP1 or the algorithms for QKP1 can be modified to solve QKP2 directly. Consider the cost matrix $\tilde{Q}$ defined by

$$\tilde{q}_{ij} = \begin{cases} -M & \text{if } q_{ij} > \beta \text{ or } q_{ij} < \alpha \\ q_{ij} & \text{otherwise.} \end{cases}$$

where M is a large number.

Theorem 16. *Let $S(x^0)$ be an optimal solution to QKP2 with cost matrix $\tilde{Q}$ and r be the index such that $z_r = Z_1(Q, S(x^0))$.*
(i) If $Z_2(\tilde{Q}, S(x^0)) = -M$ then QBKP(Q,α, β) is infeasible.
(ii) If $Z_2(\tilde{Q}, S(x^0)) > -M$ and $Z_1(Q, S(x^0)) = Z_2(Q, S(x^0))$ then $S(x^0)$ is an optimal solution to QBKP(Q,α, β).
(iii) If conditions (i) and (ii) are not satisfied, then either $S(x^0)$ is an optimal solution to QBKP(Q,α, β) or there exists an optimal solution to QBKP(Q,α, β) which is also optimal to QBKP(Q,α, γ) where $\gamma = z_{r+1}$.

The proof of this theorem can be constructed by appropriate modifications in the proof of Theorem 15 and hence is omitted. In view of Theorem 16, we can solve QBKP as a sequence of QKP2 problems. In each iteration, the algorithm maintains a lower threshold α and an upper threshold β and construct a modified cost matrix $\tilde{Q}$. Using an optimal solution

to QKP2 with cost matrix $\tilde{Q}$, the upper threshold is updated and the process is continued until infeasibility with respect to the threshold values is identified. The algorithm compares the solutions generated by the QKP2 solver in each iteration and outputs the overall best solution. The resulting algorithm is called the *type 2 iterative bottleneck knapsack algorithm* (IBK2-algorithm) and its formal description is given in Algorithm 3.5.

Algorithm 3.4: The type1 Iterative Bottleneck Knapsack Algorithm(IBK1)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$
Let Q be the cost matrix , Opt_Sol be the optimal solution of QBKP and Obj_Val be the objective function value of QBKP in each iteration
$Q' \leftarrow Q$; $Obj\_Val \leftarrow \infty$; $Opt\_Sol \leftarrow \emptyset$, $M \leftarrow 1 + z_p$; $Z_0 = z_p$;
while $(Obj\_Val \neq 0)$ and $Z_0 \neq M$ **do**
  Solve QKP2 with cost matrix Q'. Let $S(x)$ be the resulting solution.
  $Z_0 \leftarrow \max\{q'_{ij} : (i, j) \in S(x) \times S(x)\}$
  if $Z_0 < M$ **then**
    $Z_1 \leftarrow \max\{q_{ij} : (i, j) \in S(x) \times S(x)\}$
    $Z_2 \leftarrow \min\{q_{ij} : (i, j) \in S(x) \times S(x)\}$
    if $Z_1 - Z_2 < Obj - Val$ **then**
      $Opt\_Sol \leftarrow S(x)$
      $Obj\_Val \leftarrow Z_1 - Z_2$
    end if
    Modify costs:
    $$q'_{ij} = \begin{cases} q_{ij} & \text{if } Z_2 < q_{ij} < z_p \\ M & \text{Otherwise} \end{cases}$$
  end if
end while
Return Opt_Sol and Obj_Val

Algorithms 3.4 and 3.5 can be viewed as extension of an algorithm by Duin and Volgenant [17] for a generalization of the LBOP.

3.2 Polynomially solvable cases for Quadratic Balanced Knapsack Problem

Let us now consider some special cases of QBKP that can be solved in polynomial time.

Case 1: Sum Case

Algorithm 3.5: The type2 Iterative Bottleneck Knapsack Algorithm(IBK2)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ij} : (i, j) \in E \times E$

Let Q be the cost matrix , Opt_Sol be the optimal solution of QBKP and Obj_Val be the objective function value of QBKP in each iteration

$\tilde{Q} \leftarrow Q$; $Obj\_Val \leftarrow \infty$; $Opt\_Sol \leftarrow \emptyset$, $M \leftarrow 1 + z_p$; $Z_0 = z_1$;

while $(Obj\_Val \neq 0)$ and $Z_0 \neq -M$ **do**

  Solve QKP2 with cost matrix $\tilde{Q}$. Let $S(x)$ be the resulting solution.

  $Z_0 \leftarrow \min\{\tilde{q}_{ij} : (i, j) \in S(x) \times S(x)\}$

  if $Z_0 > -M$ **then**

    $Z_1 \leftarrow \max\{q_{ij} : (i, j) \in S(x) \times S(x)\}$

    $Z_2 \leftarrow \min\{q_{ij} : (i, j) \in S(x) \times S(x)\}$

    if $Z_1 - Z_2 < Obj - Val$ **then**

      $Opt\_Sol \leftarrow S(x)$

      $Obj\_Val \leftarrow Z_1 - Z_2$

    end if

    Modify costs:

    $$\tilde{q}_{ij} = \begin{cases} q_{ij} & \text{if } z_1 < q_{ij} < Z_1 \\ -M & \text{Otherwise} \end{cases}$$

  end if

end while

Return Opt_Sol and Obj_Val

We first consider the *decomposable cost matrix* where $q_{ij} = e_i + f_j$ for given $e_i \geq 0$ and $f_j \geq 0$ where $i, j \in E$. Note that

$$\begin{aligned} Z(Q, S(x)) &= \max\{q_{ij} : (i,j) \in S(x) \times S(x)\} - \min\{q_{ij} : (i,j) \in S(x) \times S(x)\} \\ &= \max\{e_i + f_j : (i,j) \in S(x) \times S(x)\} - \min\{e_i + f_j : (i,j) \in S(x) \times S(x)\} \\ &= \max\{e_i : i \in S(x)\} + \max\{f_i : i \in S(x)\} - \min\{e_i : i \in S(x)\} - \min\{f_i : i \in S(x)\} \quad (3.3) \\ &= \max\{e_i : i \in S(x)\} + \max\{-e_i : i \in S(x)\} + \max\{f_i : i \in S(x)\} - \min\{f_i : i \in S(x)\} \quad (3.4) \end{aligned}$$

Recall from Chapter 2 that if w_i be a prescribed weight of element $i \in E$ and $g : F \to \Re$. Duin and Volgenant [17] showed that combinatorial optimization problems of the type

$$\begin{aligned} \text{COP}(g): \quad & \text{Minimize } \max\{w_i : i \in S\} + g(S) \\ & \text{Subject to} \\ & \qquad S \in F. \end{aligned}$$

can be solved in $O(m\zeta(m))$ where $\zeta(m)$ is the complexity of minimizing $g(S)$ over F. Note that Here, by considering the special case of knapsack problem we have Knapsack COP (KCOP):

$$\begin{aligned} KCOP(g) \quad \text{Minimize} \quad & \max\{w_i : i \in S(x)\} + g(S(x)) \\ \text{Subject to} \quad & \sum_{i \in S(x)} a_i x_i \geq c \\ & x_i \in \{0,1\} \quad \forall i \in E \end{aligned}$$

Let

$$Z(Q, S(x)) = \max\{e_i : i \in S(x)\} + g(S(x)) \quad (3.5)$$

where $g(S(x)) = \max\{-e_i : i \in S(x)\} + g_1(S(x))$ and $g_1(S(x)) = \max\{f_i : i \in S(x)\} - \min\{f_i : i \in S(x)\}$. But minimizing $g_1(S(x))$ over F is precisely the *Linear Balanced Knapsack Problem*(LBKP) . Thus recursively applying the results of Duin and Volgenant

[17], BKCOP with a decomposable cost matrix can be solved in $O(m^2\eta(m))$ time where $\eta(m)$ is the complexity of an LBKP.

When $e_i = f_i$, then $Z(Q,S) = 2[\max\{e_i : i \in S\} - \min\{e_i : i \in S\}]$. Thus, the problem reduces to LBKP.

