QUADRATIC BOTTLENECK PROBLEMS: ALGORITHMS, COMPLEXITY AND RELATED TOPICS

by

Ruonan Zhang

B.Sc., Lanzhou University, 2003

M.Sc., University of New Brunswick, 2005

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
IN THE
DEPARTMENT OF MATHEMATICS
FACULTY OF SCIENCE

SIMON FRASER UNIVERSITY
Fall 2011

APPROVAL

Name: Ruonan Zhang

Degree: Doctor of Philosophy

Title of Thesis: Quadratic Bottleneck Problems: Algorithms, Complexity and Related Topics

Examining Committee: Dr. Tamon Stephen
Assistant Professor, Chair

Dr. Abraham Punnen
Professor, Senior Supervisor

Dr. Zhaosong Lu
Assistant Professor, Supervisor

Dr. Snezana Mitrovic-minic
Adjunct Professor, Internal Examiner

Dr. Donglei Du
Professor, Faculty of Business Administration,
University of New Brunswick, External Examiner

Date Approved: October 20, 2011

Partial Copyright Licence

Abstract

In this thesis we study the quadratic bottleneck combinatorial optimization problem (QBCOP) which generalizes the well-known class of bottleneck problems. The solvability of QBCOP is linked to that of the linear combinatorial optimization problems with conflict constraints. Various properties and algorithms for these classes of problems are developed and special cases of spanning trees, knapsack type problems, and assignment variations are explored. These problems are shown to be strongly NP-hard even on very special graphs. We then identify polynomially solvable special cases and also develop heuristic algorithms. Experimental results are reported for all the heuristics we developed. As a by-product our work, we have an approximation algorithm for the maximum edge clique partitioning problem, improving the best known performance ratio for the problem.

Acknowledgments

I express my first and deepest thankfulness to Dr. Abraham Punnen - my senior supervisor. Throughout my studies, he has consistently guided my work with great inspiration, enthusiasm and patience. Absolutely, I would not be able to accomplish this work without him. To me, Abraham is far beyond a resource of knowledge, a supervisor who provide critics, suggestions, and corrections to my work, but he is the one who has shaped my character, my ethic, and my courage on tackling the incoming difficulties. I am so gratitude for being his student and to learn from him on both the scientific and personal levels is surely a life-long process.

My special thanks to paper coauthors - Dr. Santosh Kabadi and Dr. Öncan Temel for their great contributions, it has been a great experience working with and learn from them.

Then I would like to thank my supervisory committee member Dr. Zhaosong Lu for his feedback and helpful comments. Also acknowledge helps and encouragements provided by Dr. Tamon Stephen, Dr. Natalia Kouzniak, Dr. Randall Pyke in Department of Mathematics, SFU Surrey.

I owe thanks to my colleagues in the SFU Optimization Graduate Student Lab, who have generously shared with me their experience to help my work. I would also like to thank everyone at SFU who helped me in the past six years, specially Department of Mathematics.

Last but not least, I would like to thank my family and friends. Without their unconditional love and support, I would never be who I am.

I dedicate this work to my parents - Du, Li and Zhang, Jiansheng.

Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Meaning	Page
COP	combinatorial optimization problem	1
MST	minimum spanning tree problem	1
TSP	traveling salesman problem	1
AP	assignment problem	2
QAP	quadratic assignment problem	2
LCOP	linear combinatorial optimization problem	2
KP	knapsack problem	2
LCOPC	LCOP with conflict constraints	3
QCOP	quadratic combinatorial optimization problem	4
UQP	unconstrained quadratic problem	5
QAP	quadratic assignment problem	6
QAP-L	Lawler formatted QAP	6
QAP-KB	Koopmans-Beckmann formatted QAP	6
QTSP	quadratic traveling salesman problem	8
QMST	quadratic minimum spanning tree problem	8
AQMST	adjacent-only quadratic minimum spanning tree problem	9
QKP	quadratic knapsack problem	10
BCOP	bottleneck combinatorial optimization problem	13
BSTP	bottleneck spanning tree problem	13
BAP	bottleneck assignment problem	13
BTSP	bottleneck traveling salesman problem	13
BKP	bottleneck knapsack problem	13
FBCOP	BCOP feasibility problem	15

QBCOP	quadratic bottleneck combinatorial optimization problem	17
FQBCOP	QBCOP feasibility problem	18
QBAP	quadratic bottleneck assignment problem	18
BMP	bandwidth minimization problem	19
QBAP-KB	Koopmans-Beckmann formatted QBAP	19
BOP	balanced optimization problem	19
2-sum OP	2-sum optimization problem	19
QBST	quadratic bottleneck spanning tree problem	20
QBKP	quadratic bottleneck knapsack porblem	20
MSTC	minimum spanning tree with conflict constraints	20
APC	assignment problem with conflict constraints	20
Max-ECP	maximum edge clique partitioning problem	20
Max-EBCP	maximum edge bi-clique partitioning problem	60
CQBCOP	cardinality constrained QBCOP	69
FQBST	QBST feasibility problem	83
HPP	Hamiltonian path problem	84
BUQP	bottleneck unconstrained 0-1 quadratic programming problem	97
FQBKP	QBKP feasibility problem	100
MWIP	maximum weight independent set problem	100
FQBAP	QBAP feasibility problem	115

Chapter 1

Combinatorial Optimization Problems

1.1 Introduction

Given a finite set E of cardinality m, a family $\mathcal{F}$ of subsets of E, and a real valued function $\Phi : 2^E \to \mathbb{R}$, a *combinatorial optimization problem* (COP) is formulated as follows:

$$\begin{aligned} &\text{Minimize } \Phi(S) \\ &\text{Subject to} \\ &\qquad S \in \mathcal{F}. \end{aligned}$$

The elements of $\mathcal{F}$ are called *feasible solutions*. The solvability of the COP depends on the structure of the family of feasible solutions $\mathcal{F}$ and the nature of the objective function Φ. For example, when E is the edge set of a graph G, c_e is a given cost for $e \in E$, $\mathcal{F}$ is the family of spanning trees of G and $\Phi(S) = \sum_{e \in S} c_e$ for any $S \in \mathcal{F}$, the the COP reduces to *the minimum spanning tree problem (MST)* [88, 113, 164, 200, 224]. In the MST, if $\mathcal{F}$ is replaced by the family of Hamiltonian tours in G, we get *the traveling salesman problem (TSP)* [120]. It is well known that the MST is polynomially solvable, whereas TSP is strongly NP-hard. Furthermore, if G is a bipartite graph with edge set E and we redefine $\mathcal{F}$ as the family of perfect matchings in G, the resulting problem is recognized as *the assignment problem*

(AP) and is polynomially solvable [40]. However, by introducing a quadratic cost $q(e, f)$ for every $(e, f) \in E \times E$ in the AP and letting $f(S) = \sum_{e \in S} \sum_{f \in S} q(e, f) + \sum_{e \in S} c_e$, we obtain *the quadratic assignment problem (QAP)*, which is strongly NP-hard and cannot even be approximated in polynomial time [101, 169, 213].

So, a general question that arises would be *"For what combinations of $\mathcal{F}$ and Φ the COP (or its special cases) is NP-hard, polynomially approximatable or polynomially solvable?"* In this thesis, we consider several combinatorial optimization problems, including both existing problems and newly-defined ones, and conduct a thorough study in terms of computational complexity and practical exact and heuristic algorithms. Various polynomially solvable special cases are identified for problems that are NP-hard. Different heuristics are developed and experimental results are reported. In some cases, the quality of the heuristic solutions are identified by establishing theoretical performance ratio. We hope that the study will bring more insights into the COP and provide some significant directions for future research.

1.2 Linear Combinatorial Optimization Problems (LCOP)

Let c_e be a prescribed cost associated with element $e \in E$. In COP, if $\Phi(S) = \sum_{e \in S} c_e$, we get *the linear combinatorial optimization problem (LCOP)*. The LCOP is usually specialized into different problems depending on the realizations of E and $\mathcal{F}$. For example, if E is the edge set of an undirected graph G and $\mathcal{F}$ is the family of all spanning trees on G, the LCOP reduces to the well-known minimum spanning tree problem (MST) . If E is the edge set of a balanced bipartite graph G and $\mathcal{F}$ is the family of perfect matchings in G, then the LCOP becomes the assignment problem (AP). If E is the edge set of an undirected graph G and $\mathcal{F}$ is the family of all Hamiltonian tours of G, then the LCOP is realized as the traveling salesman problem (TSP) . To consider examples outside the graph theory structure, let E be a set with m elements and a weight $w(x)$ exists for every element $x \in E$. $c > 0$ be a given capacity and $\mathcal{F} = \{S : S \subset E, \sum_{x \in S} w(x) \geq c\}$, then the resulting LCOP is identified as *the knapsack problem (KP)* [95, 156, 175]. MST, AP, TSP and KP are all classic combinatorial optimization problems which have been thoroughly studied in the literature. Their applications include network design, vehicle routing, signal processing, resource allocation, capital investment selection, scheduling, DNA sequencing and so on. It has been proved that the MST and AP can be solved in polynomial time, precisely, the best known algorithm for the AP has a complexity of $O(n^3)$ [40] when G is a complete bipartite

graph and the MST has the worst case complexity of $O(m \log \beta(m,n))$, where n is the number of nodes, m is the number of edges in the graph and $\beta(m,n) = \min\{i : \log^{(i)} n \leq m/n\}$. The TSP and KP however are NP-hard [95]. In fact, TSP is strongly NP-hard whereas KP can be solved by a pseudo polynomial time algorithm with complexity $O(mc)$, where c is the capacity. By calling this pseudo polynomial time algorithm as a subroutine, under a scaling framework, a fully polynomial approximation scheme(FPTAS) for the KP can be developed.

1.2.1 LCOP with conflict constraints

Let $P \subseteq \{\{e,f\} : e \in E, f \in E, e \neq f\}$ be a given set and we add a class of constraints to the LCOP such that for any $\{e,f\} \in P$, at most one of e, f can be included in a feasible solution S, then the resulting problem is called *the linear combinatorial optimization problem with conflict constraints (LCOPC)* [62, 235]. The elements of P are called *conflict pairs*. Mathematically, the LCOPC can be formulated as follows:

$$\text{Minimize } \sum_{e \in S} c_e$$

$$\text{Subject to}$$

$$S \in \mathcal{F}$$

$$\{e,f\} \nsubseteq S \text{ for } \forall \{e,f\} \in P.$$

Given a *LCOPC*, a *conflict graph* can be constructed as an undirected graph containing m nodes $n_1, n_2, \cdots, n_m$, each corresponding to elements in E. There is an edge (n_i, n_j) in the conflict graph if $\{i,j\}$ is a conflict pair in the LCOPC.

The LCOPC can be used to model the LCOP applications where incompatibilities exist between some pairs of elements. For instance, certain pairs of connections cannot be involved in a network, or some combinations of two investments are not provided in a portfolio, etc.

Adding the conflict constraints to the LCOP affects the complexities of the original problems. In fact Darmann et al. [62] showed that the minimum spanning tree problem on a general graph with conflict constraints is strongly NP-hard, even if in the underlying conflict graph is a 3-ladder, i.e. an undirected graph whose connected components are paths of 2 edges. When the conflict graph is a 2-ladder, i.e. consisting of only single edges as connected components, the problem is polynomially solvable. Further, they showed that

the minimum cost perfect matching problem with conflict constraints on a general graph is strongly NP-hard even if the conflict graph is a 2-ladder. The maximization version of the knapsack problem with conflict constraints has been studied by Yamada et al. [234], Hifi and Michrafy [134] and Pferschy and Schauer [196]. The problem is strongly NP hard in general, while pseudopolynomial algorithms exist when the conflict graph is a tree, a graph with bounded treewidth or a chordal graph [196]. Other cases of the LCOPC are considered in the literature for bin packing, scheduling and the shortest path [26, 27, 74, 139, 63].

In Chapters 2 and 7 of this thesis, the minimum spanning tree and assignment problem with conflict constraints are investigated, with both theoretical and computational results.

1.3 Quadratic Combinatorial Optimization Problems (QCOP)

Let us consider a generalization of the LCOP by introducing quadratic costs. In the COP, let $q(e, f)$ be a prescribed cost for each $(e, f) \in E \times E$ and $\Phi(S) = \sum_{e \in S} \sum_{f \in S} q(e, f)$, then we obtain *the quadratic combinatorial optimization problem (QCOP)*, which can be formulated as follows:

$$\text{Minimize} \sum_{e \in S} \sum_{f \in S} q(e, f)$$

$$\text{Subject to}$$

$$S \in \mathcal{F}$$

It is easy to see that if $q(e, f) = 0$ for $e \neq f$ and $q(e, e) = c_e$, then the QCOP reduces to an LCOP. Let Q be an $m \times m$ matrix where the (i, j)-th entry has value $q(i, j)$, then Q is called *the cost matrix* associated with the QCOP.

The LCOPC mentioned in Section 1.2.1 is also closely related to the QCOP.

Lemma 1. *A LCOPC can be formulated as a QCOP.*

Proof. Given a LCOPC defined on E with costs c_e, conflict set P and family of feasible solutions $\mathcal{F}$, we construct a QCOP on E with the same family of feasible solutions $\mathcal{F}$. Let

$$q(e,f) = \begin{cases} c_e & \text{if } e = f \\ M & \text{if } \{e,f\} \in P \\ 0 & \text{if } \{e,f\} \notin P, \end{cases}$$

where $M \gg \max\{\sum_{e \in S} c_e : S \in \mathcal{F}\}$. Then if the optimal objective function value of this QCOP is greater than or equal to M, the LCOPC is infeasible. Otherwise, an optimal solution to the QCOP is also an optimal solution to the LCOPC. □

The unconstrained 0-1 quadratic programming problem (UQP), the quadratic assignment problem (QAP), the quadratic minimum spanning tree problem(QMST), the quadratic knapsack problem (QKP) are well studied special cases of the QCOP.

1.3.1 The Unconstrained 0-1 Quadratic Programming Problem (UQP)

The unconstrained 0-1 quadratic programming problem (UQP) is probably the QCOP special case with the simplest requirement. Here, any subset of E serves as a feasible solution and hence a feasible solution can be identified easily. The UQP can be defined as:

$$UQP: \text{Minimize} \sum_{e \in S} \sum_{f \in S} q(e,f)$$
$$\text{Subject to}$$
$$S \subseteq E.$$

This problem was introduced by Hammer and Rudeanu [129] in 1968 and has been intensively studied since then. Considered as a common model, the UQP has been used for a broad range of discrete optimization problems covering VLSI design [17, 29, 30, 142, 162], statistical mechanics [17, 197], economics and finance [128, 167, 168, 180], and management science [92, 129, 199, 229, 230], etc. Moreover, many combinatorial optimization problems have arisen from graph theory, for example the maximum cliques, maximum cuts, maximum vertex packing, minimum coverings, maximum independent sets, and maximum independent weighted sets can be formulated as a UQP [129, 192, 198].

Even though its constraint set is "trivial", the UQP is in fact NP-hard [129]. There are many exact algorithms proposed for the UQP [18, 25, 116, 131, 148, 191] but by considering

the computational effort, solving problems that have larger than 100 variables remains a big challenge. Thus, to handle large UQP instances, various heuristics have been developed such as simulated annealing [153], tabu search [107, 109], genetic algorithms [181], scatter search [7] and so on.

We use a heuristic algorithm of the maximization version of the UQP to develop a heuristic for quadratic bottleneck knapsack problem in Chapter 6.

1.3.2 The Quadratic Assignment Problem (QAP)

Given $N = \{1, 2, \cdots, n\}$, let $\mathcal{P}_n$ be the family of all permutations of N and for each quadruplet $(i, j, k, l) \in N \times N \times N \times N$, a cost q_{ijkl} is prescribed. Then *the quadratic assignment problem* is defined as follows:

$$\begin{aligned} &\text{Minimize } \sum_{i \in N} \sum_{j \in N} q_{i\pi(i)j\pi(j)} \\ &\text{Subject to} \\ &\qquad \pi \in \mathcal{P}_n. \end{aligned}$$

The above formulation was introduced by Lawler [169] and is denoted by *QAP-L*. If we construct a complete bipartite graph $G = (V_1, V_2, E)$, where $|V_1| = |V_2| = n$, $|E| = m = n^2$, and let $\mathcal{F}$ be the set of all perfect matchings of G, then the QAP is equivalent to

$$\begin{aligned} &\text{Minimize } \sum_{e \in S} \sum_{f \in S} q(e, f) \\ &\text{Subject to} \\ &\qquad S \ \in \ \mathcal{F}, \end{aligned}$$

which indicates that it is a special case of the QCOP. There is another well known QAP definition - *the Koopmans-Beckmann QAP (QAP-KB)* [161], which is a special case of the QAP-L that was introduced earlier in the literature. Assume there exist three $n \times n$ matrices $A = (a_{ij})_{n \times n}$, $B = (b_{ij})_{n \times n}$ and $D = (d_{ij})_{n \times n}$, the QAP-KB is to:

$$\text{Minimize} \sum_{i \in N} \sum_{\substack{j \in N \\ j \neq i}} a_{ij} b_{\pi(i)\pi(j)} + \sum_{i \in N} d_{i\pi(i)}$$

Subject to

$$\pi \in \mathcal{P}_n$$

QAP-KB was initially studied in the context of facility location problems [161], with the assigning of n facilities to n locations. The cost was evaluated as a function of the distance and flow between the facilities, plus costs associated with a facility being placed at a certain location. In the above QAP-KB formulation, we can consider a_{ij} as the flow between the facility i and j ($i \neq j$), b_{kl} as the distance between the location k and l ($k \neq l$), and d_{ik} as the price of placing facility i at location k. The objective is to assign each facility to a unique location to minimize the total cost.

Since QAP-L is more general than QAP-KB, hereafter in this thesis, we refer QAP to QAP-L. Besides facility location, QAP has a wide range of applications in backboard wiring, computer manufacturing, scheduling, process communications, turbine balancing, etc. [42, 99, 166, 219, 226]

Among all of the special cases of the QCOP, QAP is perhaps the most intensively studied problem. Research work on QAP has mainly focused on the complexities, polynomially solvable special cases, lower bounds, exact algorithms and heuristics. It has been shown that QAP is strongly NP-hard and for an arbitrary $\epsilon > 0$, the existence of a polynomial time ϵ-approximation algorithm for QAP-KB implies $P = NP$ [213]. Queyranne [208] further proved that, unless $P = NP$, there is no polynomial time heuristic for QAP-KB satisfying the triangle inequality with a bounded asymptotic performance ratio. Also with certain types of neighborhood, the QAP has been proven to be PLS-complete [140]. Several types of lower bounds have been derived, including the Gilmore-Lawler type lower bounds [56, 59, 84, 101, 169], linear programming relaxation based lower bounds [2, 124, 150], variance reduction lower bounds [170], eigenvalue based lower bounds [82, 122, 121, 123, 210, 211], semi-definite relaxation based lower bounds [149, 237, 238] and improved lower bounds by decomposition [53, 151]. Exact algorithms based on the branch-and-bound method [20, 33, 35, 57, 98, 101, 165, 169, 178, 179, 183, 187, 189, 193], traditional cutting plane

methods [12, 13, 14, 21, 22, 36, 154], and polyhedral cutting planes [147, 188] have been obtained and tested. Nevertheless, solving the QAP with size $n \geq 30$ by an exact algorithm is still a challenge. On the other hand, numerous heuristic algorithms have been proposed such as construction methods [9, 101, 185], limited enumeration methods [37], improvement methods [1, 85], tabu search [19, 52, 105, 104, 216, 222], simulated annealing [13, 50, 58, 232], genetic algorithm [136, 225], greedy randomized adaptive search procedure [78, 190], ant systems [69, 173, 93], etc.