Case 2 : Product Case

In this case, we assume $q_{ij} = e_i.f_j$ where $e_i \geq 0$ and $f_j \geq 0$ for any $i \in S(x)$ and for any $j \in S(x)$. Then, we have:

$$\begin{aligned} Z(Q,S(x)) &= \max\{q_{ij} : (i,j) \in S(x) \times S(x)\} - \min\{q_{ij} : (i,j) \in S(x) \times S(x)\} \\ &= \max\{e_i.f_j : (i,j) \in S(x) \times S(x)\} - \min\{e_i.f_j : (i,j) \in S(x) \times S(x)\} \\ &= \max\{e_i : i \in S(x)\}.\max\{f_i : i \in S(x)\} - \min\{e_i : i \in S(x)\}.\min\{f_i : i \in S(x)\} \end{aligned} \tag{3.6}$$

Let $g(S(x)) = \max_{i\in S(x)} e_i.\max_{i\in S(x)} f_i$, Zhang and Punnen [62] proved that $g(S(x))$ can be solved in $O(m^2)$ and hence, QBKP with special structure of $q_{ij} = e_i.f_j$ can be solved in polynomial time.

3.3 Solving QBKP by a heuristic

In this section we formulate MWIP as an Unconstrained Quadratic Problem (UQP) and solve it by the heuristic method presented in [38] UQP is formulated as follows:

$$\begin{aligned} \max \quad & \sum_{i=1}^{m}\sum_{j=1}^{m} q_{ij}x_ix_j \\ & x_j \in \{0,1\} \end{aligned}$$

Let z_l be the lower threshold and z_u be the upper threshold in any of the algorithms for QBKP in this chapter.We formulate MWIP as UQP by defining costs c_{ij} as follows:

$$c_{ij} = \begin{cases} a_j & \text{if } i = j \text{ and } z_l \leq q_{ij} \leq z_u \\ -M & \text{if } q_{ij} < z_l \text{ or } q_{ij} > z_u \\ 0 & \text{Otherwise} \end{cases}$$

where M is a large positive number. If UQP with modified costs of c_{ij} has a solution with objective function value of greater than or equal to c then, QBKP has a feasible solution.

3.4 Computational Results:

All the algorithms in Chapter 3 were coded in C^{++} and executed on a Linux workstation with Intel Xeon E5410 CPU (2.33 GHz) and 8 GB RAM running Red Hat Enterprise Linux (kernel 2.6.18).

The MWIP problem in all exact algorithms of this chapter were solved with CPLEX 12.1.0.

Since there is no benchmark problems for QBKP, all the experiment instances were generated randomly with different ranges for costs and weights. There were two parameters considered in generating these random instances, N and PCT, where N is the number of variables (items) and PCT is the percentage of zero elements in cost matrix. PCT is 0 , 25 and 50 for all instances, $N = 10, ..., 90$ in table 3.1 and $N = 100, ..., 700$ in table 3.2 , table 3.3 and table 3.4. The instances in second table are reasonably of large size since for $N = 700$ and $PCT = 0$ there are 490000 q_{ij} values.

For each instance with size of $N = 100, 200, 300, 400, 500, 600, 700$ and $PCT = 0, 25$ and 50 three different random data sets were generated for the experiments on and the results are reported in three tables.

In table 3.1, problem instances names are in the form of $xx - y$ where xx presents N and $y = 1, 2, 3$ correspondingly stands for $PCT = 0, 25, 50$ and in tables 3.2 , 3.3 and 3.4 names are presented in $xxx - y$ form where xxx stands for the N and y indicates PCT as in table 3.1.

Since all of the algorithms are exact, the optimal value column represents the optimal value for all of the algorithms. Also, the CPU time for each algorithm is reported in the tables.

In tables 3.2 , 3.3 and 3.4, since Cplex can not solve large instances, there is no column for Cplex CPU time. Also, there is a heuristic algorithm in addition to exact algorithms. Thus in addition to above mentioned information reported for the exact algorithms, the optimal value of the heuristic method is also reported.

Considering table 3.1, Cplex can not(or it takes several days) solve instances with size

Table 3.1: Table of small instances results

Instance	Optimal Value	Cplex	DT	IDT	MDT	IBK2
40-1	98	70.74	0.79	0.04	0.06	0.04
40-2	85	1532.57	0.98	0.06	0.06	0.06
40-3	90	46.1	1.21	0.05	0.04	0.04
50-1	87	146108	1.31	0.39	0.37	0.14
50-2	87	13602.7	3.33	0.20	0.24	0.14
50-3	97	27.44	2.11	0.10	0.08	0.07
60-1	88	134388	6.46	1.49	1.48	0.19
60-2	91	122209	3.15	0.09	0.08	0.11
60-3	99	12.99	7.49	0.18	0.20	0.08
70-1	99	90363.6	4.75	0.36	0.40	0.57
70-2	77	16771	14.24	1.24	1.45	0.5
70-3	38	835.09	4.82	0.58	0.58	0.34
80-1	-	-	9.66	4.76	4.31	0.59
80-2	-	-	17.3	1.06	1.06	0.72
80-3	-	-	16.86	0.95	0.94	0.25
90-1	-	-	19.38	1.48	1.46	1.24
90-2	-	-	14.96	0.89	0.91	0.93
90-3	-	-	15.99	3.30	3.31	0.71

over 70. While other algorithms solve rather large size instances. Among all exact algorithms in the first table, IBK2 algorithm is noticeably the fastest one. MDT is in most cases faster than IDT and obviously DT is the slowest one. In tables 3.2, 3.3 and 3.4 DT, IDT , MDT and IBK2 keep the same pattern as in table 3.1 but the heuristic method is dramatically faster than any of other exact algorithms as the size of instances grow from 300 . In some cases the optimal heuristic value is exact and in other cases it is reasonably close to exact optimal value.

Table 3.2: Table of large instance for $q \leq 100$

Instance	Opt-Val	H Opt-val	DT	IDT	MDT	IBK2	Heuristic
100-1	98	99	30.34	2.3	2.32	3.01	240.12
100-2	99	100	29	2.01	2.01	2.57	300.13
100-3	0	84	0.28	2.71	2.71	0.41	0.20
200-1	97	99	583.33	426.68	428.73	687.96	240.22
200-2	100	100	2275.84	193.28	193.09	45.25	300.21
200-3	0	84	2.53	67.72	66.35	2.92	0.39
300-1	99	99	11515.1	1803.66	1722.89	1717.77	300.39
300-2	97	100	16954.3	4052.19	4057.49	1231.31	300.4
300-3	100	100	22826.8	1794.75	1712.76	26.25	300.26
400-1	99	99	9206.65	1410.68	1412.65	1328.04	300.39
400-2	84	100	10034	1183.92	1109.95	664.14	300.27
400-3	100	100	18095.6	1599.73	1568.71	153.23	300.31
500-1	98	99	11274.4	2814.84	2670.13	1347.88	300.24
500-2	100	100	13316.1	1312.23	1225.66	1213.06	300.46
500-3	99	100	16082.2	1817.22	1783.84	1165.24	300.45
600-1	97	99	11979.6	2537.1	2189.64	1005.69	240.82
600-2	100	100	14617.4	1126.43	1193.95	1069.42	300.59
600-3	100	100	15844.7	1258.83	1285.19	546.78	300.55
700-1	99	99	11133.8	2771.43	1935	875.31	300.67
700-2	100	100	15417.1	1475.02	823.78	889.44	300.82
700-3	100	100	17780.2	1895.86	1769.42	1034.72	300.74

Table 3.3: Table of large instance for $q \leq 300$

Instance	Opt-Val	H Opt-val	DT	IDT	MDT	IBK2	Heuristic
100-1	288	298	74.2	5.53	2.31	1.78	180.15
100-2	50	274	10.51	6.57	1.61	0.16	0.29
100-3	231	300	219.77	11.66	3.23	0.61	360.17
200-1	291	299	8507.96	760.46	358.16	121.92	180.31
200-2	300	300	12102.30	397.66	225.78	47.94	420.41
200-3	266	300	36412.20	853.70	851.40	20.50	420.7
300-1	288	299	37209.20	13969.9	6532.04	1294.16	180.45
300-2	300	300	37475.50	1620.29	853.13	432.11	420.34
300-3	262	300	36269.40	4214.20	1651.02	248.11	360.55
400-1	288	299	30573.10	10800.6	4743.8	974.34	180.66
400-2	275	300	29186.20	5213.18	921.19	510.74	360.49
400-3	291	300	49317.10	4564.46	1314.40	425.17	360.48
500-1	299	299	30004.8	12789.87	5876.26	1107.9	360.56
500-2	215	297	33707	9664.53	635.77	419.65	121.59
500-3	298	300	44528.60	1734.02	488.69	539.72	360.67
600-1	299	299	58127.67	15089.67	6781.33	1019.09	420.72
600-2	300	300	39933	1782.36	834.87	518.06	420.65
600-3	300	300	46720.60	1617.91	664	394.34	361.04
700-1	299	299	67376.33	17634.50	7787.91	876.55	420.62
700-2	299	300	45709.10	2115.11	838.55	935.51	361.08
700-3	290	300	47753.10	4076.11	1269.12	604.32	360.62

Table 3.4: Table of large instance for $q \leq 600$

Instance	Opt-Val	H Opt-val	DT	IDT	MDT	IBK2	Heuristic
100-1	583	599	202.1	4.9	2.8	1.7	120.2
100-2	596	600	295.8	4.5	3.1	1.5	420.2
100-3	598	600	417.6	2.8	2.3	0.2	420.2
200-1	598	599	13004.2	296.5	287.9	200.9	360.2
200-2	591	600	27098.8	550.5	542.7	47.9	420.4
200-3	557	600	36412.2	853.7	851.4	20.5	420.7
300-1	586	599	56959.5	12758.1	10037.8	652.1	181.6
300-2	596	600	76694	1295.4	1249	376.6	420.7
300-3	592	600	114.1	1332.1	1336.8	25.8	420.8
400-1	591	599	50695	7341.8	6875.8	616.7	181.8
400-2	460	595	555594.8	1115.2	868.3	287.9	3.6
400-3	599	600	88260.6	1004.2	904.2	169.9	421.6
500-1	593	599	52816.9	4553.8	3661	617.3	241.8
500-2	596	600	63626.9	1482.6	1493.9	420.5	422.6
500-3	549	600	75899.9	967.7	853.6	453.9	422
600-1	598	599	57675.9	4553.8	1285.6	716.6	301.1
600-2	599	600	81386.9	897.1	963.3	716.6	421.1
600-3	599	600	96542.5	857.6	852.9	430.8	420.9
700-1	587	599	72306.6	6745.2	5181.2	993.1	181.5
700-2	600	600	85109.7	2217.8	1178.9	484.7	421.1
700-3	566	600	97758.7	1734.9	1225	503.1	421.2

Chapter 4

Quadratic Balanced Assignment Problem(QBAP)

Let G be a complete bipartite graph on the node set $V = \{1, ..., n\}$, $E = \{e_1, ..., e_m\}$ be the set of edges and F be the family of all perfect matchings, M on E. Let

$$x_e = \begin{cases} 1 & \text{if edge e is picked in the assignment} \\ 0 & \text{Otherwise} \end{cases}$$

QBAP is formulated on G as follows:

$$\begin{aligned} \text{Minimize} \quad & \{\max_{(e,f)\in M\times M} q_{(e,f)} - \min_{(e,f)\in M\times M} q_{(e,f)}\} \\ \text{Subject to} \quad & \sum_{e\in\Delta(i)} x_e = 1 \quad \forall i \in V \\ & x_e \in \{0,1\} \end{aligned}$$

where $\Delta(i) = \{$e: e is an edge incident to node i $\}$

QBAP is NP-hard since if $q_{ef} \geq 0$ for all $(e, f) \in E \times E$ and $q_{ee} = 0$ for all $e \in E$ then QBAP reduces to *Quadratic Bottleneck Assignment Problem* which is well known to be NP-hard in [8].

4.1 The Double Threshold Algorithm for QBAP

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of q_{ef}. For any real numbers α and β where $z_1 \leq \alpha \leq \beta \leq z_p$ let $E(\alpha, \beta) = \{(e, f) \in E \times E : q_{ef} > \alpha \text{ or } q_{ef} < \beta\}$ and $F(\alpha, \beta) = \{M \in F : M \times M \cap E(\alpha, \beta) = \emptyset\}$. The following problem which we call Assignment problem Feasibility Check (AFC) is used to check feasibility of QBAP:

$$\begin{aligned} \max \quad & 0 \\ \text{Subject to} \quad & \sum_{e \in \Delta(i)} x_e = 1 \quad \forall j = 1, ..., n \\ & x_e + x_f \leq 1 \quad \forall (e, f) \in E(\alpha, \beta) \\ & x_e \in \{0, 1\} \end{aligned}$$

If AFC problem has a feasible solution then, there is a feasible solution for QBAP with $\alpha \leq q_{ef} \leq \beta$ ($F(\alpha, \beta) \neq \emptyset$) .Choose indices l and u such that $z_u = \max\{q_{ef} : (e, f) \in M \times M\}$ and $z_l = \min\{q_{ef} : (e, f) \in M \times M\}$. Then, for any α and β such that $\beta - \alpha < z_u - z_l$ there is no feasible solution to the QBAP. Furthermore, if QBAP is infeasible then, for any γ and δ such that $\alpha \leq \gamma \leq \delta \leq \beta$ QBAP is infeasible.
Thus, if we solve the AFC problem in polynomial time, then by calling Cplex in a systematic sequential search to solve AFC problem, QBAP can be solved in polynomial time.

Theorem 17. *QBAP can be solved in $O(m^2\eta(m))$ time, where $\eta(m)$ is the complexity of the AFC problem.*

Using the AFC problem as feasibility test, let us restate our double threshold algorithm for QBAP.

By solving QBAP on a complete bipartite graph, Q matrix will consist of n^4 elements. Thus, per iteration Cplex deals with a huge number of conflict pairs constraints which makes it to take unpredictable amount of time to solve the AFC problem or even it might be impossible to solve AFC peoblem after a very long time. Thus, we are able to solve QBAP just for small instances [see table 4.1]. To overcome this difficulty, in the following section we solve the problem on a sparse graph which contains at least one random perfect matching to guarantee a feasible solution.

Let us now give an integer linear formulation to QBAP.

Algorithm 4.1: Double Threshold (DT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of
$q_{ef} : (e, f) \in M \times M$ where M is a perfect matching where $M \in F$
Opt-Sol $= \emptyset$
Obj-Val $= \infty$
$l = u = 1$
while $l \leq p$ and $u \leq p$ **do**
  Solve AFC problem
  if there is a feasible solution to AFC problem such as x^0 with M^0 as its corresponding perfect matching **then**
    $z_r = \min\{q_{ef} : (e, f) \in M \times M\}$
    $z_t = \max\{q_{ef} : (e, f) \in M \times M\}$
    if $z_t - z_r < Obj - Val$ **then**
      $Obj - Val = z_t - z_r$
      $Opt - Sol = M^0$
    end if
    $l = r + 1$
  else
    $u = u + 1$
  end if
end while
Return Opt-Sol and Opt-Val

Theorem 18. *The following problem which is called QBAP-IP is equivalent to QBAP.*

$$
\begin{aligned}
QBAP-IP: \quad & min && y-z \\
& Subject\ to && \sum_{e\in\Delta(i)} x_e = 1 \quad (1) \\
& && q_{ef}\alpha_{ef} \le y + H(1-x_e) + H(1-x_f) \quad (2) \\
& && z \le q_{ef}\alpha_{ef} + H(1-x_e) + H(1-x_f) \quad (3) \\
& && \alpha_{ef} \le x_e \quad (4) \\
& && \alpha_{ef} \le x_f \quad (5) \\
& && x_e + x_f - 1 \le \alpha_{ef} \quad (6) \\
& && x_e \in \{0,1\} \\
& && \alpha_{ef} \in \{0,1\}
\end{aligned}
$$

Proof. First we prove that the optimal solution of QBAP-IP is feasible for QBAP. Let $(\alpha^*_{ij}, x^*_i, y^*, z^*)$ be the optimal solution, M^* be the corresponding optimal perfect matching , $y^* - z^*$ be the optimal objective function value to QBAP-IP and H be a large positive number. Constraints (4) , (5) and (6) force the value of α_{ef} to be equal to the value of $x_e x_f$ for the set: $\{(e,f) : (e,f) \in M \times M\}$

By constraint (1), x^* is also feasible for QBAP. Constraint (2) forces $y = \max\{q_{ef} : (e,f) \in M \times M\}$ and constraint (3) forces $z = \min\{q_{ef} : (e,f) \in M \times M\}$. By feasibility of $(\alpha^*_{ij}, x^*_i, y^*, z^*)$, we have $y^* - z^* \le \min\{max_{(e,f)\in M^*\times M^*} q_{ef} - min_{(e,f)\in M^*\times M^*} q_{ef}\}$.