Let $\mathcal{T}_n$ be the family of all the cyclic permutations (tours) on N. In the definition of the QAP, if we replace $\mathcal{P}_n$ by $\mathcal{T}_n$ we get *the quadratic traveling salesman problem (QTSP)*. More precisely, QTSP can be defined as

$$\text{Minimize} \sum_{i \in N} \sum_{j \in N} q_{i\pi(i)j\pi(j)}$$

$$\text{Subject to}$$

$$\pi \in \mathcal{T}_n$$

To the best of our knowledge, the QTSP has not been studied in the literature, except for a recent work by A. Fischer and C. Helmberg [83], which studied the adjacent-only QTSP. The authors showed that the adjacent-only QTSP can be used to solve some problems such as the Angular-Metric TSP [3], TSP with reload costs [5, 90, 112, 233], etc. The polyhedral structure of an integer linear programming formulation of the problem is explored.

1.3.3 The Quadratic Minimum Spanning Tree Problem (QMST)

Let $G = (V, E)$ be an undirected graph where $|V| = n,\ |E| = m$. For every $e \in E$, a cost c_e is given and for every $e, f \in E,\ e \neq f$, a quadratic cost $q(e, f)$ is prescribed. Then *the quadratic minimum spanning tree problem (QMST)* is:

$$\text{Minimize } z(T) = \sum_{e \in T} \sum_{\substack{f \in T \\ f \neq e}} q(e,f) + \sum_{e \in T} c_e$$

Subject to

$$T \in \mathcal{F},$$

where $\mathcal{F}$ is the family of spanning trees of G.

By letting $q(e,e) = c_e$, the QMST objective function becomes $\sum_{e \in S} \sum_{f \in S} q(e,f)$, so that the QMST formulation is consistent with that of the general QCOP.

An example of an application of the QMST is in modeling a pipeline communication problem. To design a pipeline network to connect a number of locations, a feasible plan can be represented as a spanning tree in an undirected graph, whose vertices correspond to the locations and the edges represent the connections between the locations. Assume the costs of transmitting depend on the nature of the interactions between every two pipe lines, and the objective is to minimize the total cost.

The QMST can also be applied to telecommunications, transportation, irrigation, energy distribution, etc. It also can easily be shown that the QMST is a generalization of the traveling salesman problem, quadratic assignment problem, maximum clique problem, so that applications of these models are also relevant to the QMST.

In a QMST, if $q(e,f)$ is considered only when e and f are adjacent, then it is called *the Adjacent-only Quadratic Minimum Spanning Tree Problem (AQMST)*.

QMST and AQMST were introduced by Assad and Xu [11]. They showed that both problems are NP-hard on a general graph with arbitrary cost matrix. A lower bounding procedure was derived and incorporated into a branch-and-bound algorithm, which solved problems with up to 15 nodes to optimality. Two heuristic algorithms were also given, along with preliminary computational results. The problem was further studied by various authors. Zhou and Gen [240, 239] proposed a heuristic based on a genetic algorithm using the Pr*ü*fer number encoding. The algorithm was tested on 17 randomly generated instances with 6 to 50 nodes and the best solutions obtained by the heuristics by Assad and Xu can be improved by on average 9.6% and at most 12.83%. A variation of QMST called *fuzzy QMST*(FQMST) was studied by Gao and Lu [94]. They formulated the problem as an expected value model with chance-constrained programming and dependent-chance

constrained programming and a simulation based genetic algorithm was proposed. Soak et al. [217, 218] devised evolutionary algorithms using various encoding methods and reported that, for the QMST case, the evolutionary algorithm using adaptive learning technique outperformed many other encoding strategies, including the Prüfer encoding. Recently, Temel and Punnen [64] developed a Lagrangian relaxation procedure and a local search algorithm with tabu thresholding. Their algorithms were tested on standard instances, random generated instances, and quadratic assignment problem instances from the QABLIB by using a transformation scheme. The Lagrangian relaxation scheme yielded the tightest lower bounds for all instances when compared to previous bounding approaches and the performance of the local search algorithm was also very encouraging.

1.3.4 The Quadratic Knapsack Problem (QKP)

Let $E = \{1, 2, \cdots, m\}$ be a finite set where each element j has a positive weight w_j. For each $(i, j) \in E \times E$, a cost $q(i, j)$ is also prescribed. $c > 0$ is a given capacity. Then, the QKP is defined as:

$$\text{Minimize } \sum_{i=1}^{m}\sum_{j=1}^{m} q(i,j)x_i x_j$$

$$\text{Subject to}$$

$$\sum_{j=1^m} w_j x_j \geq c$$

$$x_j = 0 \text{ or } 1 \text{ for } j = 1, 2, \cdots, m.$$

If we use $\mathcal{F}$ to denote the family of subsets of E such that $\mathcal{F} = \{S : S \subset E, \sum_{j \in S} w_j \geq c\}$, then the above formulation can be restated as

$$\text{Minimize } \sum_{i \in S}\sum_{j \in S} q(i,j)$$

$$\text{Subject to}$$

$$S \in \mathcal{F},$$

which fits the format of the QCOP.

As an example of the applications, let us consider a selection among n types of investments for a portfolio. Let $q(i,j)$ be a measure of the combined risk for choosing both investments i and j. For example, investment i with hedging j reduced the combined risk. Clearly, $q(i,j)$ is selected as zero when i and j are independent. Let w_j be the expected return on investment j. The objective is to minimize the total risk while the overall expected return is no less than a prescribed threshold c. This problem can be formulated as a QKP.

It is easy to see that QKP is a NP-hard problem as it generalizes the knapsack problem and the latter is NP-complete. Even though the QKP, as formulated above, is seldom mentioned in the literature, one of its variations, which we refer to here as QKP_max, is very well-known. QKP_max is formulated as follows:

$$\text{Maximize } \sum_{i=1}^{m}\sum_{j=1}^{m} q(i,j)x_i x_j$$

$$\text{Subject to}$$

$$\sum_{j=1^m} w_j x_j \leq c$$

$$x_j = 0 \text{ or } 1 \text{ for } j = 1, 2, \cdots, m.$$

This problem was introduced by Gallo et al. [92] in 1980. Again as a generalization of the knapsack problem (maximization version), QKP_max is NP-hard and it has been proven that when $q(i,j)$ take both positive and negative values, QKP_max does not have any polynomial time approximation algorithm with a constant ratio unless P=NP [209]. When $q(i,j)$ are non-negative, this is still an open problem. Given the difficulty of the problem, existing results are focused on deriving high quality upper bounds including relaxation from upper planes, linearization, reformulation, Lagrangian relaxation, Lagrangian decomposition and semidefinite programming [132]. Branch and bound algorithms using these upper bounds have been developed [24, 46, 92, 51, 127, 132] though it remains a challenge to solve dense QKP instances with up to 400 variables. Heuristic algorithms and variable reduction techniques have also been investigated in many research papers [24, 46, 92, 108, 126, 194].

1.3.5 Polynomially solvable cases

As seen in this section, many instances of the quadratic combinatorial optimization problems are strongly NP-hard, including those whose linear counter-parts are polynomially solvable. Therefore, while exploring the NP-hard QCOP special cases, we are also interested in developing the knowledge on polynomially solvable QCOP. One such case is *the QCOP with a linearizable cost matrix*. Given a QCOP defined on $E = \{1, 2, \cdots, m\}$ with cost matrix $Q = (q(e, f))$, if there exists a vector $d = (d_1, d_2, \cdots, d_m)$ such that for any $S \in \mathcal{F}$,

$$\sum_{e \in S} \sum_{f \in S} q(e, f) = \sum_{e \in S} d_e,$$

then we call Q *a linearizable matrix* and d *a linearization* of Q. Since the QCOP with a linearizable cost matrix Q reduces to a LCOP with costs d_e, it can be solved in polynomial time whenever the corresponding LCOP is polynomially solvable, and *the QCOP linearization problem* is defined as *"Given a QCOP with cost matrix Q, is Q linearizable? If yes, how do we find its linearization d?"*

The QAP linearization problem has been studied by several researchers [28, 40, 49, 75] and several polynomially testable sufficient conditions have been identified. A necessary and sufficient condition for an instance of the QAP to be linearizable was derived by Kabadi and Punnen [146], who came up with an $O(n^4)$ time algorithm to solve the linearization problem. For QAP-KB and the multiplicative QAP [54], the algorithm takes $O(n^2)$ time and the bounds are shown to be the best possible. The linearization problem for the QTSP has been considered [146] and a necessary and sufficient condition for its linearizability has been established, along with an $O(n^6)$ algorithm for the QTSP linearization problem.

For the QMST linearization problem, the following result can be easily established:

Lemma 2. *Given an $m \times m$ matrix $Q = (q(e, f))$, if there exist $\{a_e : e = 1, \cdots, m\}$, $\{b_e : e = 1, \cdots, m\}$ such that $q(e, f) = a_e + b_f$ for $e, f = 1, \cdots, m$, then Q is linearizable.*

Proof. Consider a QMST on G with cost matrix Q, take any spanning tree T on G,

$$\begin{aligned}\sum_{e \in T} \sum_{f \in T} q(e, f) &= \sum_{e \in T} \sum_{f \in T} (a_e + b_f) = \sum_{e \in T} \sum_{f \in T} a_e + \sum_{e \in T} \sum_{f \in T} b_f \\ &= (n-1) \sum_{e \in T} (a_e + b_e).\end{aligned}$$

Therefore $d = (a_1 + b_1, \cdots, a_m + b_m)$ is a linearization of Q. □

There are other polynomially solvable cases identified for the QAP, either by imposing structural conditions to the cost matrices, or investigating polynomially solvable cases of other combinatorial optimization problems, which can be formulated as QAPs. For example, the maximization version of the QAP with a Anti-Monge-Toeplitz cost matrix [38], the QAP with a product cost matrix or a chess-board cost matrix [49], the QAP-KB with cost matrices A and B, where A is a Kalmanson matrix and B is a Toeplitz matrix with additional properties [65].

1.4 Bottleneck Combinatorial Optimization Problems (BCOP)

In the COP, if $\Phi(S) = \max_{e \in S} c_e$, we obtain *the Linear Bottleneck Combinatorial Optimization Problem* (BCOP), which is formally stated as:

$$\begin{aligned} &\text{Minimize } \max_{x \in S} f(x) \\ &\text{Subject to} \\ &\qquad S \in \mathcal{F}. \end{aligned}$$

The BCOP is another basic type of a combinatorial optimization problem. Various special cases of the model, such as *the bottleneck spanning tree problem (BSTP)*[89], *the bottleneck assignment problem (BAP)* [67, 86, 89, 96, 195, 203], *the bottleneck traveling salesman problem (BTSP)* [97, 102, 120], *the bottleneck knapsack problem (BKP)* [176, 119], etc. are very well studied problems. Some applications of the BCOP include job assignment on parallel-working machines [86], workforce planning [40], locating objects in space, etc. Moreover, most of the applications of the LCOP have a natural interpretation in the context of the BCOP, since the LCOP objective function measures the "average" behavior while the BCOP objective function can be viewed as a measure of the "worst-case" behavior.

For the BSTP, it has been shown that every optimal solution for the MST is also optimal for the BSTP [89], so all algorithms for the MST can be used to solve the BSTP. The best known algorithm for the BSTP on an undirected graph has a complexity of $O(m)$ [89], which uses a binary search with graph contraction.For a directed graph, this complexity is $O(\min(m + n \log n, m \log^* n))$ [89], where $\log^* n$ is the iterative logarithm of n. For the

BAP, Garbow and Tarjan [89] derived an $O(m\sqrt{n \log n})$ algorithm and for dense graphs, this bound is further improved by Punnen and Nair [203] to $O(n^{1.5}\sqrt{m})$. The BTSP is NP-hard, while BKP can be solved in linear time [119].

This thesis primarily focuses on bottleneck quadratic combinatorial optimization problem (BQCOP) and many of our results can be viewed as extensions of the correspond results on the BCOP. Thus, let us now review some fundamental results on the BCOP, all of which are extendable to the BQCOP in later chapters.

1.4.1 The relation of the BCOP and the LCOP

It is well known that a BCOP can be formulated as a LCOP using exponentially increasing costs [39, 117, 141]. We give below a proof of this result, which is a modification of corresponding result for the TSP introduced by Kabadi and Punnen [144]. Let $\alpha_1 < \alpha_2 < \cdots < \alpha_t$ be an ascending arrangement of distinct costs c_e, when $c_e \leq u$ for some upper bounds u on the BCOP objective function value. Define $F_r = \{S \in \mathcal{F} : \max\{c_e : e \in S\} = \alpha_r\}$ for $r = 1, 2, \cdots, t$, $F_{t+1} = \{S \in \mathcal{F} : \max\{c_e : e \in S\} > \alpha_t\}$ and $U_r = \bigcup_{i=1}^{r} F_i = \{S \in \mathcal{F} : \max\{c_e : e \in S\} \leq \alpha_r\}$ for $r = 1, \cdots, t+1$. Let d_e be another real valued cost associated with $e \in E$.

Theorem 1. *If d satisfies $\min\left\{\sum_{e\in S} d_e : S \in F_r\right\} > \min\left\{\sum_{e\in S} d_e : S \in U_{r-1}\right\}$ for $2 \leq r \leq t+1$, then every optimal solution to the LCOP with costs d_e is also an optimal solution to the BCOP with costs c_e.*

Proof. If k is the smallest index such that F_k is non-empty, then it is obvious that any $S \in F_k$ is an optimal solution to the BCOP with the optimal objective function value α_k. Now assume that S' is an optimal solution to the LCOP with costs d_e and $S' \in F_q$ for some q. Then we have

$$\sum_{e\in S'} d_e = \min\left\{\sum_{e\in S} d_e : S \in F_q\right\} > \min\left\{\sum_{e\in S} d_e : S \in U_{q-1}\right\}.$$

Thus U_{q-1} must be empty and therefore S' is an optimal solution to the BCOP. □

For the bottleneck traveling salesman problem, Kabadi and Punnen [144] have shown several ways of constructing the costs d_e to satisfy the condition of the above theorem. It can easily be modified for the general BCOP. Nevertheless, d_e grow exponentially in the problem size $|E|$, which makes the transformation difficult to use in practice.

1.4.2 The threshold algorithm

By following the above notations we let $\alpha_1 < \alpha_2 < \cdots < \alpha_t$ be an ascending arrangement of distinct costs c_e. For any non-negative integer k, let U_k be the set containing all feasible solutions with objective function values no greater than α_k. Note that $U_1 \subseteq U_2 \subseteq \cdots \subseteq U_t$ where $U_t = \mathcal{F}$. Then we have the following theorem, the proof of which is straightforward and hence omitted.

Theorem 2. *If k is the smallest integer in $\{1, 2, \cdots, t\}$ such that $U_k \neq \emptyset$ then any $S \in U_k$ is an optimal solution to the BCOP.*

The BCOP feasibility problem (FBCOP) associated with a BCOP can be described as follows:"*Given a threshold μ, does there exist a feasible solution $S \in \mathcal{F}$ such that* $\max_{e \in S} c_e \leq \mu$*?*" Using Theorem 2, it is easy to see that the BCOP can be solved as a sequence of BCOP feasibility problems: starting from an integer k ($1 \leq k \leq t$) and consider the FBCOP with threshold α_k. Depending on whether the FBCOP has an answer "YES" or "NO", the k value can be decreased or increased so that an FBCOP with a smaller or larger threshold will be considered. This process is continued until the smallest k is found so that the FBCOP with α_k has an answer "YES", i.e. $U_k \neq \emptyset$. Binary search is a typical way for implementing this idea and we give a formal description of the algorithm, called *the binary search threshold algorithm*, in Algorithm 1.1.

Algorithm 1.1: Binary Threshold Algorithm

1: **Input:** E, $\mathcal{F}$, the costs c_e for $e \in E$ and an oracle $\gamma(.)$ which with input k verifies if $U_k = \emptyset$ and with an output 'yes' or 'no' answer along with an $S \in U_k$ whenever $U_k \neq \emptyset$.
2: **Output:** An optimal solution to the BCOP.
3: Construct an ascending arrangement $\alpha_1 < \alpha_2 < \cdots < \alpha_t$ of distinct costs c_e.
4: $l = 1$; $u = t$; $k = l$;
5: **while** $u - l > 0$ **do**
6: $\quad k = \lfloor \frac{(l+u)}{2} \rfloor$;
7: $\quad$ Call $\gamma(.)$ with k;
8: $\quad$ **if** $U_k = \emptyset$ **then** $l = k + 1$;
9: $\quad$ **else** $u = k$;
10: **end while**
11: Output any $S \in U_l$;

Theorem 3. *The BCOP can be solved in polynomial time if and only if the FBCOP is polynomially solvable. Further, the binary threshold algorithm computes an optimal solution to BCOP in $O(\phi(m)\log m)$ time, where $m = |E|$ and $O(\phi(m))$ is the complexity of FBCOP.*

Proof. If the FBCOP can be solved in polynomial time, then by Theorem 2, the binary search threshold algorithm correctly solves the BCOP. Since the algorithm calls $\gamma(.)$ for at most $O(\log(t))$ times and $t = O(m), O(\log(t)) = O(\log m)$, the complexity of the algorithm is $O(\phi(m)\log m)$. Thus if $\phi(m)$ is polynomial then the BCOP is solvable in polynomial time.

Conversely, if the QBCOP is solvable in polynomial time, let S^* be an optimal solution, choose q such that $\alpha_q = \max\{c_e : e \in S^*\}$, then $U_k = \emptyset$ if and only if $k < q$. Thus the FBCOP can be solved in polynomial time. □

1.4.3 The duality theorem

In the BCOP, a family of subsets of E is called *a clutter* if no member of the family is contained in another member of the family. Thus, without loss of generality, we assume that $\mathcal{F}$ is a clutter. For the clutter $\mathcal{F}$ on E, its blocking clutter (blocker) is the unique clutter $\mathcal{B} = \{T \subseteq E : |T \cap S| \geq 1 \ \forall S \in \mathcal{F}$ and T is minimal$\}$. Edmonds and Fulkerson [73] established a strong duality relationship between a clutter-blocker pair for the BCOP.