Now, we prove that the optimal solution of QBAP is feasible for QBAP-IP. Consider x^0 to be the optimal solution and $z(x^0)$ to be the optimal objective function value of QBAP. Constraint (1) is common between QBAP and QBAP-IP, let $y := \max\{q_{ef} : (e,f) \in M \times M\}$ and $z := \min\{q_{ef} : (e,f) \in M \times M\}$ then, constraints (2), (3) and (4) are satisfied by x^0 since the value of $x_e x_f$ results the value of α_{ef}. Then, by optimality of x^0, $z(x^0) \le y^0 - z^0$ where $y^0 - z^0$ is the optimal objective function value of QBAP-IP for x^0. Hence, QBAP-IP is equivalent to QBAP. □

4.1.1 The Improved Double Threshold Algorithm for QBAP

The Improved Double Threshold (IDT) algorithm helps to bound the search interval for optimal solution. This algorithm first solves Quadratic Bottleneck Assignment Problem

(QAP1 and QAP2). QAP1 and QAP2 are the instances of QBP1 and QBP2 as introduced in chapter 2.

$$\begin{aligned} QAP1: \quad & \text{Minimize} && \{\max_{(e,f)\in M\times M} q_{(e,f)}\} \\ & \text{Subject to} && \sum_{e\in\Delta(i)} x_e = 1 \quad \forall i \in V \\ & && x_e \in \{0,1\} \end{aligned}$$

$$\begin{aligned} QAP2: \quad & \text{Maximize} && \{\min_{(e,f)\in M\times M} q_{(e,f)}\} \\ & \text{Subject to} && \sum_{e\in\Delta(i)} x_e = 1 \quad \forall i \in V \\ & && x_e \in \{0,1\} \end{aligned}$$

QAP2 can be modified to QAP1 by converting its objective function to $\min\{\max -q_{ef} : (e,f) \in M \times M\}$. Similarly, QAP1 can be formulated as QAP2. In this sense both QAP1 and QAP2 are equivalent. QAP1 is known to be NP-hard[8] and since it can be converted to QAP2, QAP2 is also NP-hard.

Since QAP1 has a monotonic objective function, we can use binary search to solve QAP1 while we have to use linear search to solve QBAP by double threshold algorithm since QBAP objective function is not monotonic.

For any perfect matching M, let $Z_1(M) = \max\{q_{ef} : (e,f) \in M \times M\}$, $Z_2(M) = \min\{q_{ef} : (e,f) \in M \times M\}$ and $Z(M) = Z_1(M) - Z_2(M)$. Let $\bar{M}$ to be an optimal perfect matching for QAP1 and U^* be its optimal objective function value. Let $\hat{M}$ be the optimal perfect matching for QAP2 and L^* be its optimal objective function value.

Theorem 19. *For any feasible solution M, $Z_2(M) \leq L^*$. Further, for any solution with $Z_1(M) = U^*$, if $Z_2(M) < Z_2(\bar{M})$, then $Z(M) > Z(\bar{M})$*

Proof. The inequality $Z_2(M) \leq L^*$ follows from the definition of L^*. Also, for any feasible solution M, $Z_1(M) \geq Z_1(\bar{M})$. Thus, if $Z_2(M) < Z_2(\bar{M})$ we have $Z(M) = Z_1(M) - Z_2(M) > Z_1(\bar{M}) - Z_2(\bar{M}) = Z(\bar{M})$ which completes the proof. □

Using Theorem 19, we can reset the starting lower threshold value as $Z_2(\bar{M})$ and the

lower threshold is not required to increase beyond L^*. This could reduce the search interval considerably. The double threshold algorithm incorporating this enhancement is called the improved double threshold algorithm. A formal description of the algorithm is presented in Algorithm 4.2.

Algorithm 4.2: The Improved Double Threshold Algorithm (IDT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ef} : (e, f) \in E \times E$
Let Opt-Sol be the optimal solution to QBAP and Obj-Val the optimal objective function value to QBAP
Let $\bar{M}$ be the optimal solution to QAP1 and U^* be the optimal objective function value of that, l be such that $z_l = \min\{q_{ef} : (e, f) \in \bar{M} \times \bar{M}\}$. We set u be such that $(z_u \leftarrow U^*)$
Let L^* be such that z_{L^*} is the optimal objective function value of QAP2
Opt-Sol $\leftarrow \bar{M}$
Obj-Val= $z_u - z_l$
$l \leftarrow l + 1$
while $(l \leq u)$ and $(l \leq L^*)$ and $(u \leq p)$ **do**
  Solve AFC
  if AFC is feasible **then**
    Let x^* be the feasible solution of AFC and M^* be its corresponding perfect matching
    $z_r = \min\{q_{ef} : (e, f) \in M^* \times M^*\}$
    $z_t = \max\{q_{ef} : (e, f) \in M^* \times M^*\}$
    if $z_t - z_r <$ Obj-Val **then**
      Obj-Val= $z_t - z_r$
      Opt-Sol= M^*
    end if
    $l = r + 1$
  else
    $u \leftarrow u + 1$
  end if
end while
return Opt-Sol and Opt-Val

However, by using IDT the search interval is smaller than before but we can still add some conditions to IDT to improve the average performance of this algorithm.

4.1.2 The Modified Double Threshold Algorithm for Assignment Problem

Let us now discuss how to improve the average performance of the double threshold algorithm and the improved double threshold algorithm. The improvement is achieved using
1) sufficient condition for optimality and
2) a sufficient condition that allows increments of upper and lower threshold values by larger amounts.

The conditions however does not seem to affect the worst case complexity of the algorithm.

For any feasible solution, let M be the corresponding perfect matching and let $F = \{M^1, M^2, ..., M^p\}$ be the set of all solutions generated by the double threshold algorithm while completing the sequential search, we assume that M^i is generated before M^j for $i < j$. For $1 \leq t \leq p$, let $r_t = max\{q_{ef} : (e, f) \in M^t \times M^t\}$ and $s_t = min\{q_{ef} : (e, f) \in M^t \times M^t\}$. Thus,$r_1 < r_2 < ... < r_p$ and $s_1 < s_2 < ... < s_p$. For $k = 1, ..., p$ choose $t(k)$ such that $r_{t(k)} - s_{t(k)} = min\{r_i - s_i : 1 \leq i \leq k\}$
Let $F^k = \{M^1, ..., M^k\}$.

$M^{t(p)}$ is the best(least cost) solution for QBAP since $r_{t(p)} - s_{t(p)} = \min\{r_i - s_i : 1 \leq i \leq p\}$. Now, let $F^0 \subset F$ such that $F^0 = \{M^i : r_i - s_i = r_{t(p)} - s_{t(p)}\}$. Let β be a real number such that $\beta \geq \max\{M_i \in F : i \in F^0\}$. we set $\beta = \min\{q_{(e,f)} : (e, f) \in \hat{M} \times \hat{M}\}$ to obtain the best possible solution.

Theorem 20. *For any $1 \leq k \leq p$ if $r_{t(k)} - s_{t(k)} + \beta \leq r_k$ then $M^{t(k)} \in F^0$.*

Proof. Suppose $M^{t(k)} \in F^0$ then there is an $M^i \in F^0$ and $i > k$ such that $r_i - s_i < r_{t(k)} - s_{t(k)}$ then $r_i < s_i + r_{t(k)} - s_{t(k)} \leq \beta + r_{t(k)} - s_{t(k)} \leq r_k$. This implies $i \leq k$ which is a contradiction. □

Theorem 20 provides a termination criterion that could reduce the number of iterations for the double threshold algorithm.

Theorem 21. *If M does not belong to F^0 then there is an $M^q \in F^0$ for $q > k$ such that $s_q > r_k - r_{t(k)} + s_{t(k)}$.*

Proof. On contrary, suppose $s_q \leq r_k - r_{t(k)} + s_{t(k)} \leq r_q - r_{t(k)} + s_{t(k)}$ then $r_q - s_q \geq r_{t(k)} + s_{t(k)}$ then $M^{t(k)} \in F^0$ which is a contradiction. □

Theorem 21 could be used to update the lower threshold by larger quantity whenever the condition holds.