Theorem 4. *For any clutter $\mathcal{F}$ on a finite set E, there exists a unique clutter $\mathcal{B} = b(\mathcal{F})$ on E such that, for any function f from E to real numbers,*

$$\min_{S\in\mathcal{F}} \max_{e\in S} f(e) = \max_{T\in\mathcal{B}} \min_{e\in T} f(e).$$

Specifically, $\mathcal{B} = b(\mathcal{F})$ is the clutter consisting of the minimal subsets of E that have a non-empty intersection with every member of $\mathcal{B}$.

For the proof we refer to Edmonds [73].

1.4.4 Asymptotic results

Now, let us consider a theorem proved by Albrecher [4] where the asymptotic behavior of a certain class of combinatorial optimization problems with bottleneck objective function were discussed.

Let $n \in \mathcal{N}$ and for each n, P_n is a combinatorial optimization problem defined on finite set E_n. S_n is a family of feasible solutions of P_n. Let $f_n : E_n \to \mathbb{R}^+$ be a real valued weight function and $F_{n,min} = \min_{S \in S_n} \max_{e \in S} f_n(e)$.

Theorem 5. *Let $f_n(e)$ for all $e \in E_n$ and $n \in \mathcal{N}$ be identically distributed random variables in $[0, M]$ and assume that $f_n(e),\ e \in S$ are independent for every fixed feasible solution $S \in S_n,\ n \in \mathcal{N}$. Furthermore, let*

$$\log |S_n| + s_n \log \mathbb{P}\left(f_n(e) \leq \frac{M}{1 + g(n)} \right) \to -\infty \text{ as } n \to \infty$$

for some positive function $g(n)$, where $s_n = |S|$. Then

$$\mathbb{P}\left(\frac{F_{n,max} - F_{n,min}}{F_{n,min}} \leq g(n) \right) = 1 - o(1),$$

where $F_{n,max} = \max\limits_{S \in S_n} \max\limits_{e \in S} f_n(e)$ is the worst possible solution of P_n. Furthermore, if

$$\sum_{n=1}^{\infty} |S_n| \left(\mathbb{P}(f_n(e) \leq \frac{M}{1 + g(n)}) \right)^{s_n} < \infty,$$

then $\frac{F_{n,max} - F_{n,min}}{F_{n,min}} \leq g(n)$ holds almost surely, or equivalently

$$\mathbb{P}\left(\frac{F_{n,max} - F_{n,min}}{F_{n,min}} > g(n) \text{ infinitely often } \right) = 0.$$

The proof is omitted here.

1.5 Quadratic Bottleneck Combinatorial Optimization Problems (QBCOP)

Now let us introduce *the quadratic bottleneck combinatorial optimization problem (QBCOP)*. Given $E, \mathcal{F}$ as defined in the BCOP, if rather than having a cost c_e for every $e \in E$, we have costs for pairs of elements of E, i.e. for each $(e, f) \in E \times E$, a cost $q(e, f)$ is prescribed, then the QBCOP is to

$$\text{Minimize} \max_{\substack{e\in S\\ f\in S}} q(e,f)$$

$$\text{Subject to}$$

$$S \in \mathcal{F}.$$

Similar to our definition of the BCOP feasibility problem, *the QBCOP feasibility problem*, denoted FQBCOP, is to determine for a given threshold μ, if there exists a feasible solution $S \in \mathcal{F}$ such that $\max_{\substack{e\in S\\ f\in S}} q(e,f) \leq \mu$.

Lemma 3. *The FQBCOP is equivalent to the feasibility version of the LCOPC.*

Proof. Given a FBCOP with cost matrix Q and threshold μ, we define a conflict set $P = \{\{e,f\} : e \in E, f \in E, q(e,f) \geq \mu\}$. Then the FBCOP has a "YES" answer if and only if the LCOPC with P has a feasible solution.

Conversely, for an LCOPC with conflict set P, we let

$$q(e,f) = \begin{cases} 1 & \text{if } \{e,f\} \in P \\ 0 & \text{otherwise} \end{cases}$$

and then consider the FQBCOP with threshold value 0. The answer for the FQBCOP is "YES" if and only if the LCOPC is feasible. □

As we proved in Section 1.3, LCOPC can be reduced to a QCOP. Then from Lemma 3 the next result follows.

Lemma 4. *The FQBCOP reduces to the QCOP.*

QBCOP can be generally categorized into two groups: (a) QBCOP with specified $E, \mathcal{F}$, and (b) QBCOP with a special structured cost matrix Q. As formulated below, *the quadratic bottleneck assignment problem (QBAP)* (also referred to as *the bottleneck QAP* by many research papers) is the only studied problem falling into group (a).

$$\text{Minimize} \max_{i,j \in N} q_{i\pi(i)j\pi(j)}$$

$$\text{Subject to}$$

$$\pi \in \mathcal{P}_n$$

The QBAP was introduced by Steinberg as an application in backboard wiring, which is to minimize the maximum length of the involved wires [219]. Another well-known problem which can be modeled as a QBAP is *the matrix bandwidth minimization problem (MBMP)*[55, 177]. Given a matrix $A = \{a_{ij}\}$, its bandwidth is defined as the maximum absolute difference between i and j where $a_{ij} \neq 0$. The MBMP is to find a permutation of the rows and columns of A such that the nonzero elements are as close to the diagonal as possible, i.e. to minimize the bandwidth of A. It is easy to see that by letting

$$q_{i\pi(i)j\pi(j)} = \begin{cases} |\pi(i) - \pi(j)| & \text{if } a_{ij} \neq 0 \\ 0 & \text{otherwise,} \end{cases}$$

the MBMP can be modeled as a QBAP with cost matrix Q. In graph theory, a *graph bandwidth minimization* is to find a labeling of the vertices of a graph G such that the minimum absolute value of differences of labels of vertices which are connected by an edge is minimized. By taking A as the adjacency matrix of G, this problem is a MBMP. Thus, in the rest of this thesis, we simply call the MBMP *the bandwidth minimization problem (BMP)*.

The QBAP is an NP-hard problem, since it contains the BTSP as a special case. Burkard [34] derived several enumeration algorithms to solve the QBAP-KB (the bottleneck counterpart of the QAP-KB), and a Gilmore-Lawler-like bound was provided which solves a sequence of BAPs. For the general QBAP, he proposed a threshold procedure to reduce the coefficients. Some asymptotic results have also been presented by Burkard and Fincke [41].

For QBCOP special cases in group (b), we have the BCOP, in which $q(e, f) = 0$ for all $e, f \in E, e \neq f$; *the balanced optimization problem (BOP)* [45, 174], where $q(e, f) = |c_e - c_f|$; and *the 2-sum optimization problem (2-sum OP)* [118, 206] when $q(e, f) = c_e + c_f$. These three classes of problems are polynomially solvable whenever the corresponding LCOP can

be solved in polynomial time. We will discuss additional polynomially solvable cases later in this thesis.

1.6 Contributions

Despite the vast amount of work on QCOP, QBCOP has not received much attention in the literature. This is the primary motivation of this work. We investigate the general QBCOP, *the quadratic bottleneck spanning tree problem (QBST)*, *the quadratic bottleneck knapsack problem (QBKP)*, and the QBAP, together with some other related problems mentioned earlier such as *the minimum spanning tree with conflict constraints (MSTC)*, *the assignment problem with conflict constraints*, and *the maximum edge clique partitioning problem*. Summarized below are the main contributions of the thesis:

1. The minimum spanning tree problem with conflict constraints (MSTC) is considered in Chapter 2. The only known polynomially solvable case of the MSTC case is when the conflict graph is a collection of 2-paths. We have extended this result to the case when the conflict graph is a collection of disjoint cliques. Additional polynomially solvable cases of the MSTC are also explored. When the conflict graph is general but the original graph is sparse such as fan-stars, fans, wheels, ladders, and (k, n)-ladders, we have shown that the MSTC is NP-hard, whereas the adjacent-only MSTC and the AQMST on a ladder are polynomially solvable. We also obtained a recursive formula to count the number of spanning trees on a (k, n)-ladder unifying and generalizing similar formulas for fans and ladders [135, 214]. For MSTC on a cactus, we show that the feasibility problem is polynomially bounded whereas the optimization version is still NP-hard. Strong lower bounds of the MSTC are derived by exploiting polynomially solvable special cases and we also discuss various heuristic algorithms and feasibility tests. Results of preliminary experimental evaluation of heuristics and lower bounding schemes are given.

2. The maximum edge clique partitioning problem (Max-ECP) is investigated. This problem is NP-hard but efficient approximate solutions for it will help us get improved lower bounds for the MSTC, and this has motivated us to consider the Max-ECP. We show that if the maximum clique problem can be solved by a polynomial time ϵ-approximation algorithm, then the Max-ECP can be solved by a polynomial time

$\frac{2(p\epsilon-1)}{p-1}$-approximation algorithm for any fixed integer $p \geq 2$. As a consequence we have a polynomial time approximation algorithm for the Max-ECP with performance ratio bounded by $O\left(\frac{2}{p-1}(p\frac{n(\log\log n)^2}{(\log n)^3} - 1)\right)$ where $p \geq 2$ is a fixed integer and n is the number of nodes in the associated graph. This improves the best known bound on the performance ratio of an approximation algorithm for the problem. We also consider a related problem, where bi-cliques are extracted in the Max-ECP instead of cliques. We show that when considering only balanced bi-cliques, the greedy algorithm computes $4-\frac{4}{n}$ optimal solution and for the edge partitioning with general bi-cliques, the greedy algorithm can be arbitrarily bad.

3. We then consider the QBCOP, which has not been studied in the literature except a few isolated work. We show that the QBCOP can be formulated as a single quadratic min-sum problem with exponentially increasing cost coefficients. A weak duality theorem is proved and we give an example illustrating the existence of duality gap. A general purpose algorithm to solve the QBCOP is then developed, the complexity of which depends on a feasibility test which in turn depends on the complexity of feasibility version of the LCOP with conflict pairs. We show that the special case of the QBCOP, where feasible solutions are subsets of a finite set having the same cardinality, is NP-hard. When the cost function (or equivalently, matrix) is decomposable, multiplicative, or comparable, we show that the QBCOP is solvable in polynomial time whenever an associated linear bottleneck problem can be solved in polynomial time. A Gilmore-Lawler type lower bound is established for the QBCOP and based on it, we showed that the QBCOP with a row graded cost matrix and an additional property is also polynomially solvable, whenever the BCOP can be solved in polynomial time. As a consequence, the QBCOP with feasible solutions that form spanning trees, s-t paths, matchings, etc. of a graph are solvable in polynomial time with these types of cost functions (matrices). A similar result on the QBCOP with a row graded matrix can be extended to the quadratic matroid problem and it leads a new matroid characterization. The quadratic threshold algorithm can be implemented by solving a sequence of quadratic combinatorial optimization problems. An asymptotic result is presented at the end of this chapter. For the rest of the thesis, we consider special cases of the QBCOP and demonstrate how to apply the general purpose results by exploiting special problem structures.

4. As a special case of the QBCOP, the quadratic bottleneck spanning tree problem (QBST) is considered and the problem is shown to be NP-hard on a bipartite graph even if the cost function takes 0-1 values only. Solvable cases and the quadratic threshold algorithm discussed for the general QBCOP are then specified in the context of the QBST and we give computational complexities and the quadratic spanning tree threshold algorithm. Two lower bounds for the QBST are derived and compared. Efficient heuristic algorithms are presented for the QBST along with computational results.

5. The second special case of the QBCOP we look into is the quadratic bottleneck knapsack problem (QBKP), which has not appeared in literature yet. It is shown to be NP-hard and does not admit polynomial time ϵ-approximation algorithms. Polynomially solvable special cases, exact and heuristic algorithms are developed for the QBKP and we conducted experimental work on randomly generated test instances. We show that the heuristics can compute good quality solutions for large size problems in reasonable running time. Several variations of QBKP such as the quadratic bottleneck 0-1 programming problem, and the QBKP with "max-min" format of objective function and alternative capacity constraints are also discussed.

6. We then move on to the quadratic bottleneck assignment problem(QBAP). It is known to be NP-hard but to the best of our knowledge, there is no computational work derived to solve the QBAP in the literature. A quadratic assignment threshold algorithm is proposed and in each iteration, it needs to solve a assignment problem with conflict constraints (APC). Therefore we look into the APC, establish computational complexity results, identify nontrivial polynomially solvable cases and then develop algorithms such as local search, Tabu Search, and two variations of Tabu Search - the frequency guided tabu search (Tabu_FG) and the genetic mutation guided tabu search (Tabu_GM) to solve the problem, and experimental results are reported. Then by calling Tabu_GM in each iteration, we implement the algorithm for the QBAP and test it on the general QBAP and two special cases - the KB-QBAP and the bandwidth minimization problem.

The rest of the thesis is organized as follows: In Chapter 2, the minimum spanning tree with conflict constraints are studied. The maximum clique partitioning problem is discussed in Chapter 3. Chapter 4 deals with the general quadratic bottleneck combinatorial

optimization problem and after that, its special cases - the quadratic bottleneck spanning tree problem, the quadratic bottleneck knapsack problem and the quadratic bottleneck are sequentially considered in Chapter 5, 6, and 7. As a closely related problem, the assignment problem with conflict constraints are also included in Chapter 7. Concluding remarks are given in Chapter 8.

For convenience, the notations $V(G)$ and $E(G)$ are used sometimes to denote, respectively, the node set and edge set of a graph G. The quadratic cost $q(e_i, e_j)$ is simply written as $q(i, j)$.

Chapter 2

The Minimum Spanning Tree Problem with Conflict Constraints

2.1 Introduction

Let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. For each edge $e \in E$ a cost c_e is prescribed. A *conflict set* P is also given, which consists of some two-element subsets of E, and each $\{e, f\} \in P$ is called a *conflict pair.* A spanning tree T is called *conflict free* if T contains at most one edge from each conflict pair in P. Then, the problem MSTC is to find a least cost conflict free spanning tree of G.

The MSTC also includes as a special case the well-studied Hamiltonian path problem on a directed graph. To see this, consider an instance of a Hamiltonian path problem [120] from a specified node s to a specified node t in a directed graph $D = (V, A)$. Without loss of generality, let us assume that D does not contain any arc incident into s, any arc incident out of t and arc (s, t). Construct an undirected graph $G' = (V', E')$ as follows: $V' = \{s, t\} \cup \{u', u'' : u \in V \setminus \{s, t\}\}$; $E' = \{(u', u'') : u \in V\} \cup \{e'_i : e_i \in A\}$, where for each $e'_i \in E'$, if $e_i = (u, v)$ then $e'_i = (s, v')$ if $u = s$, $e'_i = (u'', t)$ if $v = t$ and $e'_i = (u'', v')$ otherwise. Assign to each edge $e'_i \in E'$ cost c_{e_i}; and to each of the edges $\{(u', u'') : u \in V\}$ assign a zero cost. For every pair e'_i, e'_j of distinct edges in G' such that e_i and e_j are both incident into or both incident out of a common node u, let $\{e'_i, e'_j\} \in P$. Then it is easy to verify that MSTC problem on (G, P) is equivalent to the instance of Hamiltonian path problem.

Let $\hat{G} = (\hat{V}, \hat{E})$ be the undirected graph with node set $\hat{V} = E$ and edge set $\hat{E}$ defined such that $(e, f) \in \hat{E}$ if and only if $\{e, f\} \in P$. Then $\hat{G}$ is called the *conflict graph*. It has been showed that MSTC is solvable in polynomial time if the associated conflict graph is a collection of disjoint edges, whereas the problem is NP-hard if the conflict graph is a collection of disjoint 2-edge paths with $0 - 1$ edge costs [62]. From this, it follows that computing a feasible solution or an ϵ-optimal solution to the MSTC is NP-hard for any $\epsilon > 0$.

Given a MSTC with conflict set P, if for all $\{e, f\} \in P$, e and f are adjacent, then we obtain a special case of the MSTC called *the minimum spanning tree with adjacent-only conflict pairs (MSTAC)*. The feasibility version of the MSTC, denoted by FSTC, is the decision problem: "Given a conflict set P, does there exists a spanning tree of G containing at most one of the edges in each conflict pair in P and if yes, how to find it?" and we denote the adjacent-only counterpart of the FSTC by FSTAC.

2.2 Complexity and polynomially solvable cases

The complexity result of Darmann et al [62] considers a general graph G with a very simple configuration for the conflict graph (disjoint 2-paths). This raises an interesting question: what is the complexity of the MSTC when G has a simple configuration whereas $\hat{G}$ is arbitrary? Let us consider some typical sparse graphs.

Given a path $P_n = v_1 - v_2 - \cdots - v_n$, *a fan ($F_n$)* is obtained by adding a node u and edges (u, v_i) for $i = 1, 2, \cdots, n$. If we add one more edge (v_1, v_n) to F_n, then the graph obtained is called *a wheel (W_n)*. If n is a multiple of 3, i.e. $n = 3h$ for some integer h, then we define a graph called *fan-star (FS_n)* by deleting the edges (v_{3i}, v_{3i+1}) $(i = 1, \cdots, h-1)$ from F_n. Examples of a fan, a wheel and a fan-star are shown in Figure 2.1.

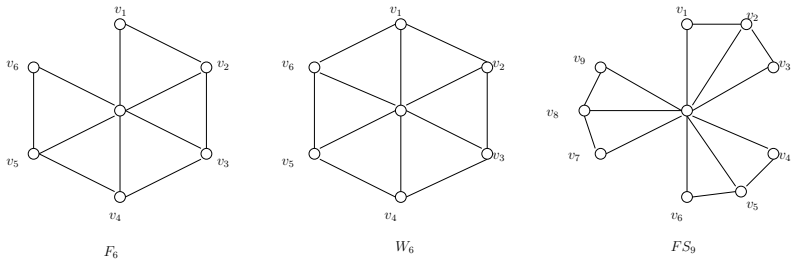

Figure 2.1: Examples: fan, wheel and fan-star

Given two node-disjoint paths $P_1 = v_1 - v_2 - \cdots - v_n$ and $P_2 = u_1 - u_2 - \cdots - u_n$, if we connect P_1 and P_2 by adding edges (v_i, u_i) for $i = 1, \cdots, n$, then the resulting graph is called a *ladder* (L_n), as shown in Figure 2.2.

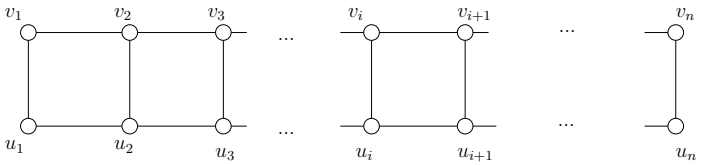

Figure 2.2: Example: ladder

Given a graph G, we use $\tau(G)$ to denote the number of spanning trees in G. Spanning tree counting formulas are known for F_n, W_n and L_n [135, 214]. As summarized in the following theorem, the number of spanning trees in these graphs can be exponentially large.

Theorem 6.

$$\tau(F_n) = \frac{1}{\sqrt{5}} \left\{ \left(\frac{3+\sqrt{5}}{2} \right)^n - \left(\frac{3-\sqrt{5}}{2} \right)^n \right\}.$$

$$\tau(W_n) = \left\{ \left(\frac{3+\sqrt{5}}{2} \right)^n - \left(\frac{3-\sqrt{5}}{2} \right)^n \right\} - 2.$$

$$\tau(L_n) = \frac{\sqrt{3}}{6} \left\{ \left(2+\sqrt{3} \right)^n - \left(2-\sqrt{3} \right)^n \right\}.$$

To generalize the fan and ladder structure, we introduce a new type of graph called *(k, n)-ladders*, where k is an integer greater than 2.