Results similar to Theorem 20 and 21 were used by Ahuja [2], Martins[40], Punnen [45] and Punnen and Nair [49] in solving other combinatorial optimization problems. Incorporating the conditions presented in Theorems 20 and 21 into the improved double threshold algorithm, we get the modified double threshold algorithm to solve QBAP. A formal description of the modified double threshold algorithm is given in Algorithm 4.3.

Algorithm 4.3: The Modified Double Threshold Algorithm (MDT)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ef} : (e, f) \in E \times E$
Let Opt-Sol be the optimal solution to QBAP and Obj-Val the optimal objective function value to QBAP
Let $\bar{M}$ be the optimal solution to QAP1 and U^* be the optimal objective function value of that, l be such that $z_l = \min\{q_{ef} : (e, f) \in \bar{M} \times \bar{M}\}$. We set u be such that $(z_u \leftarrow U^*)$
Let L^* be such that z_{L^*} is the optimal objective function value of QAP2
Opt-Sol $\leftarrow \bar{M}$
Obj-Val= $z_u - z_l$
$l \leftarrow l + 1$
while $(l \leq u)$ and $(l \leq L^*)$ and $(u \leq p)$ **do**
  Solve AFC
  if AFC is feasible **then**
    Let x^* be the feasible solution of AFC and M^* be its corresponding perfect matching
    $z_r = \min\{q_{ef} : (e, f) \in M^* \times M^*\}$
    $z_t = \max\{q_{ef} : (e, f) \in M^* \times M^*\}$
    if $z_t - z_r <$ Obj-Val **then**
      Obj-Val= $z_t - z_r$
      Opt-Sol= M^*
    end if
    if Obj-Val $+L^* \leq z_r$ **then**
      return Obj-Val and Opt-Sol
    end if
    choose smallest $q \geq l$ such that $z_q \geq z_u - Obj - Val$
    $l \leftarrow q$
  else
    $u \leftarrow u + 1$
  end if
end while
return Opt-Sol and Opt-Val

4.1.3 The Iterative bottleneck algorithm

Let us now discuss two additional algorithms to solve QBAP. In both cases, we solve a sequence of quadratic bottleneck assignment problems (QAP1 or QAP2) instead of AFC problem. The worst case complexities of these algorithms, in general, are higher than that of the DT algorithm and its variations, the average performance is expected to be better. Recall that

$$\begin{aligned} QAP1: \quad \text{Minimize} \quad & \{\max_{(e,f)\in M\times M} q_{(e,f)}\} \\ \text{Subject to} \quad & \sum_{e\in\Delta(i)} x_e = 1 \quad \forall i \in V \\ & x_e \in \{0,1\} \end{aligned}$$

$$\begin{aligned} QAP2: \quad \text{Maximize} \quad & \{\min_{(e,f)\in M\times M} q_{(e,f)}\} \\ \text{Subject to} \quad & \sum_{e\in\Delta(i)} x_e = 1 \quad \forall i \in V \\ & x_e \in \{0,1\} \end{aligned}$$

QAP1 minimizes $Z_1(Q, M)$ while QAP2 maximizes $Z_2(Q, M)$.

To develop our first iterative bottleneck assignment algorithm, we consider a generalization of QBAP where lower and upper thresholds restriction are imposed on feasible solutions. Let α and β be real numbers such that $\alpha \leq \beta$. Consider the problem

$$\begin{aligned} \text{QBAP(Q,}\alpha,\beta\text{):= Minimize} \quad & Z_1(Q,M) - Z_2(Q,M) \\ \text{Subject to} \quad & \sum_{e\in\Delta(i)} x_e = 1 \quad \forall i \in V \\ & Z_2(Q,M) \geq \alpha \\ & Z_1(Q,M) \leq \beta \\ & x_e = 0,1 \quad \forall i \in E \end{aligned}$$

Consider the cost matrix Q' defined by

$$q'_{ef} = \begin{cases} H & \text{if } q_{ef} < \alpha \text{ or } q_{ef} > \beta \\ q_{ef} & \text{otherwise.} \end{cases}$$

where H is a large number.

Theorem 22. *Let M^0 be an optimal solution to QAP1 with cost matrix Q' and q be the index such that $z_q = Z_2(Q, M^0)$.*
(i) If $Z_1(Q, M^0) = H$ then QBAP(Q,α, β) is infeasible.
(ii) If $Z_1(Q, M^0) < H$ and $Z_1(Q, M^0) = Z_2(Q, M^0)$ then M^0 is an optimal solution to QBAP(Q,α, β).
(iii) If conditions (i) and (ii) above are not satisfied, then either M^0 is an optimal solution to QBAP(Q,α, β) or there exists an optimal solution to QBAP(Q,α, β) which is optimal to QBAP(Q,γ, β) where $\gamma = z_{q+1}$.

Proof. The proof of (i) and (ii) are straightforward. Let us now prove (iii).

Let $F = \{M \in F(\alpha, \beta) : Z_2(Q, M) \leq Z_2(Q, M^0)\}$. By definition of F

$$Z_2(Q, M^0) \geq Z_2(Q, M) \text{ for all } M \in F. \tag{4.1}$$

By optimality of M^0 to QAP1 with cost matrix Q' and condition (i) of the theorem is not satisfied, we have

$$Z_1(Q, M^0) \leq Z_1(Q, M) \text{ for all } M \in F. \tag{4.2}$$

Multiply inequality (4.1) by -1 and adding to inequality (4.2) we have QBAP$(Q, M^0) \leq$ QBAP(Q, M) for all $M \in F$. Thus either M^0 is an optimal solution to QBAP(Q,α, β) or there exists an optimal solution M to QBAP(Q,α, β) satisfying $Z_2(Q, M) > Z_2(Q, M^0)$ and the result follows. □

In view of Theorem 22, we can solve QBAP as a sequence of QAP1 problems. Per iteration, the algorithm maintains an upper threshold β and a lower threshold α and construct a modified cost matrix Q' which depends on the values of α and β. Then using an optimal solution to QAP1 with cost matrix Q', the lower threshold is updated until infeasibility with respect to the threshold values is identified or optimality of an intermediate solution is identified using condition (ii) of Theorem 22. The algorithm compares the solutions generated

by the QAP1 solver and outputs the overall best solution with respect to the QBAP objective function. The resulting algorithm is called the *type 1 iterative bottleneck assignment algorithm* (IBA1-algorithm) and its formal description is given in Algorithm 4.4.

QAP2 can be reformulated as QAP1 or the algorithms for QAP1 can be modified to solve QAP2 directly. Consider the cost matrix $\tilde{Q}$ defined by

$$\tilde{q}_{ef} = \begin{cases} -H & \text{if } q_{ef} > \beta \text{ or } q_{ef} < \alpha \\ q_{ij} & \text{otherwise.} \end{cases}$$

Theorem 23. *Let M^0 be an optimal solution to QAP2 with cost matrix $\tilde{Q}$ and r be the index such that $z_r = Z_1(Q, M^0)$.*
(i) If $Z_2(\tilde{Q}, M^0) = -H$ then QBAP(Q,α, β) is infeasible.
(ii) If $Z_2(\tilde{Q}, M^0) > -H$ and $Z_1(Q, M^0) = Z_2(Q, M^0)$ then M^0 is an optimal solution to QBAP(Q,α, β).
(iii) If conditions (i) and (ii) are not satisfied, then either M^0 is an optimal solution to QBAP(Q,α, β) or there exists an optimal solution to QBAP(Q,α, β) which is also optimal to QBAP(Q,α, γ) where $\gamma = z_{r+1}$.

The proof of this theorem can be constructed by appropriate modifications in the proof of Theorem 22 and hence is omitted. In view of Theorem 23, we can solve QBAP as a sequence of QAP2 problems. Per iteration, the algorithm maintains a lower threshold α and an upper threshold β and construct a modified cost matrix $\tilde{Q}$. Using an optimal solution to QAP2 with cost matrix $\tilde{Q}$, the upper threshold is updated and the process is continued until infeasibility with respect to the threshold values is identified. The algorithm compares the solutions generated by the QKP2 solver in each iteration and outputs the overall best solution. The resulting algorithm is called the *type 2 iterative bottleneck assignment algorithm* (IBA2-algorithm) and its formal description is given in Algorithm 4.5.

Algorithms 4.4 and 4.5 can be viewed as extension of an algorithm by Duin and Volgenant [17] for a generalization of the LBOP.