2.2.1 The (k, n)-ladder and a spanning tree counting formula

The process of constructing a (k, n)-ladder can be described in a recursive way as follows: given $k \geq 3$, let $L(k, 1)$ be a cycle of length k and $C_k^1 = L(k, 1)$. Then for any $n \geq 1$, $L(k, n+1)$ is obtained by adding a cycle of length k, denoted by C_k^{n+1}, to $L(k, n)$ such that C_k^{n+1} has $k-2$ new nodes (i.e. do not occur in $L(k, n)$) and exactly one common edge with C_k^n. Figure 2.3 shows examples of $L(5, 8)$ and $L(4, 9)$.

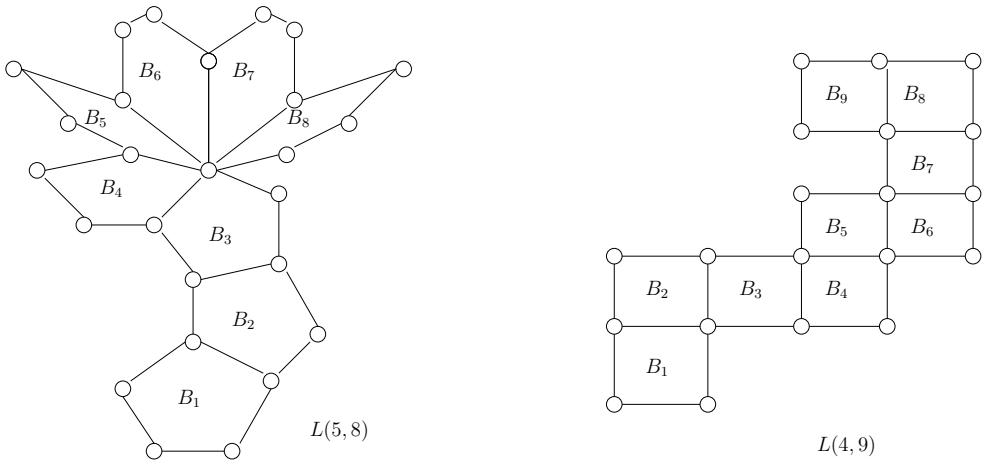

Figure 2.3: Examples: (k, n)-ladder

It can be seen that when $k = 3$ and all the C_3^t $(t = 1, \cdots, n)$ in the recurve definition have a common node, the (k, n)-ladder becomes a fan F_n. When $k = 4$ and $V(C_4^t \cap C_4^{t-2}) = \emptyset$ for $t = 3, \cdots, n$, the (k, n)-ladder reduces to a ladder L_n. Thus (k, n)-ladder is a proper generalization of the structures F_n and L_n.

As discussed in Theorem 6, the formulas on calculating the number of spanning trees in fans and ladders are already known. Now we generalize the results to (k, n)-ladders.

Lemma 5. *Let e be any edge of a graph G, then $\tau(G) = \tau(G - e) + \tau(G/e)$, where $G - e$ is the graph obtained by deleting e from G and G/e is obtained by contracting e.*

By contracting an edge $e = (u, v)$, we mean remove e and merge u and v into a new vertex w, where the edges originally incident to either u or v are now incident to w. The proof of Lemma 5 is straightforward since the total number of spanning trees of G is the number of spanning trees including e, plus the number of spanning trees excluding e [184].

Theorem 7. $\tau(L(k, n)) = k \cdot \tau(L(k, n-1)) - \tau(L(k, n-2))$.

Proof. Given a $(k, n-1)$-ladder $L(k, n-1)$, if we add a cycle C_r^n of length $r (2 \leq r \leq k)$ to it, where C_r^n contains $r-2$ new nodes and the two remaining nodes are adjacent in C_k^{n-1}, then the resulting graph is denoted by $\tilde{L}(r, n)$. Note that $\tilde{L}(k, n) = L(k, n)$. Take any $e \in C_r^n$ and from Lemma 5,

$$\tau(\tilde{L}(r, n)) = \tau(\tilde{L}(r, n) - e) + \tau(\tilde{L}(r, n)/e). \tag{2.1}$$

In (2.1), $\tilde{L}(r,n)-e$ consists of $\tilde{L}(k,n-1)$ and $r-2$ edges whose removal will make the graph unconnected, so $\tau(\tilde{L}(r,n)-e)=\tau(L(k,n-1))$. When $r\geq 3$, coalescing the endpoints of e will change C_r^n to a cycle of $(r-1)$ nodes, i.e. $\tau(\tilde{L}(r,n)/e)=\tau(\tilde{L}(r-1,n))$. , then from (2.1) we have

$$\tau(\tilde{L}(r,n))=\tau(L(k,n-1))+\tau(\tilde{L}(r-1,n). \tag{2.2}$$

Adding (2.7) for r from 3 to k yields

$$\tau(\tilde{L}(k,n))-\tau(\tilde{L}(2,n))=(k-2)\cdot\tau(L(k,n-1)).$$

i.e.

$$\tau(L(k,n))=\tau(\tilde{L}(2,n))+(k-2)\cdot\tau(L(k,n-1)). \tag{2.3}$$

Take any e from C_2^n of $\tilde{L}(2,n)$, $\tau(\tilde{L}(2,n)-e)=\tau(L(k,n-1))$, $\tau(\tilde{L}(2,n)/e)=\tau(\tilde{L}(k-1,n-1))$. So from Lemma 5,

$$\tau(\tilde{L}(2,n))=\tau(L(k,n-1))+\tau(\tilde{L}(k-1,n-1))=2\tau(L(k,n-1))-\tau(L(k,n-2)). \tag{2.4}$$

The result then holds by plugging (2.8) to (2.9). □

Theorem 7 gives a recursive formula for enumerating spanning trees on general ladders. When $k=3$, $\tau(L(k,n))=3\cdot\tau(L(k,n-1))-\tau(L(k,n-2))$, by solving it we obtain

$$\tau(L(3,n))=\frac{1}{\sqrt{5}}\left\{\left(\frac{3+\sqrt{5}}{2}\right)^n-\left(\frac{3-\sqrt{5}}{2}\right)^n\right\}. \tag{2.5}$$

When $k=4$, $\tau(L(k,n))=4\cdot\tau(L(k,n-1))-\tau(L(k,n-2))$, so

$$\tau(L(4,n))=\frac{\sqrt{3}}{6}\left\{\left(2+\sqrt{3}\right)^n-\left(2-\sqrt{3}\right)^n\right\}. \tag{2.6}$$

(2.12) and (2.6) are precisely the spanning tree enumeration formulas for fans and ladders given in Theorem 6. Thus Theorem 7 generalizes Theorem 6. Moreover, Theorem 7 always can be used to deduce explicit formulas for all the (k,n)-ladders.

2.2.2 The MSTC on fan-stars, fans, wheels, ladders and (k, n)-ladders

It is well known that many NP-hard problems become polynomially solvable when they are constricted to sparse graphs such as fans, ladders or more general, Halin graphs, series-parallel graphs etc. However, the case is not trivial for the MSTC.

Lemma 6. *The feasibility version of the minimum spanning tree problem with adjacent-only conflict constraints is NP-complete on fan-stars.*

Proof. We reduce the 3-SAT problem to a FSTAC on FS_n.

Given a 3-SAT problem in conjunctive normal form $s = (x_{11} \vee x_{12} \vee x_{13}) \wedge (x_{21} \vee x_{22} \vee x_{23}) \wedge \cdots \wedge (x_{n1} \vee x_{n2} \vee x_{n3})$, we construct a graph G with node set $\{u, v_{11}, v_{12}, v_{13}, \ldots, v_{n1}, v_{n2}, v_{n3}\}$ and edge set $E_1 \cup E_2$, where $E_1 = \{e_{ij} = (u, v_{ij}) : i = 1, \cdots, n, j = 1, 2, 3\}$, $E_2 = \{(v_{ij}, v_{ij+1}) : i = 1, \cdots n, j = 1, 2\}$. As shown in Figure 2.4, G is a fan-star.

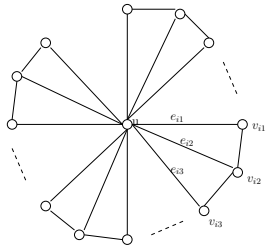

Figure 2.4: A fan-star constructed from a 3-SAT problem

The conflict set P is defined as $P = \{(e_{ij}, e_{kl}) :$ if x_{ij} and x_{kl} are negations of each other $\}$. Then the edges in each conflict pair are adjacent.

If T is a solution to the FSTAC on G, then let $X_T = \{x_{ij} : i = 1, \cdots, n, j = 1, 2, 3\}$ where x_{ij} is true if $e_{ij} \in T$, and false otherwise. For each $i = 1, \cdots, n$, there must be at least one of $x_{i1},\ x_{i2},\ x_{i3}$ with value true since T contains at least one of $e_{i1},\ e_{i2},\ e_{i3}$. Moreover, e_{ij} and e_{kl} can not both in T if $(e_{ij}, e_{kl}) \in P$, so in X_T, at most one of x_{ij}, x_{kl} will be true if they are negations of each other and hence X_T is a true assignment for the 3-SAT problem.

Conversely, suppose X is a true assignment for the 3-SAT problem, then $P = \{e_{ij} : x_{ij} =$ ture in $X\}$ is an acyclic subgraph of G with at most one edge from each conflict pairs. If P spans G, then it is a solution to the FSTAC. Otherwise we add necessary edges from E_2 to S to form a spanning tree T and T is then a solution to the FSTAC.

Therefore, we reduce the 3-SAT problem to a FSTAC on a fan-star and the NP-completeness of the latter is proved. □

Theorem 8. *The MSTAC is NP-complete on fan-stars.*

Proof. The result holds directly from Lemma 6. □

Then since a fan-star is a subgraph of a fan and a wheel, a MSTAC on a fan-star can be reduced to a MSTAC on a fan and a wheel by assigning large costs on additional edges. Thus we have the following corollaries.

Corollary 9. *The MSTAC is NP-complete on fans and wheels.*

Since a fan is a special case of a (k, n)-ladder, we have

Corollary 10. *The MSTAC is NP-complete on (k, n)-ladders.*

Corollary 11. *The MSTC is NP-complete on fans, wheels and (k, n)-ladders.*

Lemma 7. *The Feasibility version of the MSTC on ladders is NP-complete.*

Proof. Again we reduce the 3-SAT problem to a FSTC on a ladder.

Given a 3-SAT problem in conjunctive normal form $s = (x_{11} \vee x_{12} \vee x_{13}) \wedge (x_{21} \vee x_{22} \vee x_{23}) \wedge \cdots \wedge (x_{n1} \vee x_{n2} \vee x_{n3})$, we construct the ladder G shown in Figure 2.5. Let $P = \{(e_{ij}, e_{kl}) :$ if x_{ij} and x_{kl} are negations of each other $\}$, note that in P the conflict edges are not necessarily adjacent.

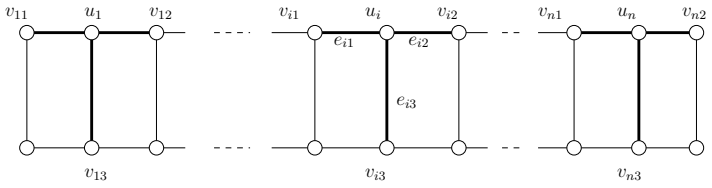

Figure 2.5: A ladder constructed from a 3-SAT problem

Then there exists a solution T to the FSTC on G, if and only if the 3-SAT problem has a true assignment. The detailed proof is similar to the one in Lemma 6 and hence omitted. □

Corollary 12. *MSTC on ladders is NP-complete.*

Despite the NP-hardness of the MSTC on a ladder, the MSTAC on a ladder can be solved in polynomially time. As a special case of the (k, n)-ladder, a ladder L_n can be obtained in a "dynamic" way, by keep adding a path of length 3 to L_{i-1}, where i grows

from 2 to n. If the conflict set P only includes adjacent edge pairs, then every time when a new path is added, the number of conflicts it would bring can be enumerated in polynomial time, so the MSTAC on L_i is polynomially solvable. This process can then be continued until the MSTAC on L_n is solved. In fact, this result holds for a more general problem - the QMST, i.e. the AQMST on ladders can be solved in polynomial time.

2.2.3 The MSTC on cactus

We now show that when G is a cactus (i.e. every edge in E lies on at most one cycle in G), the feasibility version of MSTC is solvable in polynomial time whereas the optimization version remains NP-hard. Suppose $G = (V, E)$ is a cactus , but the conflict set P is arbitrary. We assume, without loss of generality, that every edge in E lies on a cycle. This gives us a partition $(E_1, E_2, \ldots, E_k)$ of E where each $E_i = \{e_{1,i}, e_{2,i}, \ldots, e_{l_i,i}\}$ is the edge set of a cycle in G. Obviously, $l_i \geq 3\ \forall i$, and any $T \subseteq E$ is the edge set of a tree in G if and only if $|T \cap E_i| = l_i - 1\ \forall i$. Our problem thus reduces to choosing a set $T^* \subseteq E$ such that (i) $E - T^* = X^*$ contains precisely one edge from each E_i and X^* contains at least one element of each conflict pair in P, (feasibility), and (ii) $\sum_{e \in T^*} c_e$ is minimum, (or equivalently, $\sum_{e \in X^*} c_e$ is maximum), (optimality).

First of all, we shall show that we can assume, without loss of generality, that for each $i = 1, 2, \ldots, k$ the set P contains at most one 2-element set of the type $\{e_{p,i}, e_{q,i}\}$. For this, the following two operations will be useful.

Inclusion of an edge $e_{p,i}$ involves fixing $e_{p,i} \in X^*$ and deleting it from the set E_i, deleting from P all the sets of the type $\{e_{p,i}, e_{q,j}\}$ and performing the *exclusion* operation on all the edges in $E_i - \{e_{p,i}\}$, where the operation *exclusion* is defined as follows.

Exclusion of an edge $e_{p,i}$ involves deleting $e_{p,i}$ from the set E_i and performing the operation *inclusion* on all the edges $\{e_{q,j} : \{e_{p,i}, e_{q,j}\} \in P\}$.

Suppose for some $i \in \{1, 2, \ldots, k\}$, there exist more than one sets of the type $\{e_{p,i}, e_{q,i}\}$ in P. If the intersection of all such sets is empty the problem is obviously infeasible; else if $e_{p,i}$ lies in all such sets then inclusion of $e_{p,i}$ gives us a reduced, equivalent problem.

We shall now show that the MSTC problem on cactus is equivalent to the well-known weighted 2-SAT problem [76]. From the results on the 2-SAT problem it will then follow that the feasibility version of our problem can be solved in $O(\max\{|E|, |P|\})$ time [10], while the optimality version of the problem is NP-hard [76]. To show the equivalence, we shall

need the following problem.

Generalized 2-SAT problem (G2SAT): Here we are given a graph $\tilde{G} = (\tilde{N}, F)$ and a partition $(\tilde{N}_1, \tilde{N}_2, \ldots, \tilde{N}_k)$ of the set $\tilde{N}$, where $\tilde{N}_i = \{\tilde{n}_{1,i}, \tilde{n}_{2,i}, \ldots, \tilde{n}_{l_i,i}\}$ for some $l_i \geq 3$. For each node $p \in \tilde{N}$, a non-negative weight w_p is prescribed. The problem is to choose $\tilde{N}^* \subseteq \tilde{N}$ such that (i) $\tilde{N}^*$ contains exactly one node from each $\tilde{N}_i$ and each edge in F is incident to at least one node in $\tilde{N}^*$, and (ii) $\sum_{p \in \tilde{N}^*} w_p$ is maximum.

An instance of MSTC on a cactus can be formulated as G2SAT problem by defining $\tilde{n}_{i,j} = e_{i,j}$ and $w_{\tilde{n}_{i,j}} = c_{e_{i,j}}$, $\forall i, j$, and $\{\tilde{n}_{p,i}, \tilde{n}_{q,j}\} \in F$ if and only if $\{e_{p,i}, e_{q,j}\} \in P$.

Conversely, given an instance of G2SAT, define graph $G = (V, E)$ as follows: $V = \{0\} \cup \{i_j : i \in \{1, 2, \ldots, k\}, j \in \{i, 2, \ldots, l_i - 1\}\}$. Associate with each $\tilde{N}_i$ a cycle $(0, i_1, i_2, \ldots, i_{l_i-1}, 0)$, label the edges in this cycle as $E_i = \{e_{1,i}, e_{2,i}, \ldots, e_{l_i,i}\}$, and match each edge $e_{p,i}$ with element $\tilde{n}_{p,i}$ of $\tilde{N}_i$. Let $c_{e_{p,i}} = w_{\tilde{n}_{p,i}}$ $\forall p, i$ and let $\{e_{p,i}, e_{q,j}\} \in P$ if and only if $\{\tilde{n}_{p,i}, \tilde{n}_{q,j}\} \in F$. It is easy to see that $G = (V, E)$ is a cactus and this instance of MSTC is equivalent to G2SAT.

We now show that G2SAT is equivalent to the well-studied weighted 2-SAT problem [76], which can be stated as follows: We are given a collection $\{x_1, x_2, \ldots, x_k\}$ of k boolean variables each taking value either 1(= true) or 0(= false), a weight w_i for each x_i and a set $\{C_1, C_2, \ldots, C_m\}$ of m clauses, each of which is a 2-element subset of $\{v_1, v_2, \ldots, v_{2k}\} = \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_k, \bar{x}_k\}$, (here $\bar{x}_i = 1 - x_i$). The problem is to assign values to the variables such that (i) for each clause $C_i = \{v_p, v_q\}, v_p + v_q \geq 1$, and (ii) $\sum w_i x_i$ is maximum.

For a given instance of weighted 2-SAT problem, we construct an equivalent instance of G2SAT as follows: For each $i = 1, 2, \ldots, k$, define $\tilde{N}_i = \{\tilde{n}_{1,i}, \tilde{n}_{2,i}, \tilde{n}_{3,i}\}$, where $\tilde{n}_{1,i}$ corresponds to x_i, $\tilde{n}_{2,i}$ corresponds to $\bar{x}_i$ and $\tilde{n}_{3,i}$ is a dummy node. Let $w_{\tilde{n}_{1,i}} = w_i$, $w_{\tilde{n}_{2,i}} = w_{\tilde{n}_{3,i}} = 0$.

For any clause $C_i = \{v_p, v_q\}$, let $v_p \in \{x_i, \bar{x}_i\}$ and $v_q \in \{x_j, \bar{x}_j\}$. Add edge $(\tilde{n}_{a,i}, \tilde{n}_{b,j})$ to F, where $a = 1$ if $v_p = x_i$, $a = 2$ if $v_p = \bar{x}_i$, $b = 1$ if $v_q = x_j$ and $b = 2$ if $v_q = \bar{x}_j$. In addition, for each $i = 1, 2, \ldots, k$, add edge $(\tilde{n}_{1,i}, \tilde{n}_{2,i})$ to F.

It is easy to see that no solution $\tilde{N}^*$ to G2SAT contains a node $\tilde{n}_{3,i}$ for any i. From any solution to the weighted 2-SAT problem, we can construct a solution $\tilde{N}^*$ to G2SAT that has the same objective function value, and vice versa, as follows: For each $i = 1, \ldots, k$, $x_i = 1$ if and only if $\tilde{n}_{1,i} \in \tilde{N}^*$; $x_i = 0$ if and only if $\tilde{n}_{2,i} \in \tilde{N}^*$.

Now, suppose we are given an instance of G2SAT problem. We construct a corresponding instance of weighted 2-SAT problem as follows: Our variables are $\{x_{ij} : j = 1, 2, \ldots, k,\ i = 1, 2, \ldots, l_j\}$. With each edge $(\tilde{n}_{p,i}, \tilde{n}_{q,j}) \in F$, we associate a clause $C_{piqj} = \{x_{pi}, x_{qj}\}$. In addition, we create clauses $\{\{\bar{x}_{pi}, \bar{x}_{qi}\} : i = 1, 2, \ldots, k; \{p, q\} \subseteq \{1, 2, \ldots, l_i\}, p \neq q\}$. Let $w_{ij} = w_{\tilde{n}_{j,i}} + M$, where M is a sufficiently large integer.

It is easy to see that for any feasible solution to the instance of weighted 2-SAT problem, if for some $j \in \{1, 2, \ldots, k\}$, $x_{ij} = 0\ \forall i$, then by arbitrarily choosing an $i \in \{1, 2, \ldots, l_j\}$ and making $x_{ij} = 1, \bar{x}_{ij} = 0$ gives us an alternate feasible solution. Using this, equivalence of sets of optimal solutions of the two problems as well as equivalence of the corresponding feasibility problems are easy to verify and the proof is omitted.

Since the weighted 2-SAT problem is NP-hard [76] but its feasibility version can be solved in $O(m)$ [10], we get the following theorem[1]:

Theorem 13. *The MSTC on a cactus is NP-hard. However, its feasibility version can be solved in* $O(\max\{|E|, |P|\})$ *time.*

2.2.4 The MSTC with special conflict graphs

Now let us consider the polynomially solvable MSTC cases when G is arbitrary and $\hat{G}$ is special structured.

The conflict set (conflict relation) P is said to be *transitive* if and only if for distinct e, f, g in E, $\{e, f\} \in P$, and $\{f, g\} \in P$, imply $\{e, g\} \in P$.

Lemma 8. *The conflict graph* $\hat{G}$ *is a collection of disjoint cliques if and only if the conflict set* P *is transitive.*

The straightforward proof of Lemma 8 is omitted here. We now show that the MSTC is polynomially solvable whenever P is transitive. Let $\hat{G}_1, \hat{G}_2, \ldots, \hat{G}_p$ be the connected components of the conflict graph $\hat{G} = (\hat{V}, \hat{E})$. Then by Lemma 8, each $\hat{G}_i$ is a clique. Let $V(\hat{G}_i) = E_i$ for $i = 1, 2, \ldots, p$. Then any conflict free tree T of G contains at most one edge from each E_i. Let F_1 be the family of all subsets of E satisfying the property that $Q \in F_1 \Leftrightarrow |Q \cap E_i| \leq 1$ for all i. Then (E, F_1) is a partition matroid [231]. Let F_2 be the collection of edge sets of all spanning trees of G. Then (E, F_2) is the base system of a graphic matroid [231]. Thus, in this case, MSTC reduces to the problem of finding

[1]The proof given here was suggested by my collaborator Dr. Santosh Kabadi.

a minimum cost set T in $F_1 \cap F_2$, i.e. MSTC is a special case of the well-solved weighted matroid intersection problem [72]. The preceding discussion is summarized in the theorem below.

Theorem 14. *When the conflict graph is a collection of disjoint cliques (equivalently when P is transitive), MSTC can be solved in polynomial time.*

Although the general weighted matroid intersection problem is polynomially solvable, it may be noted that special algorithms are available when the underlying matroids are of graphic and partition types with better worst case complexity [32]. Since our matroid intersection problem is precisely of this type, we can solve the problem in $O(nK^2+nm+Kn^2)$ time, where K is the number of cliques. Our next two theorems indicate that when the conflict graph is slightly deviated from the structure of a collection of disjoint cliques, MSTC can still be solved in polynomial time.

Theorem 15. *Suppose the conflict graph $\hat{G}$ is such that deletion of a fixed number, k, of nodes, that can be identified in polynomial time, reduces $\hat{G}$ to a collection of node-disjoint cliques. Then the corresponding instance of MSTC can be solved in polynomial time.*

Proof. Consider each of the k nodes in $\hat{G}$, deletion of which reduces $\hat{G}$ to a collection of node-disjoint cliques. If the edge in G corresponding to such a node, x, is to be excluded from a tree, then we delete the edge from G and the node x from $\hat{G}$. If this edge is to be included in a tree, then we change its cost in G to $-M$, where M is a suitably large number and in $\hat{G}$, we delete all the neighbors of x from $\hat{G}$. There are 2^k ways one can choose the k nodes to be included or excluded. It can be verified that in any given such selection, after performing the above mentioned operations, the resulting conflict graph will be a collection of disjoint cliques. By Theorem 14, this problem can be solved in polynomial time. If the optimal solution does not include all the edges of cost $-M$, the selection is not feasible and we discard it. Otherwise, we compute the objective function value of the resulting tree with respect to the original costs. Repeating this for all possible 2^k selections and choosing the overall best tree gives us an optimal solution to the original MSTC. Since k is fixed, the result follows from Theorem 14. □

Note that in Theorem 15, although we fix the number of nodes to be deleted, the number of conflict pairs associated with these nodes need not be fixed since each such node can have

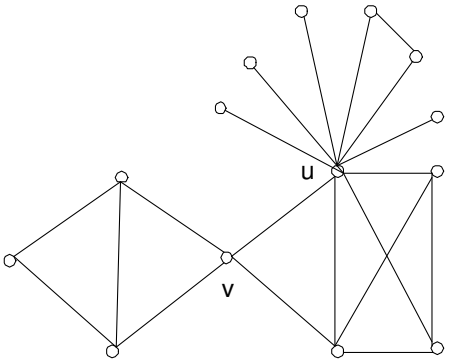

Figure 2.6: Example of a graph satisfying conditions of Theorem 15. Here, $k = 2$ and nodes u and v are the nodes to be deleted.

a degree as large as $(n-1)$. Figure 2.6 gives an example of a conflict graph satisfying conditions of Theorem 15.

As a corollary of Theorem 15, we have the following.

Corollary 16. *If the conflict graph has a fixed number k of edges, which can be identified in polynomial time, so that deletion of these edges makes the conflict graph a collection of disjoint cliques, then the MSTC can be solved in polynomial time.*

When the conflict graph is not a collection of disjoint cliques (and also does not satisfy the condition of Theorem 15 for small value of k), one way to obtain a lower bound for MSTC is to relax some conflict relations (equivalently, delete some edges from $\hat{G}$) so that the resulting graph is a collection of disjoint cliques with maximum number of edges. We will discuss this type of lower bounds in the coming section.

2.3 Integer programming formulations and lower bounds

Let $E = \{1, 2, \ldots, m\}$ and T be the edge set of a spanning tree of G. The incidence vector $x = (x_1, x_2, \ldots, x_m)$ of T is defined as

$$x_i = \begin{cases} 1 & \text{if } i \in T \\ 0 & \text{Otherwise.} \end{cases}$$

Let $\mathcal{F}$ be the spanning tree polytope of G, i.e. $\mathcal{F}$ is the convex hull of incidence vectors of spanning trees of G. Then MSTC can be formulated as a 0-1 integer linear program

$$\begin{aligned} \text{ILP: Minimize } & \sum_{e \in E} c_e x_e \\ \text{Subject to} & \\ & x \in \mathcal{F} \\ & x_e + x_f \leq 1 \ \forall \ \{e, f\} \in P \\ & x_e = 0 \text{ or } 1 \ \forall e \in E \end{aligned} \tag{2.7}$$

Another integer programming formulation of MSTC can be described as follows:

$$\begin{aligned} \text{ILP-Star: Minimize } & \sum_{e \in E} c_e x_e \\ \text{Subject to} & \\ & x \in \mathcal{F} \\ & d_e x_e + \sum_{j \in \hat{V}(e)} x_j \leq d_e \ \forall \ e \in E \\ & x_e = 0 \text{ or } 1 \ \forall e \in E \end{aligned} \tag{2.8}$$

where $\hat{V}(e)$ is the set of nodes in $\hat{G}$ that are adjacent to e and $d_e = |\hat{V}(e)|$. It can be verified that the set of binary solutions of (2.7) and (2.8) are equivalent and hence ILP and ILP-star are equivalent. We call constraints (2.8) the *star inequalities*. ILP-Star gives a more compact representation of MSTC than ILP.

MSTC can also be formulated as a *quadratic minimum spanning tree problem* (QMST) as follows:

$$\begin{aligned} \text{Minimize } & \sum_{e \in E} \sum_{f \in E} d_{ef} x_e x_f \\ \text{Subject to} & \\ & x \in \mathcal{F} \end{aligned}$$

$$x_e = 0 \text{ or } 1 \ \forall e \in E$$

where $D = (d_{ef})$ is the $m \times m$ matrix defined as

$$d_{ef} = \begin{cases} c_e & \text{if } e = f \\ M & \text{if } \{e, f\} \in P, e \neq f \\ 0 & \text{if } \{e, f\} \notin P, e \neq f \end{cases}$$

and M is a large number. So that if the QMST has optimal objective function value less than M, then the optimal solution to the QMST is also optimal to the MSTC. Otherwise the MSTC is infeasible.

The algorithms for the QMST can be used to solve MSTC. However, exploiting the special structure of MSTC, we develop additional heuristic algorithms.

Let us now focus our attention to computing good quality lower bounds for the problem MSTC. We first consider a somewhat straightforward lower bound using Lagrangian relaxation of constraints (2.7) in ILP. Let λ_{ef} be the Lagrange multiplier associated with conflict constraint $x_e + x_f \leq 1$ in ILP for all $\{e, f\} \in P$. For any node e in the conflict graph $\hat{G}$, let $\hat{V}(e)$ be the set of its adjacent nodes. Consider the Lagrangian function

$$\begin{aligned} L(\lambda) &= \min \left\{ \sum_{e \in E} c_e x_e + \sum_{\{e,f\} \in P} \lambda_{ef}(x_e + x_f - 1) : x \in \mathcal{F} \right\} \\ &= - \sum_{\{e,f\} \in P} \lambda_{ef} + \min \left\{ \sum_{e \in E} \left(c_e + \sum_{f \in \hat{V}(e)} \lambda_{ef} \right) x_e : x \in \mathcal{F} \right\} \end{aligned}$$

From Lagrangian duality, $L(\lambda)$ is a lower bound for the optimal objective function value of MSTC for each $\lambda \geq 0$. Thus

$$L^* = \max_{\lambda \geq 0} L(\lambda)$$

is a lower bound on the optimal objective function value of MSTC. Note that L^* can be obtained using any algorithm for a non-differentiable convex optimization problem; in particular the subgradient algorithm [215] or the volume algorithm [16]. In each subgradient iteration, we need to evaluate $L(\lambda)$ for a given vector $\lambda = (\lambda_{ef} : \{e, f\} \in P)$. This can be

accomplished by solving the minimum spanning tree problem:

$$\text{Minimize } \left\{ \sum_{e \in E} \left(c_e + \sum_{f \in \hat{V}(e)} \lambda_{ef} \right) x_e \right\}$$

$$\text{Subject to}$$

$$x \in \mathcal{F}.$$

It is possible to obtain a lower bound, say L^*_{Star} similar to the one discussed above, by considering the Lagrangian relaxation of the star inequalities of ILP-Star. However, it is observed that the resulting bound is inferior to the bound L^* and the computational advantage in this case is not significant. Thus we discarded L^*_{Star} from further investigation.

2.3.1 Improving the lower bound

The lower bound L^* obtained above can be improved by investing additional computational effort. Let Δ be a subset of the edge set of conflict graph $\hat{G}$ such that the graph $\tilde{G} = \hat{G} - \Delta$ is a collection of disjoint cliques. From Theorem 14, MSTC on G with $\tilde{G}$ as conflict graph can be solved in polynomial time. To exploit this information in computing an improved lower bound, we rewrite ILP as

$$\text{ILP-MI: Minimize} \sum_{e \in E} c_e x_e$$

$$\text{Subject to}$$

$$x \in \mathcal{F}$$

$$x_e + x_f \leq 1 \ \forall \ (e, f) \in E(\tilde{G}) \tag{2.9}$$

$$x_e + x_f \leq 1 \ \forall \ (e, f) \in \Delta \tag{2.10}$$

$$x_e = 0 \text{ or } 1 \text{ for all } e \tag{2.11}$$

Let $\bar{G}$ be the subgraph of $\hat{G}$ with edge set Δ. For any node e in the graph $\bar{G}$, let $\bar{V}(e)$ be the set of its adjacent nodes. Consider the Lagrangian function $\ell(\lambda)$ obtained by relaxing (2.10) in ILP-MI. We have,

$$\ell(\lambda) = \min \left\{ \sum_{e \in E} c_e x_e + \sum_{(e,f) \in \Delta} \lambda_{ef}(x_e + x_f - 1) : x \in \mathcal{F}, x_e + x_f \leq 1 \ \forall \ (e,f) \in E(\tilde{G}) \right\}$$

$$= - \sum_{(e,f) \in \Delta} \lambda_{ef} + \min \left\{ \sum_{e \in E} \left(c_e + \sum_{f \in \bar{V}(e)} \lambda_{ef} \right) x_e : x \in \mathcal{F}, x_e + x_f \leq 1 \ \forall \ (e,f) \in E(\tilde{G}) \right\}$$

From Lagrangian duality,

$$\ell^* = \max_{\lambda \geq 0} \ell(\lambda)$$

is a lower bound on the optimal objective function value of MSTC. As discussed in the case of L^*, the lower bound ℓ^* can be obtained using the subgradient algorithm [215] or using the volume algorithm [16]. In each subgradient iteration, we need to evaluate $\ell(\lambda)$ for a given vector $\lambda = (\lambda_{ef} : (e,f) \in \Delta)$. This can be accomplished by solving the weighted matroid intersection problem:

$$\text{MI}(\lambda) : \min \sum_{e \in E} \left(c_e + \sum_{f \in \bar{V}(e)} \lambda_{ef} \right) x_e$$

Subject to

$$x \in \mathcal{F}$$

$$x_e + x_f \leq 1 \ \forall \ (e,f) \in E(\tilde{G}) \tag{2.12}$$

It is easy to see that $\ell^* \geq L^*$. The quality of the lower bound ℓ^* depends on clever choices of Δ. Ideally we want to choose Δ such that $|\Delta|$ is as small as possible so that the resulting graph $\tilde{G}$ is a collection of disjoint cliques with maximum number of edges. This is precisely the maximum edge clique partitioning problem (Max-ECP) [68, 81, 158], which is known to be NP-hard. Here we consider an approximate algorithm to solve the Max-ECP, called *the approximate greedy algorithm*, which iteratively extracts a clique of reasonably large size from the conflict graph (although we are going to solve the Max-ECP on the conflict graph, to keep the generality of our results, we assume that the Max-ECP is defined on a general graph G). So in each iteration of the approximate algorithm, a procedure Approx-Clique(G) is available which with input a graph G outputs an approximate solution (clique) for the

maximum clique problem on G. A formal description of the algorithm is given below.

Algorithm 2.1: The Approximate Greedy Algorithm

Input: The graph G;
$H = \emptyset$, G^1=G, k=1;
while $E(G^k) \neq \emptyset$ **do**
  D^k = Approx-Clique(G^k);
  $H = H \cup \{D^k\}$;
  $G^{k+1} = G^k - V(D^k)$;
  $k = k + 1$;
end while;
Output: $H \cup \{G^k\}$.

In Chapter 3, we will discuss the Max-ECP approximate algorithms in more details. A 2-phase approximate greedy algorithm will be derived, which improves the existing approximation bound for the Max-ECP, and Algorithm 2.1 is in fact a special case of the 2-phase approximate greedy algorithm.

Another heuristic algorithm to solve Max-ECP on $\hat{G}$ (i.e. to generate $\tilde{G}$ from $\hat{G}$) is obtained by designating $\tilde{G}$ as a maximum cardinality matching in $\hat{G}$. We call this the *matching heuristic*. Again, the performance ratio for matching heuristic was discussed in Chapter 3. Although the approximate greedy heuristic may appear stronger than the matching heuristic in general, it may be noted that the matching heuristic could perform significantly better in certain classes of graphs. For example consider the graph in Figure 2 which consists of a collection of r disjoint 3-edge paths.

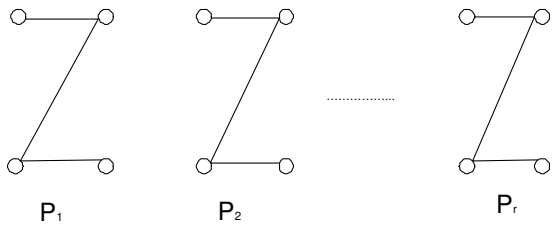

Figure 2.7: Example: Matching heuristic vs Approximate greedy

Potentially, the approximate greedy heuristic could output the central edge from each of the 3-edge paths resulting in a solution of cardinality r where as the matching heuristic

will produce the optimal solution to Max-ECP with cardinality $2r$.

2.4 Upper bounds and heuristics

We now discuss several heuristics for the MSTC. These algorithms include construction heuristics, local search, tabu search, tabu thresholding, and Lagrangian based heuristics. As noted earlier, computing a feasible solution to MSTC itself is NP-hard. Thus, our heuristic algorithms may not always compute a feasible solution. When a feasible solution is not obtained, we try to minimize the number of violated conflict constraints.

2.4.1 A simple construction heuristic

A very fast construction heuristic for MSTC can be obtained as follows. In the graph G we choose an $e \in E$ which appears in the maximum number of conflict pairs, delete e from G and delete from P all the conflict pairs containing e. We repeat this process with the updated G and P, until $P = \emptyset$. If G is still connected, then the cost of its minimum spanning tree is an upper bound to the MSTC, otherwise the algorithm returns no feasible solution.

2.4.2 Local Search

Let T be a spanning tree and $e \in E - T$, i.e. e is a non-tree edge. The graph $T + e$ contains a unique cycle with edges $e_1, e_2, \ldots, e_{t_e}$. Then $T_i = T + e - e_i$ is a spanning tree of G, for $1 \leq i \leq t_e$. Let $R(e) = \{T + e - e_i : i = 1, 2, \ldots, t_e\}$ and $N(T) = \cup_{e \in E-T} R(e)$. The set $N(T)$ is called a *2-exchange neighborhood*. The neighborhood $N(T)$ is well known and is used by many researchers for developing local search algorithms for spanning tree type problems. We use this neighborhood to develop a local search heuristic for MSTC.