Algorithm 4.4: The type1 Iterative Bottleneck Assignment Algorithm(IBA1)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ef} : (e, f) \in E \times E$

Let Q be the cost matrix , Opt_Sol be the optimal solution of QBAP and Obj_Val be the objective function value of QBAP in each iteration

$Q' \leftarrow Q$; $Obj\_Val \leftarrow \infty$; $Opt\_Sol \leftarrow \emptyset$, $H \leftarrow 1 + z_p$; $Z_0 = z_p$;

while $(Obj\_Val \neq 0)$ and $Z_0 \neq M$ **do**

  Solve QAP2 with cost matrix Q'. Let M be the resulting solution.

  $Z_0 \leftarrow \max\{q'_{ef} : (e, f) \in M \times M\}$

  if $Z_0 < H$ **then**

    $Z_1 \leftarrow \max\{q_{ef} : (e, f) \in M \times M\}$

    $Z_2 \leftarrow \min\{q_{ef} : (e, f) \in M \times M\}$

    if $Z_1 - Z_2 < Obj - Val$ **then**

      $Opt\_Sol \leftarrow M$

      $Obj\_Val \leftarrow Z_1 - Z_2$

    end if

    Modify costs:

    $$q'_{ef} = \begin{cases} q_{ef} & \text{if } Z_2 < q_{ef} < z_p \\ H & \text{Otherwise} \end{cases}$$

  end if

end while

Return Opt_Sol and Obj_Val

Algorithm 4.5: The type2 Iterative Bottleneck Assignment Algorithm(IBA2)

Let $z_1 < z_2 < ... < z_p$ be an ascending arrangement of distinct values of $q_{ef} : (e, f) \in E \times E$

Let Q be the cost matrix , Opt_Sol be the optimal solution of QBAP and Obj_Val be the objective function value of QBAP in each iteration

$\tilde{Q} \leftarrow Q$; $Obj\_Val \leftarrow \infty$; $Opt\_Sol \leftarrow \emptyset$, $H \leftarrow 1 + z_p$; $Z_0 = z_1$;

while $(Obj\_Val \neq 0)$ and $Z_0 \neq -H$ **do**

  Solve QAP2 with cost matrix $\tilde{Q}$. Let M be the resulting solution.

  $Z_0 \leftarrow \min\{\tilde{q}_{ef} : (e, f) \in M \times M\}$

  if $Z_0 > -H$ **then**

    $Z_1 \leftarrow \max\{q_{ef} : (e, f) \in M \times M\}$

    $Z_2 \leftarrow \min\{q_{e,f} : (e, f) \in M \times M\}$

    if $Z_1 - Z_2 < Obj - Val$ **then**

      $Opt\_Sol \leftarrow M$

      $Obj\_Val \leftarrow Z_1 - Z_2$

    end if

    Modify costs:

    $$\tilde{q}_{ef} = \begin{cases} q_{ij} & \text{if } z_1 < q_{ef} < Z_1 \\ -H & \text{Otherwise} \end{cases}$$

  end if

end while

Return Opt_Sol and Obj_Val

4.2 Polynomially solvable cases for Quadratic Balanced Assignment Problem

Let us now consider some special cases of QBAP that can be solved in polynomial time.

Case 1: Sum Case

We first consider the *decomposable cost matrix* where $q_{ef} = a_e + b_f$ for given $a_e \geq 0$ and $b_f \geq 0$ where $e, f \in E$. Note that

$$\begin{aligned} Z(Q,M) &= \max\{q_{ef} : (e,f) \in M \times M\} - \min\{q_{ef} : (e,f) \in M \times M\} \\ &= \max\{a_e + b_f : (e,f) \in M \times M\} - \min\{a_e + b_f : (e,f) \in M \times M\} \\ &= \max\{a_e : e \in M\} + \max\{b_e : e \in M\} - \min\{a_e : e \in M\} - \min\{b_e : e \in M\} \qquad (4.3) \\ &= \max\{a_e : e \in M\} + \max\{-a_e : e \in <\} + \max\{b_e : e \in M\} - \min\{b_e : e \in M\} \qquad (4.4) \end{aligned}$$

Recall from Chapter 2 that if w_i be a prescribed weight of element $i \in E$ and $g : F \to \Re$. Duin and Volgenant [17] showed that combinatorial optimization problems of the type

$$\begin{aligned} \text{COP}(g): \quad & \text{Minimize } \max\{w_i : i \in S\} + g(S) \\ & \text{Subject to} \\ & \qquad S \in F. \end{aligned}$$

can be solved in $O(m\zeta(m))$ where $\zeta(m)$ is the complexity of minimizing $g(S)$ over F. Note that Here, by considering the special case of assignment problem we have Assignment COP (ACOP):

$$\begin{aligned} ACOP(g) \quad \text{Minimize} \quad & \max\{w_e : e \in M\} + g(M) \\ \text{Subject to} \quad & \sum_{e \in \Delta(i)} x_e = 1 \quad \forall i \in V \\ & x_e \in \{0,1\} \quad \forall e \in E \end{aligned}$$

Let

$$Z(Q,M) = \max\{a_e : e \in M\} + g(M) \qquad (4.5)$$

where $g(M) = \max\{-a_e : e \in M\} + g_1(M)$ and $g_1(M) = \max\{a_e : e \in M\} - \min\{b_e : e \in M\}$. But minimizing $g_1(M)$ over F is precisely the Linear Balanced Assignment Problem (LBAP) . Thus recursively applying the results of Duin and Volgenant [17], ACOP with a decomposable cost matrix can be solved in $O(m^2\eta(m))$ time where $\eta(m)$ is the complexity of an LBAP.

When $a_e = b_e$, then $Z(Q, M) = 2[\max\{a_e : e \in M\} - \min\{a_e : e \in M\}]$. Thus, the problem reduces to LBAP.

Case 2: Product Case

In this case, we assume $q_{ef} = a_e.b_f$ where $a_e \geq 0$ and $b_f \geq 0$ for any $e \in M$ and for any $f \in M$. Then, we have:

$$\begin{aligned} Z(Q, M) &= \max\{q_{ef} : (e, f) \in M \times M\} - \min\{q_{ef} : (e, f) \in M \times M\} \\ &= \max\{a_e.b_f : (e, f) \in M \times M\} - \min\{a_e.b_f : (e, f) \in M \times M\} \\ &= \max\{a_e : e \in M\}.\max\{b_e : e \in M\} - \min\{a_e : e \in M\}.\min\{b_e : e \in M\} \quad (4.6) \end{aligned}$$

Let $g(M) = \max_{e \in M} a_e.\max_{e \in M} b_e$, then by simple modifications in max+sum algorithm in [17] we can solve $g(M)$ and thus QBAP in polynomial time.

4.3 Computational Results

All the algorithms in this chapter were coded in C^{++}, solved by calling Cplex 12.1.0 and executed on the same system as Chapter 3.

There are no benchmark problems for QBAP. Then, we generated random instances for QBAP. For the case of complete bipartite graph there are two parameters for generating the instances, N and PCT where N is the number of vertices of the complete graph and PCT is the percentage of zero elements of cost matrix. PCT is 0,25 and 50 for all instances and N=5,6,7,10,12 since Cplex is unable to solve AFC for instances with $N > 12$. Table 4.1 shows the results for this case.

In the sparse graph case, each instance is including at least a random perfect matching and some additional random edges to produce the sparse graph on which we find the optimal solution to QBAP. There are three parameters for generating these instances. N, M and

PCT, where N is the number of vertices, $M = \log N$ is the number of additional random edges and PCT is the percentage of zero elements in the cost matrix. There are three tables indicating the results of all the algorithms in sparse graph case for different ranges of q_{ij}. PCT is set to 0,25 and 50 for all instances and N=30,40,50,60,70,90,100,200.

The optimal value of all the algorithms and CPU time for each of them is reported in all of the following tables.