Since finding a feasible solution to MSTC is NP-hard, to have a meaningful local search algorithm we consider a modified objective function $f(T) = c(T) + \alpha g(T)$ where $c(T) = \sum_{e \in T} c_e$, α is a large number and $g(T) = |\{\{e, f\} : \{e, f\} \in P \text{ and } e, f \in E(T)\}|$. Thus $g(T)$ measures the number of violated conflict constraints. The spanning tree T is feasible if and only if $g(T) = 0$. For any edge e of G let $Z(e, T) = \{h : \{e, h\} \in P, h \in T, e \neq h\}$. Given the value of $f(T)$ the objective function value of the tree $T_i = T + e - e_i$ can be computed

as $f(T_i) = C(T_i) + \alpha g(T_i)$ where $C(T_i) = C(T) + c_e - c_{e_i}$ and

$$g(T_i) = \begin{cases} g(T) + |Z(e,T)| - |Z(e_i,T)| - 1 & \text{if } \{e, e_i\} \in P \\ g(T) + |Z(e,T)| - |Z(e_i,T)| & \text{otherwise} \end{cases}$$

Note that the size of the neighborhood $N(T)$ is $O(mn)$ and the neighborhood can be searched for an improving solution in $O(mn)$ time by maintaining appropriate data structure.

The local search algorithm for MSTC is now straightforward. Starting with a spanning tree T, the algorithm examines the neighborhood $N(T)$ and move to the first improving solution, and then we check the neighborhood of the new solution for the first improving solution again. This recursion will stop when a local optimum T^* is reached. If $g(T^*) = 0$, the local optimum solution is feasible. Otherwise, we report the objective function values in terms of $C(T)$ and $g(T)$ where $g(T)$ gives the number violated conflict constraints.

2.4.3 Tabu Thresholding Heuristic

Whether the local search terminates with a feasible solution or not, further improvements in terms of reducing the number of conflict pair violations or improving the objective function value of a feasible solution can be explored by careful and systematic search procedures. Tabu thresholding [106] is one such strategy that can be used to achieve this goal. This approach was useful in finding good quality solutions for various combinatorial optimization problems [23, 47]. Tabu thresholding can be viewed as a special randomized local search algorithm [155]. The algorithm uses two parameters δ and ω to control solution quality and search diversification.

Suppose T is a local solution as defined in the local search algorithm. Let $T_1, T_2, \ldots, T_\sigma$ be the spanning trees of G in $N(T)$ where $f(T_1) \leq f(T_2) \leq \cdots \leq f(T_\sigma)$. Choose a spanning tree T^0 randomly from $\{T_1, T_2, \ldots, T_\omega\}$ where ω is a parameter. Move to the solution T^0 and we call such a move *reasonably random move*. Continue the reasonably random moves for δ iterations, where δ is another parameter that controls the algorithm. After completing the reasonably random move phase, we switch back to local search phase until a local minimum is reached and then reasonably random moves are invoked. We continue the process until a prescribed number, say *max-iter*, of iterations (number of times local search is invoked) are completed and we output the best solution obtained.

2.4.4 Tabu Search Heuristic

Another way to explore the neighborhood $N(T)$ beyond a local optimum is to employ the tabu search method [104, 105, 60]. For the current solution T, if for some $e_i \in E \setminus T$, $e_j \in T$, $T + e_i - e_j$ is a spanning tree of G, we call (e_i, e_j) a *2-exchange pair*. The tabu list we construct in this algorithm consists of some 2-exchange pairs. We define $N_A(T) = \cup_{e_i \in E \setminus T}\{T + e_i - e_j : (e_i, e_j)\ is\ 2-exchange\ pair,\ (e_i, e_j) \notin Tabu\ List\}$, so $N_A(T) \subseteq N(T)$ and for any moves in $N_A(T)$, the corresponding 2-exchange pairs are not on the tabu list.

In each iteration, we first find the best solution T^0 in $N(T)$. Suppose $T^0 = T + e_i^0 - e_j^0$, then if (e_i^0, e_j^0) is not on the tabu list, we will move from T to T^0 and add (e_i^0, e_j^0) to the Tabu-List. To keep the length of the Tabu-List within a fixed length L, we add the new pair to the end of the list and once the length of the list is L, the pair on the top will be removed. If (e_i^0, e_j^0) is on the tabu list, we will check the aspiration criteria, which is set as $f(T^0)$ is less than the best objective function value so far. If the aspiration criteria is satisfied, we will make the move without updating the tabu list, otherwise we will find the best solution $\tilde{T}^0$ in $N_A(T)$ and move to $\tilde{T}^0$. The process is continued until a prescribed number of iterations (*max-iter*) is completed and output the best solution obtained.

2.4.5 Lagrangian Heuristic

When solving the Lagrangian problem of ILP or ILP-MI, using subgradient optimization, we generate a sequence of spanning trees. It is useful to keep track of the objective function value $f(T)$ of these solutions and the best solution so obtained is output as a heuristic solution.

2.5 Infeasibility tests

Note that testing if MSTC has a feasible solution or not is an NP-hard problem. We now develop some sufficiency conditions that can be used to test infeasibility of the problem.

Lemma 9. *Let α be an upper bound on the independence number of the conflict graph $\hat{G}$. If $\alpha < |V(G)| - 1$ then MSTC is infeasible.*

Proof. Note that nodes of $\hat{G}$ are precisely edges of G. Let κ be the independence number of $\hat{G}$. For any feasible tree T of G, $E(T)$ corresponds to an independent set of $\hat{G}$. Thus

$|V(G)|-1 = |E(T)| \leq \kappa$. Thus, if $|V(G)|-1 > \kappa$, MSTC must be infeasible. If $|V(G)|-1 > \alpha$ holds, then $|V(G)| - 1 > \kappa$ holds and hence MSTC is infeasible. □

Although computing κ in Lemma 9 is NP-hard, reasonable values of α can be obtained by solving various relaxations of the maximum independent set problem on $\hat{G}$. We used the following scheme to get an estimate of α. Consider the integer programming formulation of the maximum independent set problem and invoke an integer programming solver (we used CPLEX) using branch and bound/ branch and cut algorithm. At any stage of the algorithm, if the smallest upper bound among the upper bounds at active nodes is less than $n-1$, by Lemma 9 we can conclude that the MSTC is infeasible. Also, at any stage of the algorithm, if an independent set of size $n-1$ or more is found, we can conclude that the test fails and the algorithm terminates. The algorithm can also be terminated based on computational time constraints. We call this infeasibility test, the *maximum independent set test* (MIS test).

Lemma 10. *If β is a lower bound on the size of the minimum vertex cover of $\hat{G}$ and $|E(G)| - \beta < |V(G)| - 1$ then MSTC is infeasible. Further, if $D_1, D_2, \ldots, D_r$ is a clique partitioning of $\hat{G}$ and $|V(G)| - 1 > |E(G)| + r - \sum_{i=1}^{r} |D_i|$ then MSTC is infeasible.*

Proof. Let $m = |E(G)| = |V(\hat{G})|$ and $n = |V(G)|$. If β is a lower bound on minimum vertex cover of $\hat{G}$, them $m - \beta$ is an upper bound on the independence number of $\hat{G}$. The first part of the result follows from Lemma 9. If $D_1, D_2, \ldots, D_r$ is a clique partitioning of $\hat{G}$, then any vertex cover must select at least $|D_i| - 1$ vertices from D_i for each i. Thus $\sum_{i=1}^{r} |D_i| - r$ is a lower bound for the size of minimum vertex cover of $\hat{G}$ and the result follows. □

Corollary 17. *Let M be a maximum cardinality matching in $\hat{G}$. If $|V(G)|-1 > |E(G)|-|M|$ then MSTC is infeasible.*

Note that a clique partition of $\hat{G}$ is constructed for computing the lower bound SUB+MI later. This can be exploited to apply an infeasibility test based on the second part of Lemma 10 and we call it the *clique partition test* (CP test). Also, similar to the MIS test, Lemma 10 can be used to develop another test based on the minimum vertex cover problem. Consider the integer programming formulation of the minimum vertex cover problem and apply an integer programming solver (we used CPLEX) using branch and bound/ branch and cut algorithm. At any stage of the algorithm, if the largest lower bound among the lower bounds at the active nodes is less than $n-1$, by Lemma 10 we can conclude that the

MSTC is infeasible. Also, at any stage of the algorithm, if a vertex cover of size $m - n + 1$ or more is found, we can conclude that the test fails and the algorithm terminates. The algorithm can also be terminated based on computational time constraints. We call this feasibility test, the *minimum vertex cover test* (MVC test).

ILP can be reformulated with a compact representation using the single commodity formulation of the minimum spanning tree problem [172] along with the conflict constraints. Let $\vec{G}$ be the directed version of G obtained by replacing each undirected edge $\{i, j\}$ of G by two directed edges (i, j) and (j, i). Let $\delta^+(i) = \{j : (i, j) \in V(\vec{G})\}$, $\delta^-(i) = \{j : (j, i) \in V(\vec{G})\}$ and $V(G) = \{1, 2, \ldots, n\} = V(\vec{G}\}$. The following formulation gives a compact integer programming representation of MSTC.

$$\text{CILP: Minimize } \sum_{e \in E(G)} c_e x_e$$

Subject to

$$\sum_{j \in \delta^+(1)} g_{1j} - \sum_{j \in \delta^-(1)} g_{j1} = n - 1$$

$$\sum_{j \in \delta^-(i)} g_{ji} - \sum_{j \in \delta^+(i)} g_{ij} = 1 \text{ for all } i \in V(G) \setminus \{1\}$$

$$g_{ij} \leq (n-1)x_e \text{ for every edge } e = \{i, j\}$$

$$g_{ji} \leq (n-1)x_e \text{ for every edge } e = \{i, j\}$$

$$\sum_{e \in E} x_e = n - 1$$

$$x_e + x_f \leq 1 \; \forall \; \{e, f\} \in P$$

$$g_{ij} \geq 0, \; g_{ji} \geq 0, x_e = 0 \text{ or } 1 \text{ for all } e = \{i, j\} \in E(G).$$

Solving CILP using CPLEX with a time limit of 1000 seconds was used as yet another feasibility test. By increasing the permitted running time, we used CILP to explore exact optimal solutions also.

2.6 Computational results

The experiments were conducted primarily on two machines: on a Dell PC with 3.40 GHz Intel pentium processor and 2.0 GB memory running Windows XP operation system and a Dell workstation with a 2.0 Ghz Intel Xeon processor and 512 MB of memory running the Linux operating system. The lower bound algorithms, feasibility tests, and heuristics were coded in C++. The public domain compiler Dev C++ was used for all compilation works on the PC and GNU gcc 3.6 compiler was used on the workstation. All general integer programs were solved using CPLEX 9.1 on the workstation. The goals of this preliminary experimental study were to (1) identify relative strengths of our lower bounds; (2) examine the strength of the lower bounds in relation to heuristic upper bounds; (3) examine the relative quality of the heuristic solutions and (4) assess the quality of the infeasibility tests.

There are no standard benchmark problems available for MSTC. Thus random test instances are generated as follows. First, a random connected graph G is constructed with the number of nodes ranging from 10 to 300 and the number of edges from 20 to 1000. Edge costs are random integers in the interval [0, 500]. Note that the conflict set determines if the problem is feasible or not. Given a random graph G, let P be the conflict set and $p = |P|$. If p is large, the corresponding MSTC is likely to be infeasible. Keeping this in mind, we have generated two types of conflict sets and the resulting problem instances are classified as Type 1 and Type 2. For Type 1 problems, p is selected randomly as an integer in the range $\left[\lceil \frac{m}{100} \rceil, 15\lceil \frac{m}{100} \rceil\right]$ where m is the number of edges. Thus the number of conflict pairs in this class is between 1% and 15% of the total number of edges. There is no guarantee that these problems are feasible. For the Type 2 class of problems, we make sure that the instances generated are feasible. For this, we first choose $p \in \left[25\lceil \frac{m}{100} \rceil, 35\lceil \frac{m}{100} \rceil\right]$ and generate a random conflict set P^*. The resulting MSTC instance is solved using local search with a random spanning tree as the starting solution and the objective function $g(T)$ is chosen to guide the search. If a feasible solution is obtained, then P is chosen as P^*. Otherwise, conflict pairs violated by the resulting solution are deleted from P^* and the remaining conflict pairs form the set P. Thus, the Type 2 instances generated are guaranteed to be feasible.

Experiments are conducted using the following algorithms or combination of algorithms:

1. Sub+MST: subgradient algorithm solving Lagrangian relaxation problem of ILP to compute lower bound;

2. Sub+MI: subgradient algorithm solving Lagrangian relaxation problem of ILP-MI to compute lower bound; To obtain the conflict pairs such that the corresponding conflict graph is a collection of disjoint cliques, we use the heuristic algorithm [77] to compute a maximum clique within our greedy algorithm for Max-ECP.

3. LS: local search algorithm using the Sub+MST solution as the starting solution;

4. TT: tabu thresholding algorithm with the spanning tree obtained by solving

$$\text{Minimize} \sum_{\{e,f\} \in P} (x_e + x_f)$$

$$\text{Subject to}$$

$$x \in \mathcal{F} \tag{2.13}$$

   as the starting solution. Default parameter values are set as Max-Iter = 10, δ=10, and ω=15.

5. TS: tabu search algorithm with the same starting solution as that of Tabu Thresholding. Default parameter values are set as Tabu-size =7 and Max-Iter = 100.

6. Infeasibility sets: CP test, MIS test, MVS test, and CILP with a time limit.

Note that Sub+MST and Sub+MI computes lower bounds. We set the maximum number of subgradient iterations to 100, and the Lagrangian multipliers are updated using standard way of using small step lengths.

In the class of Type 1 problems, 85 instances are generated using different values of the triplet (n, m, p) where n is the number of nodes, m is the number of arcs and p is the cardinality of P. From the experiments on these instances, possible outcomes are: (1) some problems are identified as infeasible, (2) some problems for which an ILP solver/heuristic was able to compute an optimal/feasible solution and (3) feasibility is not known for some problems. In the first set of experiments, we applied our infeasibility tests. This eliminated several Type 1 problems from further consideration.

The MIS-test solves the integer program

$$\text{Maximize} \sum_{e \in V(\hat{G})} x_e$$

Subject to

$$x_e + x_f \leq 1$$

using CPLEX with a time limit of 1000 seconds. Also, the algorithm is terminated when all active nodes have an upper bound value less than $n-1$ with a certificate of infeasibility. At any stage, if the algorithm obtained an independent set of size $n-1$ or larger, we terminate the algorithm with a flag that the MIS test failed. If none of these termination criterion is satisfied after 1000 seconds of CPU time was elapsed, the algorithm is terminated with the flag that MIS test failed. Instead of solving the integer program, we also experimented with its linear programming relaxation and the test was not very effective, but the integer programming version as explained above identified infeasibility of several problems. The MVC test solves the integer program

$$\text{Minimize} \sum_{e \in V(\hat{G})} x_e$$

Subject to

$$x_e + x_f \geq 1$$

using CPLEX with a time limit of 1000 seconds. As in the case of MIS test, the algorithm is terminated when all active nodes have a lower bound value less than $m - n + 1$ with a certificate of infeasibility. At any stage if the algorithm obtained a vertex cover of size $m-n+1$ or larger, we terminate the algorithm with a flag that the MVC test failed. If none of these termination criterion is satisfied after 1000 seconds of CPU time, the algorithm is terminated with the flag that MVC test failed. As in the case of MIS test, linear programming relaxation of this integer program was not very successful as a good bound to use in the test but the integer version, as discussed above, proved to be very effective.

Out of the 85 Type 1 test problems generated, 30 turned out to be provably infeasible. These instances are tabulated in Table 2.1. In the table, a "-" indicates that the infeasibility test failed and the number in a columns shows the CPU time for the corresponding test so

n	m	p	CP test	MIS test	MVC test	CPLEX
50	200	1990	-	-	722.68	618.22
50	200	2985	-	6.7	7.42	9.34
100	300	2242	-	64.73	67.85	65.09
100	300	4484	0.375	0.15	0.16	0.83
100	300	6726	0.484	0.22	0.22	0.64
200	400	798	-	-	-	0.12
200	400	1596	0.625	0.07	0.06	0.15
200	400	2394	0.593	0.08	0.09	0.15
200	400	3990	0.615	0.11	0.11	0.17
200	400	7980	0.735	0.26	0.26	0.25
200	400	11970	0.875	0.47	0.44	0.35
200	600	8985	-	1.61	1.63	6.25
200	600	17970	2.000	0.96	0.93	4.38
200	600	26955	2.375	1.82	1.87	4.06
200	800	31960	-	25.17	29.79	49.68
200	800	47940	5.297	12.49	10.33	56.92
300	600	1797	2.079	0.07	0.09	0.21
300	600	3594	1.803	0.09	0.13	0.25
300	600	5391	1.750	0.15	0.17	0.3
300	600	8985	1.687	0.25	0.29	0.37
300	600	17970	2.297	0.7	0.78	0.55
300	600	26955	2.291	1.39	1.58	0.87
300	800	6392	-	768.89	598.14	2.97
300	800	9588	4.406	0.56	0.62	0.45
300	800	15980	4.422	0.72	0.68	0.53
300	800	31960	5.109	1.82	1.87	0.87
300	800	47940	5.531	3.47	3.52	1.20
300	1000	24975	-	18.46	18.05	60.96
300	1000	49950	8.829	7.32	7.42	58.79
300	1000	74925	11.281	6.48	6.25	40.53

Table 2.1: Type 1 problems - Infeasible

that the problem is conclusively identified as infeasible. The column CPLEX in the table corresponds to solving CILP using the integer programming solver CPLEX with a time limit of 1000 seconds.

From the experimental results it can be seen that all infeasibility tests performed reasonably well. Infeasibility is detected in majority of cases in less than 50 seconds for most problems. In Table 2.2, we summarize experimental results on Type 1 problems where a feasible solutions is obtained. For the two lower bound algorithms and heuristics, we recorded the bounds 'LB' and 'UB' along with CPU times. The column 'CPLEX' contains the optimal objective function value 'obj' and CPU time for solving CILP with a time limit of 5000 seconds using CPLEX. The gap parameter 'Gap(%)' for SUB+MST and SUB+MI

were calculated by $\frac{Obj-LB}{Obj}\%$. The lower bound obtained by SUB+MI is consistently superior to that of SUB+MST. However, its computation time is significantly larger. To solve the matroid intersection problems, we used the algorithm of [31]. We believe an efficient implementation of the matroid intersection algorithm and good quality fast heuristics for solving Max-ECP could result in better running times. Nevertheless, we believe that both these algorithms are useful in developing specialized branch and bound algorithms taking advantage of the strengths of each in a hybrid way.