Table 4.1: Results on complete graph

Instance	Optimal Value	Cplex	DT	IDT	MDT	IBA2
5-1	111	0.82	0.52	0.65	0.66	0.71
5-2	140	0.53	0.43	0.41	0.47	0.41
5-3	144	0.47	0.33	0.34	0.32	0.35
7-1	159	11.4	1.52	1.60	1.55	1.90
7-2	155	8.16	1.33	1.43	1.24	1.21
7-3	145	1.35	1.35	1.26	1.28	1.26
10-1	165	-	33.15	31.54	29.30	26.75
10-2	167	-	16.50	16.34	13.39	14.23
10-3	142	-	11.70	11.78	10.97	9.28
12-1	168	-	192.44	176.78	165.90	132.67
12-2	150	-	145.21	123.56	124.00	109.76
12-3	147	-	83.65	84.34	83.22	67.50

In summary, by comparing table 4.1 with table 4.2 , 4.3 and 4.4 it is indicated that we can solve noticeably larger instances of QBAP on sparse graph rather than complete graph. Note that for $N = 12$, Q matrix contains $12^4 = 20736$ elements in complete graph case. Tables 4.2, 4.3 and 4.4 show the results for three different random generated data sets with ranges of 300,400 and 500 for q_{ij}. Here, Cplex could solve instances up to size $N = 40$ only, and other algorithms were tested on instances up to size $N = 200$. The computational results in tables 4.2, 4.3 and 4.4 points out that DT performs as the slowest algorithm among all of the algorithms while IDT performs gradually faster than DT. MDT is considerably faster than DT and IDT whereas IBA2 algorithm outperforms other algorithms provided in this chapter.

Table 4.2: Results on sparse graph for $q \leq 300$

Instance	Optimal Value	Cplex	DT	IDT	MDT	IBA2
30-1	291	1.8	0.55	0.03	0.03	0.00
30-2	286	2.38	0.49	0.01	0.01	0.02
30-3	294	0.71	0.37	0.01	0.01	0.04
40-1	292	10.35	1.19	0.03	0.03	0.01
40-2	296	3.62	0.09	0.01	0.02	0.04
40-3	290	3.28	0.66	0.01	0.01	0.06
50-1	294	254.37	2.28	0.07	0.08	0.04
50-2	293	1.63	1.63	0.05	0.05	0.08
50-3	293	17.98	1.17	0.12	0.12	0.08
60-1	293	-	3.84	0.17	0.06	0.02
60-2	295	-	2.81	0.16	0.15	0.12
60-3	291	-	1.87	0.21	0.15	0.15
70-1	294	-	5.67	0.18	0.17	0.10
70-2	295	-	4.39	0.17	0.15	0.12
70-3	291	-	3.2	0.20	0.19	0.06
80-1	295	-	7.73	0.19	0.19	0.16
80-2	295	-	5.94	0.11	0.09	0.06
80-3	293	-	5.00	1.09	1.07	0.70
90-1	295	-	10.22	0.32	0.32	0.14
90-2	297	-	7.63	0.16	0.15	0.31
90-3	294	-	5.28	0.20	0.51	0.18
100-1	296	-	14.05	0.50	0.51	0.21
100-2	296	-	10.94	1.05	1.03	1.00
100-3	295	-	7.25	0.86	0.68	0.65
200-1	297	-	4026.50	3413.30	1336.87	464.62
200-2	298	-	102.58	54.70	53.55	33.61
200-3	297	-	347.71	291.77	290.52	290.98

Table 4.3: Results on sparse graph for $q \leq 400$

Instance	Optimal Value	Cplex	DT	IDT	MDT	IBA2
30-1	382	12.88	0.91	0.08	0.07	0.01
30-2	389	1.81	0.67	0.02	0.01	0.01
30-3	380	1.71	0.64	0.02	0.02	0.03
40-1	386	41.31	1.71	0.09	0.07	0.02
40-2	394	7.86	1.41	0.04	0.04	0.06
40-3	390	4.60	0.91	0.02	0.01	0.07
50-1	391	-	3.21	0.11	0.08	0.04
50-2	387	-	2.53	0.04	0.04	0.09
50-3	389	-	1.76	0.05	0.05	0.07
60-1	391	-	5.32	0.14	0.12	0.06
60-2	390	-	4.06	0.08	0.08	0.13
60-3	385	-	2.89	0.07	0.06	0.11
70-1	394	-	8.42	0.24	0.23	0.10
70-2	393	-	6.37	0.14	0.14	0.12
70-3	387	-	4.42	0.08	0.08	0.03
80-1	394	-	12.65	1.67	1.61	0.63
80-2	393	-	9.36	0.67	0.35	0.34
80-3	389	-	6.55	0.98	0.59	0.56
90-1	394	-	18.43	3.33	4.12	0.14
90-2	393	-	11.58	0.35	0.34	0.34
90-3	394	-	7.96	0.29	0.27	0.42
100-1	394	-	20.16	1.36	1.08	0.98
100-2	394	-	17.24	1.05	1.03	0.20
100-3	398	-	9.31	0.10	0.10	0.46
200-1	397	-	1798.04	353.54	352.56	0.96
200-2	397	-	822.56	778.70	780.55	341.45
200-3	395	-	842.18	391.77	390.52	290.98

Table 4.4: Results on sparse graph for $q \leq 500$

Instance	Optimal Value	Cplex	DT	IDT	MDT	IBA2
30-1	469	4.62	1.03	0.02	0.02	0.01
30-2	472	0.36	0.79	0.02	0.01	0.02
30-3	464	3.37	0.95	0.04	0.03	0.03
40-1	488	9.43	1.93	0.06	0.06	0.02
40-2	486	10.28	1.55	0.04	0.02	0.05
40-3	474	5.04	1.14	0.02	0.02	0.07
50-1	489	-	3.75	0.12	0.12	0.04
50-2	488	-	3.08	0.05	0.05	0.09
50-3	484	-	2.24	0.06	0.06	0.13
60-1	492	-	7.29	0.35	0.25	0.22
60-2	489	-	5.94	0.17	0.18	0.17
60-3	483	-	4.01	0.18	0.16	0.15
70-1	492	-	11.12	0.51	0.52	0.25
70-2	494	-	8.15	0.39	0.33	0.21
70-3	499	-	4.43	0.19	0.10	0.14
80-1	491	-	13.93	1.39	1.21	0.23
80-2	494	-	10.99	0.62	0.29	0.20
80-3	489	-	7.04	0.23	0.23	0.19
90-1	492	-	19.80	1.83	1.74	0.28
90-2	494	-	14.36	0.79	0.29	0.39
90-3	492	-	9.76	0.21	0.22	0.29
100-1	494	-	45.39	1.76	1.72	0.39
100-2	493	-	34.52	0.94	0.94	0.54
100-3	490	-	19.37	0.55	0.53	0.48
200-1	496	-	16976.80	17800.10	6982.57	1924.36
200-2	497	-	1137.01	941.00	301.55	230.45
200-3	494	-	1798.19	1357.50	1190.23	1222.34

Chapter 5

Conclusion

In this thesis, we introduced a new class of combinatorial optimization problems called "Quadratic Balanced Optimization Problems". The problem is shown to be NP-hard even if the family of feasible solution is restricted only by cardinality and the cost elements are just 0/1 . Several general purpose algorithms are proposed. These algorithms also can be used as an efficient heuristic. This is illustrated using the special cases of knapsack problem as perfect matching problem in bipartite graphs. Computational results are also provided. We also identify several polynomially solvable special cases. Improving the complexity of these algorithms and identify new polynomially solvable cases are topics for future research.

Bibliography

[1] V. Aggarwal, V.G. Tikekar, and L.F. Fern Hsu. Bottleneck assignment problems under categorization. *Computers and Operations Research*, 13(1):11–26, 1986.

[2] R.K. Ahuja. Minimum cost to reliability ratio problem. *Computers and Operations Research*, 15(1):83–89, 1988.

[3] R.K. Ahuja. The balanced linear programming problem. *European Journal of Operational Research*, 101(1):29–38, 1997.

[4] H. Albrecher. A note on the asymptotic behaviour of bottleneck problems. *Operations Research Letters*, 33(2):183–186, 2005.

[5] R.D. Armstrong and Z.Y. Jin. Solving linear bottleneck assignment problems via strong spanning trees. *Operations Research Letters*, 12(3):179–180, 1992.

[6] R.E. Burkard, M. Dellamico, and S. Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[7] R.E. Burkard and U. Fincke. On random quadratic bottleneck assignment problems. *Mathematical Programming*, 23(2):227–232, 1982.

[8] R.E. Burkard and R. Rissner. polynomially solvable special cases of the quadratic bottleneck assignment problem. *Journal of Combinatorial Optimization*, 20(1):1–12, 2010.

[9] R.E. Burkard and W. Sandholzer. Efficiently solvable special cases of bottleneck traveling salesman problems. *Discrete Applied Mathematics*, 32(1):61–76, 1991.