As the table shows, the heuristics algorithms LS, TT, and TS produced good quality solutions. The performance of TT and TS are somewhat similar, but TS seems slightly better. Interestingly, the solutions obtained by LS for the problems (100,300,448), (100,500,1247) and (100,500, 2495) are better than that of TT and TS. This may appear counterintuitive, but note that LS uses a special starting solution generated by SUB+MST. This suggests that the solution produced by SUB+MST is a good candidate starting solution and could be used to initiate any local search, including TT and TS or should be considered as one of the starting solutions in a multi-start version of TT and TS. Detailed analysis of various fine-tuning mechanisms and enhancements of TS and TT are beyond the scope of this preliminary experimental study. Such a study will be interesting, especially when relevant practical applications warrant that. The optimality of the problems marked ('#') were not achieved within the time limit of 5000 seconds provided to CPLEX and we recorded the best upper bound. It may be noted that the tabu search heuristic was able to find a feasible solution in all cases. For the problem (100,500,3741) tabu search found a heuristic solution in 35 seconds but CPLEX could not find a feasible solution in 5000 seconds.

Computational results on the remaining Type 1 problems are summarized in Table 2.3. For these problems, CPLEX and our heuristics failed to produce a feasible solution. We give the number of violated conflict pairs in the column 'vio' for each heuristic, along with the lower bound values. We are glad to provide our test problems for anyone interested in performing further computational study on MSTC.

Table 2.4 summarizes the results on Type 2 problems, where the problems are known to be feasible. LS, TT and TS returned optimal solutions for every problem in this category, except one (when (n, m, p)=(200,800,62625)). Optimality was verified using CPLEX. Unlike Table 2.2, where the problems were random, CPLEX had better running time on many problems in this class, compared to the heuristics but for some problems it took more time. In fact CPLEX detected feasibility and optimality in the preprocessing stage itself. This may

n	m	p	LB-MST			LB-MI			Heuristics						Opt.	
			LB	CPU	Gap(%)	LB	CPU	Gap(%)	LS		TT		TS		Obj	CPU
									UB	CPU	UB	CPU	UB	CPU		
50	200	199	701.089	0.156	0.98	702.793	10.453	0.74	708	0.157	735	2.172	711	2.563	708	41.22
50	200	398	739.838	0.204	3.92	757.816	10.828	1.58	797	0.265	789	2.594	785	2.484	770	51.08
50	200	597	782.67	0.453	14.65	807.745	51.203	11.91	-	0.438	1044	2.609	1086	2.141	917	42.93
50	200	995	835.68	0.500	36.88	877.495	51.641	33.72	1424	0.625	1721	1.984	1629	2.531	1324	162.59
100	300	448	3893.48	1.844	3.65	3991.18	301.601	1.23	4102	1.906	4316	14.156	4207	13.859	4041	801.51
#100	300	897	4508.16	1.796	-	4624.24	418.613	-	-	3.844	-	11.907	-	13.125	6523	5000
100	500	1247	4124.53	3.297	3.52	4165.68	794.078	2.56	4293	4.703	4913	28.985	4539	38.782	4275	1364.66
#100	500	2495	4701.87	3.125	-	4805.40	753.453	-	6603	6.375	7959	31.031	6812	37.422	6653	5000
#100	500	3741	4743.83	3.735	-	4871.27	781.235	-	-	8.641	10066	28.407	8787	32.719	-	5000

Table 2.2: Type 1 problems - Feasible

n	m	p	LB-MST		LB-MI		Heuristics					
			LB	CPU	LB	CPU	LS		TT		TS	
							vio	CPU	vio	CPU	vio	CPU
100	300	1344	4520.42	2.313	4681.27	349.72	23	3.187	14	16.922	13	14.829
100	500	6237	4724.35	4.828	4968.99	733.25	23	10.203	14	32.156	11	33.219
100	500	12474	4624.59	6.984	5194.67	921.39	49	15.141	41	22.204	41	32.063
200	600	1797	1111.6	13.484	11425.8	4448.64	15	32.156	2	186.408	2	158.721
200	600	3594	11896.9	11.563	12487	5456.88	74	78.204	65	193.845	67	153.83
200	600	5391	11953.7	12.453	12873.2	5947.77	177	68.501	149	178.126	149	175.471
200	800	3196	17428.8	17.313	17922.6	6640.04	5	52.313	1	246.845	2	230.409
200	800	6392	19061.3	19.078	19705.7	8193.83	63	88.36	32	298.955	39	214.643
200	800	9588	19229.6	19.797	20684.8	8429	126	139.532	105	253.689	95	212.659
200	800	15980	18778.1	25.609	20226.9	9020.98	189	173.626	180	239.283	178	229.8
300	800	3196	28617.2	36.126	30190.1	19021.5	89	274.064	61	569.16	63	524.788
300	1000	4995	39407.2	47.735	40732.7	26828.3	61	311.215	39	940.771	38	663.649
300	1000	9990	40748.4	46.422	42902.5	28421.5	231	670.119	191	919.522	207	756.104
300	1000	14985	40729.9	57.297	44639.1	27509.5	355	755.3	342	1188.62	351	835.011

Table 2.3: Type 1 problems - Feasibility unknown

be because of the way we forced feasibility for this class of problems which generated large number of bridges and tabu search/tabu threshholding could not take advantage of this while CPLEX could. The SUB+MI lower bound is significantly superior to SUM+MST lower bound on this class problems as well. However, as observed and discussed earlier, the computation time for our implementation of SUB+MI is not very attractive. For the problem (200,800,62625) CPLEX failed to terminate with an optimality certificate within our time limit. For this problem, the objective function value given is the best solution obtained and the CPU time is the time taken to reach this solution.

			LB-MST			LB-MI			Opt.		CPU-Heu		
n	m	p	LB	CPU	Gap(%)	LB	CPU	Gap(%)	obj.	CPU	LS	TT	TS
10	20	86	57.943	0.047	21.70	65.386	0.045	11.64	74	0.187	0.015	0.047	0.093
10	30	182	93.220	0.140	51.45	132.915	0.641	30.77	192	0.813	0.093	0.078	0.141
10	40	190	102.71	0.109	61.82	144.28	0.766	46.36	269	2.281	0.188	0.076	0.142
10	45	475	67.487	0.031	54.40	88.774	0.797	40.04	148	2.514	0.016	0.047	0.156
50	200	3903	775.118	1.047	52.62	877.467	81.281	46.37	1636	3.69	1.328	2.219	2.891
50	200	4877	698.676	1.719	65.80	887.478	78.015	56.56	2043	0.51	1.657	2.454	4.234
50	200	5864	626.918	1.328	73.18	1030.25	80.781	55.93	2338	0.74	2.5	2.094	3.422
100	300	8609	4043.38	4.407	45.61	5754.85	636.137	22.59	7434	10.67	4.766	15.627	17.578
100	300	10686	3970.52	3.875	50.17	6192.29	567.48	22.29	7968	4.65	5.328	11.891	16.36
100	300	12761	3936.27	4.672	51.80	6758.57	585.84	17.24	8166	1.38	5.406	16.641	17.266
100	500	24740	4289.85	14.687	66.09	5104.900	1032.32	59.65	12652	942.73	30.047	33.236	42.251
100	500	30886	3972.68	24.938	64.63	5078.820	952.713	54.78	11232	1333.48	37.454	32.564	36.719
100	500	36827	3918.06	34.375	65.87	5710.770	860.776	50.26	11481	264.43	49.688	24.798	33.203
200	400	13660	14085.8	4.203	20.54	17245.9	427.185	2.72	17728	0.38	4.313	62.204	66.298
200	400	17089	14067.3	5.625	24.44	18048.2	443.482	3.06	18617	0.47	4.828	55.064	84.61
200	400	20470	13998.7	9.797	26.86	18646.2	550.566	2.58	19140	0.53	10.844	49.422	58.345
200	600	34504	9466.060	47.985	54.31	15393.1	4880.88	25.69	20716	657.52	65.313	128.158	180.659
200	600	42860	9100.640	67.172	49.51	13971.5	5138.65	22.49	18025	192.67	60.813	114.5	210.584
200	600	50984	8734.530	76.032	58.14	16708.1	5313.14	19.92	20864	545.49	73.031	132.704	268.644
*200	800	62625	16806.500	115.641	57.87	23792.300	7705.11	40.36	39895	29370.95	275.674	219.049	256.956
200	800	78387	15803.100	140.173	58.05	22174.200	7464.64	41.14	37671	1008.87	296.83	236.673	233.284
8200	800	93978	15470.100	170.313	60.13	24907.000	7421.59	35.80	38798	11523.08	280.939	197.048	230.878
300	600	31000	34154.200	38.266	21.88	42720.6	2626.15	2.29	43721	0.77	40.01	200.532	245.347
300	600	38216	33320.100	21.563	24.73	43486.7	1710.3	1.76	44267	0.96	21.266	246.971	337.004
300	600	45310	32072.300	11.266	25.54	42149.0	1395.09	2.14	43071	1.16	12.765	155.673	216.799
300	800	59600	24384.300	117.876	43.46	36629.600	19945.3	15.06	43125	1.48	165.72	442.581	469.756
300	800	74500	22913.200	140.36	45.82	38069.300	18248.7	9.98	42292	1.79	144.016	298.783	419.662
300	800	89300	21624.600	41.188	50.98	38843.000	17969.2	11.95	44114	2.13	46.001	405.378	536.819
300	1000	96590	36544.700	217.376	48.93	56048.300	30428.1	21.68	71562	32.71	700.583	531.503	659.789
300	1000	120500	34380.800	253.658	54.97	58780.100	27779.3	23.01	76345	82.45	729.676	571.191	629.602
300	1000	144090	33481.200	317.83	57.55	60810.800	27788.1	22.91	78880	99.38	627.207	616.598	712.103

Table 2.4: Type 2 problems

Chapter 3

The Maximum Edge Clique Partitioning Problem

3.1 Introduction

Let $G = (V, E)$ be a graph on n nodes. Then *the maximum edge clique partitioning problem (Max-ECP)* is to find a partition $V_1, V_2, \cdots, V_t$ of V such that the subgraph $G_i = (V_i, E_i)$ of G induced by V_i is a clique for $1 \leq i \leq t$ and $\sum_{i=1}^{t} |E_i|$ is maximized.

As an example, take G as the graph (a) in Figure 3.1, then graph (b) and (c) are two different clique partitions of G, with size 9 and 10 respectively.

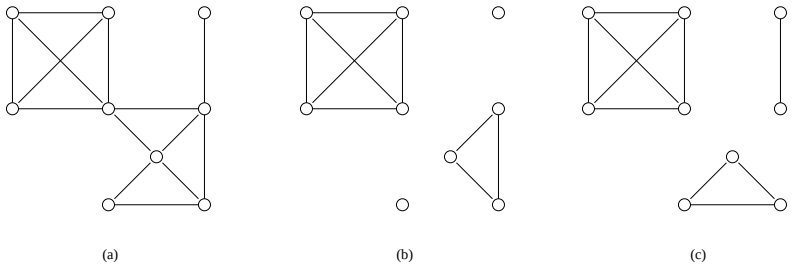

Figure 3.1: Example: edge clique partitioning

As illustrated in Chapter 2, we could obtain a lower bound for the MSTC by first identifying a clique partition of the conflict graph, and solving a weighted matroid intersection problem on the partition. Therefore, the quality of the MSTC lower bound is significantly effected by the Max-ECP solution and thus we are motivated to look into the Max-ECP in

this chapter.

It is well known that the Max-ECP is NP-hard and cannot be approximated within a factor of $n^{1-O(1/(\log n)^{\gamma})}$ unless $NP \subseteq ZPTIME(2^{(\log n)^{O(1)}})$, for any fixed γ [158]. Dessmark et al.[68] showed that a maximum cardinality matching in G provides an ϱ-optimal solution to the Max-ECP where ϱ is the largest cardinality of a clique in G. This is the best known performance ratio for a polynomial time approximation algorithm for the problem. Since ϱ can be $O(n)$, the best known data independent performance ratio for the Max-ECP is $O(n)$. The bound ϱ can be slightly improved in a data dependent way, by modifying the proof of performance ratio $O(\varrho)$ given in [68] and exploiting properties of possible alternative optimal solutions.

Let $\{\{V_1^k, V_2^k, \ldots, V_{m_k}^k\},\ k = 1, 2, \ldots, p\}$ be the set of all the optimal solutions to the Max-ECP. For each k, define $\delta^k = \max_{1 \le j \le m_k} |V_j^k|$. Let $\delta = \min_{1 \le k \le p} \delta^k$.

Lemma 11. *If $M \subseteq E$ is a maximum cardinality matching in G and OPT is the optimum objective function value of the Max-ECP on G, then $|M| \ge \frac{OPT}{\delta}$.*

Proof. Let $\{V_1, V_2, \ldots, V_t\}$ be an optimal solution to the Max-ECP such that $\max_{1 \le i \le t} |V_i| = \delta$. Let $M^0 = \cup_{i=1}^{t} M_i^0$, where M_i^0 is a maximum cardinality matching in the subgraph of G induced by V_i. Then

$$\begin{aligned} |M| \ge |M^0| &= \sum_{i=1}^{t} |M_i^0| \\ &\ge \sum_{i=1}^{t} \frac{|V_i| - 1}{2} \\ &\ge \sum_{i=1}^{t} \frac{|V_i|}{\delta} \left(\frac{|V_i| - 1}{2} \right) \\ &\ge \frac{1}{\delta} \sum_{i=1}^{t} |V_i| \left(\frac{|V_i| - 1}{2} \right) = \frac{1}{\delta} OPT. \end{aligned}$$

□

Note that $\delta \le \varrho$. As in the case of ϱ, the ratio δ could also be $O(n)$. However, the advantage of Lemma 11 is that it links the performance ratio to the smallest value of the size of the largest clique in an optimal clique partition. Let us now discuss how to improve this bound.

3.2 The 2-phase approximate greedy algorithm

We consider a variation of the greedy algorithm [68], called *the 2-phase approximate greedy algorithm*. In phase 1, we iteratively extract a clique D^k using an approximate algorithm for the maximum clique problem such that $|V(D^k)|$ is no less than p for a given integer $p \geq 2$. The process is continued until no such cliques can be identified and the algorithm switches to phase 2. In phase 2, since the remaining graph contains only cliques with size at most $p-1$, an exact maximum clique can be found in polynomial time for fixed p. We keep extracting maximum cliques until the edge set of the remaining graph becomes empty. A formal description of the algorithm is given below. Here Approx-Max-Clique(G) is a procedure which accepts G as input and outputs an ϵ-optimal solution for the maximum clique problem on G.

Algorithm 3.1: The 2-phase Approximate Greedy Algorithm

Input: $G = (V, E)$;
Phase 1:
$H^1 = \emptyset$, $H^2 = \emptyset$, G^1=G, $k = 0$;
while G^{k+1} contains cliques of size $\geq p$ **do**
  $k = k + 1$;
  D^k = Approx-Max-Clique(G^k);
  If $|V(D^k)| < p$ **then** choose a clique of size p in G^k and designate it D^k;
  $H^1 = H^1 \cup D^k$;
  $G^{k+1} = G^k \setminus V(D^k)$;
end while;
Phase 2:
$k = k + 1$;
while $E(G^k) \neq \emptyset$ **do**
  $\bar{D}^k$ = Max-Clique(G^k); /* Max-Clique (G) computes a maximum clique in G */
  $H^2 = H^2 \cup \bar{D}^k$;
  $G^{k+1} = G^k \setminus V(\bar{D}^k)$;
  $k = k + 1$;
end while;
$H = H^1 \cup H^2$;
Output: $H \cup G^k$.

Note that the maximum clique extraction in phase 2 can be done in $O(n^p)$ time by complete enumeration and hence it is polynomial for fixed p. Also the condition of the while

loop in phase 1 can be verified in $O(n^p)$ time by complete enumeration. Thus the algorithm is polynomially bounded whenever Approx-Max-Clique(G) in phase 1 is polynomially bounded. In the 2-phase approximate greedy algorithm, if we let $p = 2$, then phase 2 is redundant.

Lemma 12. *Let $a_1, a_2, \cdots, a_n$ be n non-negative real numbers with mean $\bar{a} > 1$. Then $\sum_{i=1}^{n} a_i \leq \frac{1}{\bar{a}-1} \sum_{i=1}^{n} a_i(a_i - 1)$*

Proof. By the sum of squares inequality, we have $\frac{1}{n}(\sum_{i=1}^{n} a_i)^2 \leq (\sum_{i=1}^{n} a_i^2)$. Subtracting $\sum_{i=1}^{n} a_i$ from both sides and simplifying, we get $(\sum_{i=1}^{n} a_i)(\frac{1}{n}\sum_{i=1}^{n} a_i - 1) \leq \sum_{i=1}^{n}(a_i^2 - a_i)$. Note that $\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$ and hence $(\bar{a}-1)\sum_{i=1}^{n} a_i \leq \sum_{i=1}^{n} a_i(a_i - 1)$. Since $\bar{a} > 1$ the result follows. □

Theorem 18. *If Approx-Max-Clique(G) computes an ϵ-optimal solution to the maximum clique problem on G, then the 2-phase approximate greedy algorithm gives a $\frac{2(\alpha\epsilon-1)}{\alpha-1}$-optimal solution to Max-ECP on G, where α is the average size of cliques extracted in phase 1.*

Proof. Let $Q^1 = (Q_1^1, Q_2^1, \cdots, Q_t^1)$ be an optimal solution to the Max-ECP on G. Also, let r be the number of iterations in phase 1 and s be the total number of iterations in the algorithm. In each iteration k of phase 1, let $|V(D^k)| = d^k$ and ρ^k be the size of a maximum clique in G^k. Since D^k is an ϵ-optimal solution to the maximum clique problem on G^k, $d^k \leq \rho^k \leq \epsilon d^k$ for $k \leq r$. Let $Q^k = (Q_1^k, Q_2^k, \cdots, Q_t^k)$ for $1 \leq k \leq s$ and $Q_i^{k+1} = Q_i^k - V(D^k)$ for $1 \leq k \leq r$. Note that for $k \leq r$, $|E(Q_i^k)| - |E(Q_i^{k+1})| \leq |V(Q_i^k) \cap V(D^k)|(\rho^k - 1)$ if $V(Q_i^k) \cap V(D^k) \neq \emptyset$ and $|E(Q_i^k)| - |E(Q_i^{k+1})| = 0$ if $V(Q_i^k) \cap V(D^k) = \emptyset$. Also, $\sum_{i=1}^{t} |V(Q_i^k) \cap V(D^k)| = d^k$ for $k \leq r$. Thus

$$|E(Q^k)| - |E(Q^{k+1})| = \sum_{i=1}^{t}(|E(Q_i^k)| - |E(Q_i^{k+1})|) \leq d^k(\rho^k - 1) \leq d^k(\epsilon d^k - 1), \text{ for } k \leq r. \tag{3.1}$$

Adding (3.1) for $k = 1$ to r we get

$$\sum_{k=1}^{r} \Big(|E(Q^k)| - |E(Q^{k+1})|\Big) \leq \sum_{k=1}^{r} d^k(\epsilon d^k - 1). \tag{3.2}$$