[10] M. Camerini, F. Maffioli, S. Martello, and P. Toth. Most and least uniform spanning trees. *Discrete Applied Mathematics*, 15(2-3):181–187, 1986.

[11] P.M. Camerini. The min-max spanning tree problem and some extensions. *Information Processing Letters*, 7(1):10–14, 1978.

[12] P. Cappanera and M.G. Scutella. Balanced paths in acyclic networks: Tractable cases and related approaches. *Networks*, 45(2):104–111, 2005.

[13] G. Carpaneto, S. Martello, and Toth. P. An algorithm for the bottleneck traveling salesman problem. *Operations Research*, 32(2):380–389, 1984.

[14] Y. Dai, H. Imai, K. Iwano, N. Katoh, K. Ohtsuka, and N. Yoshimura. A new unifying heuristic algorithm for the undirected minimum cut problem using minimum range cut algorithms. *Discrete Applied Mathematics*, 65(1):167–190, 1996.

[15] A. Darmann, U. Pferschy, and J. Schauer. Minimal spanning trees with conflict graphs, 2009. `http://www.optimization-online.org/DB_FILE/2009/01/2188.pdf/`.

[16] U. Derigs and U. Zimmermann. An augmenting path method for solving linear bottleneck assignment problems. *Computing*, 19:285–295, 1978.

[17] C.W. Duin and A. Volgenant. Minimum deviation and balanced optimization: A unified approach. *Operations Research Letters*, 10(1):43–48, 1991.

[18] J. Edmonds and D.R. Fulkerson. Bottleneck extrema. *Journal of Combinatorial Theory*, 8(3):299–306, 1970.

[19] D. Eppstein. Minimum range balanced cuts via dynamic subset sums. *Journal of Algorithms*, 23(2):375–385, 1997.

[20] Z. Galil and B. Schieber. On finding most uniform spanning trees. *Discrete Applied Mathematics*, 20(2):173–175, 1988.

[21] H.N. Garbow and R.E. Tarjan. Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9(3):411–417, 1988.

[22] R.S. Garfinkel and K.C. Gilbert. The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. *Journal of the Association for Computing Machinery*, 25(3):435–448, 1978.

[23] R.S. Garfinkel and M.R. Rao. The bottleneck transportation problem. *Naval Research Logistic Quarterly*, 18:465–472, 1971.

[24] S. Geetha and K.P.K. Nair. On stochastic spanning tree problem. *Networks*, 23(8):675–679, 1993.

[25] P.C Gilmore and R.E. Gomory. Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Operations Research*, 12(5):655–679, 1964.

[26] S.K. Gupta and A.P. Punnen. Minimum deviation problems. *Operations Research Letters*, 7(4):201–204, 1988.

[27] S.K. Gupta and A.P. Punnen. Minmax linear knapsack problem with group variables and gub constraints. *Optimization*, 28(1):85–94, 1993.

[28] H. Ishii and T. Nishida. Stochastic bottleneck spanning tree problem. *Networks*, 13(3):443–449, 1983.

[29] H. Ishii and S. Shiode. Chance-constrained bottleneck spanning tree problem. *Annals of Operations Research*, 56(1):177–187, 1995.

[30] M.Y. Kao and M. Sanghi. *An approximation algorithm for a bottleneck traveling salesman problem*, volume 3998 of *Lecture notes in computer science*. 2006.

[31] H. Katagiri, M. Sakawa, and H. Ishi. Fuzzy random bottleneck spanning tree problems using possibility and necessity measures. *European Journal of Operational Research*, 152(1):88–95, 2004.

[32] N. Katoh. An epsilon-approximation scheme for combinatorial problems with variance criterion. *Discrete Applied Mathematics*, 35(2):131–141, 1992.

[33] N. Katoh and K. Iwano. Efficient algorithms for minimum range cut problems. *Networks*, 24(7), 1994.

[34] H. Kellerer and G. Wirsching. Bottleneck quadratic assignment problems and the bandwidth problem. *Asia-Pacific Journal of Operations Research*, 15:169–177, 1998.

[35] J. LaRusic and A.P. Punnen. The balanced traveling salesman problem. *Computers and Operations Research*, 38(5):868–875, 2011.

[36] E.L. Lawler. The quadratic assignment problem. *Management Science*, 9:586–599, 1963.

[37] E.L Lawler, Rinehart Holt, and Winston. *Networks and Matroids*. Dover, 1976.

[38] Z. Lu, F. Glover, and J. Hao. A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research*, 207(3):1254–1262, 2010.

[39] S. Martello, W.R. Pulleyblank, P. Toth, and D. de Werra. Balanced optimization problems. *Operations Research letters*, 3(5):275–278, 1984.

[40] E.Q.V. Martins. An algorithm to determine a path with minimal cost/capacity ratio. *Discrete Applied Mathematics*, 8(2):189–194, 1984.

[41] T. Nemoto. An efficient algorithm for the minimum range ideal problem. *Journal of the Operations Research Society of Japan*, 42(1):88–97, 1999.

[42] R.G. Parker and R.L. Radring. Guaranteed performance heuristics for the bottleneck traveling salesman problem. *Operations Research Letters*, 2(6):269–272, 1984.

[43] U. Pferschy. The random linear bottleneck assignment problem. *Rairo-Recherche Ooerationnelle-Operations Research*, 30(2):127–142, 1996.

[44] J.M. Phillips, A.P. Punnen, and S.N. Kabadi. A linear time algorithm for the bottleneck traveling salesman problem on a halin graph. *Information Processing Letters*, 67(2):105–110, 1998.

[45] A.P. Punnen. On combined minmax-minsum optimization. *Computers and Operations Research*, 21(6):707–716, 1994.

[46] A.P. Punnen. A fast algorithm for a class of bottleneck problems. *Computing*, 56(4):397–401, 1996.

[47] A.P. Punnen and Y.P. Aneja. Lexicographic balanced optimization problems. *Operations Research Letters*, 32(1):27–30, 2004.

[48] A.P. Punnen and K.P.K Nair. Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. *Discrete Applied Mathmatics*, 55(1):91–93, 1994.

[49] A.P. Punnen and K.P.K Nair. An improved algorithm for the constrained bottleneck spanning tree problem. *INFORMS Journal on Computing*, 8(1):41–44, 1996.

[50] A.P. Punnen and R. Zhang. Quadratic bottleneck problems. *Naval Research Logistics*, 58(2):153–164, 2011.

[51] R. Ramakrishnan, P. Sharma, and A.P. Punnen. An efficient heuristic algorithm for the bottleneck traveling salesman problem. *OPSEARCH*, 46(3):275–288, 2009.

[52] A. Ravindran and V. Ramaswami. On the bottleneck assignment problem. *Journal of optimization theory and applications*, 21(4):451–458, 1977.

[53] A. Shioura and M. Shigeno. The tree center problems and the relationship with the bottleneck knapsack problems. *Networks*, 29(2):107–110, 1997.

[54] M.Z. Spivey. Asymptotic moments of the bottleneck assignment problem. *Mathematics of Operations Research*, 36(2):205–226, 2011.

[55] L. Steinberg. The backboard wiring problem: A placement algorithm. *SIAM Review*, 3:37–50, 1961.

[56] S. Tigan, E. Iacob, and I.M. Stancu-Minasian. Monotonic balanced optimization problems. In *Annals of the Tiberiu Popoviciu Itinerant Seminar of Functional Equations, Approximation and Convexity*, volume 3, pages 183–197, 2005.

[57] Pferschy. U. Solution methods and computational investigations for the linear bottleneck assignment problem. *Computing*, 59(3):237–258, 1997.

[58] G.L. Vairaktarakis. On gilmore-gomory's open question for the bottleneck tsp. *Operations Research Letters*, 31(6):483–491, 2003.

[59] U. Yechiali. Stochastic bottleneck assignment problem. *Management Scince*, 14(11):732–734, 1968.

[60] Z. Zeitlin. Minimization of maximum absolute deviation in integers. *Discrete Applied Mathematics*, 3(3), 1981.

[61] R. Zhang, S.N. Kabadi, and A.P. Punnen. The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191–205, 2011.

[62] R. Zhang and A.P. Punnen. Quadratic bottleneck knapsack problem. *Forthcomming in Journal of Heuristics*, 2011. DOI: 10.1007/s10732-011-9175-1.