In phase 2, an exact maximum clique $\bar{D}^k$ is extracted in each iteration k, For $r < k \leq$

$s-1$, let $Q_i^{k+1} = Q_i^k - V(\bar{D}^k)$ and now, $\sum_{i=1}^{t} |V(Q_i^k) \cap V(\bar{D}^k)| = \rho^k$ for $r < k \leq s$. Thus

$$|E(Q^k)| - |E(Q^{k+1})| \leq \rho^k(\rho^k - 1), \text{ for } k = r+1, \ldots, s. \tag{3.3}$$

Adding (6.1) from $k = r+1$ to s we have

$$\sum_{k=r+1}^{s} \Big(|E(Q^k)| - |E(Q^{k+1})|\Big) \leq \sum_{k=r+1}^{s} \rho^k(\rho^k - 1). \tag{3.4}$$

From (3.2) and (3.4) we get

$$\begin{aligned}
\sum_{k=1}^{s}(|E(Q^k)| - |E(Q^{k+1})|) &\leq \sum_{k=1}^{r} d^k(\epsilon d^k - 1) + \sum_{k=r+1}^{s} \rho^k(\rho^k - 1) \\
&= \epsilon \sum_{k=1}^{r} d^k(d^k - 1 + 1 - \frac{1}{\epsilon}) + \sum_{k=r+1}^{s} \rho^k(\rho^k - 1) \\
&= \epsilon \sum_{k=1}^{r} d^k(d^k - 1) + (\epsilon - 1)\sum_{k=1}^{r} d^k + \sum_{k=r+1}^{s} \rho^k(\rho^k - 1).
\end{aligned}$$

By definition, $\alpha = \frac{1}{r}\sum_{k=1}^{r} d^k$ and using Lemma 12 we have,

$$\begin{aligned}
\sum_{k=1}^{s}(|E(Q^k)| - |E(Q^{k+1})|) &\leq \epsilon \sum_{k=1}^{r} d^k(d^k - 1) + \frac{\epsilon - 1}{\alpha - 1}\sum_{k=1}^{r} d^k(d^k - 1) + \sum_{k=r+1}^{s} \rho^k(\rho^k - 1) \\
&= \frac{\alpha\epsilon - 1}{\alpha - 1}\sum_{k=1}^{r} d^k(d^k - 1) + \sum_{k=r+1}^{s} \rho^k(\rho^k - 1) \\
&= \frac{2(\alpha\epsilon - 1)}{\alpha - 1}|E(H^1)| + 2|E(H^2)| \\
&\leq \frac{2(\alpha\epsilon - 1)}{\alpha - 1}(|E(H^1)| + |E(H^2)|) \\
&= \frac{2(\alpha\epsilon - 1)}{\alpha - 1}|E(H)|.
\end{aligned}$$

On the left hand side,

$$\sum_{k=1}^{s}(|E(Q^k)| - |E(Q^{k+1})|) = |E(Q^1)| - |E(Q^{s+1})| = |E(Q^1)| \text{ since } E(Q^{s+1}) = \emptyset.$$

Therefore, $|E(Q^1)| \leq \frac{2(\alpha\epsilon-1)}{\alpha-1}|E(H)|$, where $|E(Q^1)|$ is the optimal objective function value of the Max-ECP on G and $|E(H)|$ is the objective function value of the approximate solution returned by Algorithm 3.1. □

The bound established above contains α which is data dependent. Let us now consider a data independent bound.

Lemma 13. *For $a \geq b > 1$ and $\epsilon \geq 1$, $\frac{a\epsilon-1}{a-1} \leq \frac{b\epsilon-1}{b-1}$.*

Proof. If $\epsilon = 1$, the proof is trivial. Assume $\epsilon > 1$. Since $a \geq b > 1$, we have $1 - \frac{1}{a} \geq 1 - \frac{1}{b}$ and hence $\frac{a}{a-1} \leq \frac{b}{b-1}$. Also $\epsilon > 1$. Thus $\frac{a}{a-1}(\epsilon - 1) + 1 \leq \frac{b}{b-1}(\epsilon - 1) + 1$. Simplifying this inequality yields the required result. □

Note that the average size of the cliques extracted in phase 1 is at least p. Thus $\alpha \geq p \geq 2$. From Lemma 13 and Theorem 18, we have the performance ratio of the 2-phase approximate greedy algorithm is bounded by $\frac{2(p\epsilon-1)}{p-1}$. Therefore, the 2-phase approximate greedy algorithm has a performance ratio bound of $4\epsilon-2$ when $p = 2$ and $3\epsilon-1$ when $p = 3$. As the value of α increases, the performance ratio bound of Algorithm 3.1 approaches to 2ϵ, especially, when $\epsilon = 1$, we get a performance ratio of 2 and in this case, the algorithm reduces to the greedy algorithm of Dessmark et al [68] and hence Theorem 18 is a proper generalization of the corresponding result of [68]. Since the maximum clique problem can be approximated within a factor of $\frac{Kn(\log\log n)^2}{(\log n)^3}$ for an appropriate constant K [77], Max-ECP can be approximated within a factor of $\frac{2}{p-1}(p\frac{Kn(\log\log n)^2}{(\log n)^3} - 1)$ for any fixed integer $p \geq 2$. This improves the best known performance bound for a polynomial time approximation algorithm for Max-ECP.

Note that the maximum clique problem on a graph can be solved in polynomial time by complete enumeration if its clique number is fixed. However it appears that such an approach doesn't extend to Max-ECP.

Theorem 19. *Max-ECP is NP-hard on graphs with clique number no more than 3 but solvable in polynomial time on graphs with clique number no more than 2.*

Proof. We reduce the 3-dimensional matching problem(3DM) to Max-ECP on a graph with clique number no more than 3. Let $X,\ Y,\ Z$ be disjoint sets with $|X| = |Y| = |Z| = n$ and $S \subseteq X \times Y \times Z$. Then the 3DM is to find a subset M of S such that for any two distinct triples (x_1, y_1, z_1) and (x_2, y_2, z_2) in M we have $x_1 \neq x_2,\ y_1 \neq y_2$, and $z_1 \neq z_2$ and $|M| = n$.

Given an instance X, Y, Z, S of 3DM, construct a tripartite graph G with partite sets X, Y, and Z. For each triple $(x, y, z) \in S$, there are edges (x, y), (y, z) and (x, z) in G. Clearly the clique number of G is no more than 3. Note that $3n$ is an upper bound on the optimal objective function value of Max-ECP on G and a solution to max-ECP achieves this value precisely when the given instance of 3DM has a 3-dimensional matching M with $|M| = n$. Since 3DM is NP-hard [95] Max-ECP is also NP-hard.

For graphs with clique number no more than two, it can be verified that a maximum cardinality matching solves Max-ECP. □

3.3 The maximum edge bi-clique partitioning problem

As a variation of the Max-ECP, let us define a *bi-clique* as a complete bipartite graph $G = (V_1, V_2, E)$, where we assume $|V_1| \leq |V_2|$. Then *the maximum edge bi-clique partitioning problem (Max-EBCP)* is to find a partition $V_{11}, V_{12}, V_{21}, V_{22}, \cdots, V_{t1}, V_{t2}$ of V such that $G_i = (V_{i1}, V_{i2}, E_i)$ of G induced by V_{i1} and V_{i2} is a bi-clique for $1 \leq i \leq t$ and $\sum_{i=1}^{t} |E_i|$ is maximized.

Again we consider G as graph (a) in Figure 3.2, a bi-clique partition is given by graph (b). Note that singletons are counted as bi-cliques of size 0.

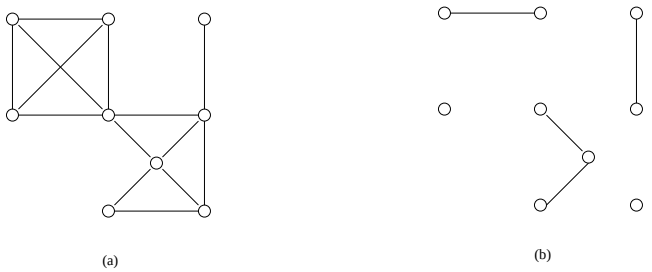

Figure 3.2: Example: edge bi-clique partitioning

Since a maximum cardinality matching in G can be viewed as a bi-clique partition, we could solve the maximum cardinality matching problem on G and use it as an approximate solution to the Max-EBCP. Let $\{\{V_{11}^k, V_{12}^k, V_{21}^k, V_{22}^k \ldots, V_{m_k 1}^k, V_{m_k 2}^k\},\ k = 1, 2, \ldots, p\}$ be the set of all the optimal solutions to the Max-EBCP. For each k, define $\delta^k = \max_{1 \leq j \leq m_k} |V_{j2}^k|$. Let $\delta = \min_{1 \leq k \leq p} \delta^k$. Then similar to Lemma 11, we have the following result for the Max-EBCP.

Lemma 14. *If $M \subseteq E$ is a maximum cardinality matching in G and OPT is the optimum objective function value of the Max-EBCP on G, then $|M| \geq \frac{OPT}{\delta}$.*

Proof. Let $\{V_{11}, V_{12}, V_{21}, V_{22} \ldots, V_{t1}, V_{t2}\}$ be an optimal solution to the Max-EBCP such that $\max_{1 \leq i \leq t} |V_{i2}| = \delta$. Let $M^0 = \cup_{i=1}^{t} M_i^0$, where M_i^0 is a maximum cardinality matching in the subgraph of G induced by V_{i1} and V_{i2}. Then

$$\begin{aligned} |M| \geq |M^0| &= \sum_{i=1}^{t} |M_i^0| \\ &= \sum_{i=1}^{t} |V_{i1}| \\ &\geq \sum_{i=1}^{t} \frac{|V_{i2}|}{\delta} |V_{i1}| \\ &\geq \frac{1}{\delta} \sum_{i=1}^{t} |V_{i1}||V_{i2}| = \frac{1}{\delta} OPT. \end{aligned}$$

□

As described in Algorithm 3.2, a greedy algorithm is also established to solve the Max-EBCP, where in each iteration, a subroutine is used to find the maximum bi-clique of the reduced graph G^k.

Algorithm 3.2: The Greedy Algorithm

Input: $G = (V, E)$;
$H = \emptyset$, G^1=G, k=1;
while $E(G^k) \neq \emptyset$ **do**
  D^k= Max-Bi-Clique(G^k);
  $H = H \cup \{D^k\}$;
  $G^{k+1} = G^k - V(D^k)$;
  $k = k + 1$;
end while;
Output: $H \cup \{G^k\}$.

In a Max-EBCP, if we consider only balanced bi-cliques, i.e. $|V_{i1}| = |V_{i2}|$ for $i = 1, \cdots, t$, then the problem is called *the maximum edge balanced bi-clique partitioning problem.*

Theorem 20. *The greedy algorithm computes a $4 - \frac{4}{n}$ optimal solution to the maximum edge balanced bi-clique partitioning problem on G, where n is the number of nodes of G.*

Proof. Let $Q^1 = (Q_1^1, Q_2^1, \cdots, Q_t^1)$ be an optimal solution to the Max-EBCP on G and s be the number of iterations in the algorithm. Here for convenience of calculating the number edges in a balanced bi-clique, we assume that in each iteration k, $|V(D^k)| = 2d^k$, then $|E(D^k)| = (d^k)^2$. Let ρ^k be the size of a maximum balanced bi-clique in G^k. Since D^k is an exact solution to the maximum balanced bi-clique problem on G^k, $\rho^k = 2d^k$ for $k = 1, 2, \cdots, s$. Let $Q^k = (Q_1^k, Q_2^k, \cdots, Q_t^k)$ for $1 \leq k \leq s$. Note that for $k \leq s$, if $V(Q_i^k) \cap V(D^k) \neq \emptyset$

$$\begin{aligned} |E(Q_i^k)| - |E(Q_i^{k+1})| &\leq |V(Q_i^k) \cap V(D^k)| \left((\frac{\rho^k}{2})^2 - (\frac{\rho^k - 2}{2})^2 \right) \\ &= |V(Q_i^k) \cap V(D^k)|(\rho^k - 1) \end{aligned}$$

and $|E(Q_i^k)| - |E(Q_i^{k+1})| = 0$ if $V(Q_i^k) \cap V(D^k) = \emptyset$. Also, $\sum_{i=1}^t |V(Q_i^k) \cap V(D^k)| = \rho^k$ for $1 \leq k \leq s$. Thus

$$|E(Q^k)| - |E(Q^{k+1})| \leq \rho^k(\rho^k - 1), \text{ for } k = 1, \cdots, s. \tag{3.5}$$

Adding (3.5) from $k = 1$ to s we have

$$\begin{aligned} \sum_{k=1}^{s} \Big(|E(Q^k)| - |E(Q^{k+1})| \Big) &\leq \sum_{k=1}^{s} \rho^k(\rho^k - 1) \\ &= \sum_{k=1}^{s} \rho^k \cdot \rho^k - \rho^k = \sum_{k=1}^{s} (\rho^k)^2 - \frac{1}{\rho^k}(\rho^k)^2 \\ &\leq \sum_{k=1}^{s} (\rho^k)^2 - \frac{1}{n}(\rho^k)^2 \\ &= (4 - \frac{4}{n}) \sum_{k=1}^{s} |E(D^k)| = (4 - \frac{4}{n})|E(H)|. \end{aligned}$$

On the left hand side,

$$\sum_{k=1}^{s}(|E(Q^k)| - |E(Q^{k+1})|) = |E(Q^1)| - |E(Q^{s+1})| = |E(Q^1)| \text{ since } E(Q^{s+1}) = \emptyset.$$

Therefore, $|E(Q^1)| \leq (4-\frac{4}{n})|E(H)|$, where $|E(Q^1)|$ is the optimal objective function value of the Max-EBCP on G and $|E(H)|$ is the objective function value of the approximate solution returned by Algorithm 3.2. □

However, for the general Max-EBCP, Algorithm 3.2 could result in an arbitrarily bad solution. To see this, let us consider a graph as shown in Figure 3.3.

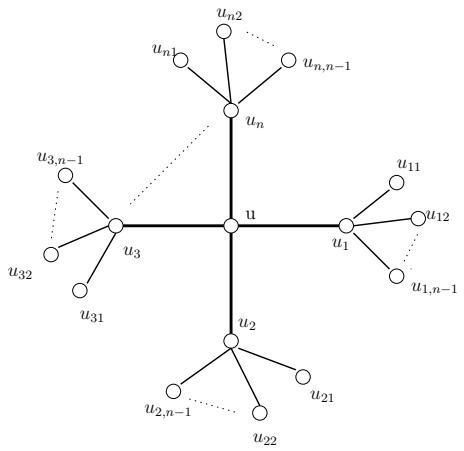

Figure 3.3: Example: Greedy algorithm for the Max-EBCP

It is obvious that an optimal solution to the Max-EBCP of this graph is $\cup_{i=1}^{n}(V_{i1} \cup V_{i2})$, where $V_{11} = \{u\}$, $V_{12} = \{u_1, u_{11}, u_{12}, \cdots, u_{1,n-1}\}$, $V_{i1} = \{u_i\}$, $V_{i2} = \{u_{i1}, u_{i2}, \cdots, u_{i,n-1}\}$ for $i = 2, \cdots, n$ and the total number of edges in this bi-clique partition is $n + (n-1)^2$. But the greedy algorithm could return $\{u\} \cup \{u_1, u_2, \cdots, u_n\}$ in the first iteration, then the remaining graph is a collection of singletons $u_{11}, u_{12}, \cdots, u_{nn}$ and thus the number of edges in the bi-clique partition is n. Therefore, the approximate solution returned by the Algorithm 3.2 could be arbitrarily bad as n grows.

Chapter 4

The Quadratic Bottleneck Combinatorial Optimization Problem

4.1 Introduction

We first restate the problem QCOP and QBCOP. Let $E = \{1, 2, \cdots, m\}$, $\mathcal{F}$ be the family of feasible solutions $\mathcal{F}$ and $q(e, f)$ be a given real valued cost for each $(e, f) \in E \times E$, then *the quadratic combinatorial optimization problem (QCOP)* is to:

$$\text{Minimize} \sum_{e \in S} \sum_{f \in S} q(e, f)$$

$$\text{Subject to}$$

$$S \in \mathcal{F}$$

and *the quadratic bottleneck combinatorial optimization problem (QBCOP)* is to:

$$\text{Minimize } \max\{q(e, f) : e \in S, f \in S\}$$

$$\text{Subject to}$$

$$S \in \mathcal{F}$$

The $m \times m$ matrix Q whose $(i,j)^{th}$ entry has value $q(i,j)$ is called *the cost matrix* associated with the QCOP(QBCOP). Also we refer q as *the cost function* defined on $E \times E$. In this chapter, we first consider the QBCOP where the cost matrix Q is symmetric and the diagonal entries are zeros, i.e. if we denote $P[\Theta] = \{\{e,f\} : e \in \Theta, f \in \Theta, e \neq f\}$ for any subset Θ of E, then it can be viewed that the cost function q is defined on $P[E]$. Later on in Section 4.3, we show that this assumption can be made without loss of generality.

We have introduced Theorem 1 in Chapter 1 showing that any bottleneck combinatorial optimization problem (BCOP) can be formulated as a linear combinatorial optimization problem (LCOP) by using exponentially large costs. Extending this result, we now observe that any QBCOP can be formulated as a quadratic combinatorial optimization problem (QCOP). For each feasible solution $S \in \mathcal{F}$ let $g(S) = \max\{q(\{e,f\}) \ : \ \{e,f\} \in P[S]\}$. Let $z^1 < z^2 < \cdots < z^p$ be an ascending arrangement of distinct costs from $\{q(\{e,f\}) : \{e,f\} \in P[E]\}$ and $F^r = \{S \in \mathcal{F} : g(S) = z^r\}$ for $r = 1, 2, \cdots, p$. Also, let $U_r = \bigcup_{i=1}^{r} F^i = \{S \in \mathcal{F} : g(S) \leq z^r\}$ for $r = 1, \cdots, p$.

Let h be another real valued cost function defined on $P[E]$.

Theorem 21. *If $h(\{e,f\})$ satisfies*

$$\min\{\sum_{\{e,f\} \in P[S]} h(\{e,f\}) : S \in F^r\} > \min\{\sum_{\{e,f\} \in P[S]} h(\{e,f\}) : S \in U_{r-1}\}$$

for $2 \leq r \leq p$, then every optimal solution to the QCOP with cost function h is also an optimal solution to the QBCOP with cost function q.

Proof. If k is the smallest index such that F^k is non-empty, then it is obvious that any $S \in F^k$ is an optimal solution to the QBCOP with the optimal objective function value z^k. Now assume that S' is an optimal solution to the QCOP with cost function h and $S' \in F^q$ for some q, then we have

$$\begin{aligned}\sum_{\{e,f\} \in P[S']} h(\{e,f\}) &= \min\{\sum_{\{e,f\} \in P[S]} h(\{e,f\}) : S \in F^q\} \\ &> \min\{\sum_{\{e,f\} \in P[S]} h(\{e,f\}) : S \in U_{q-1}\}.\end{aligned}$$

Thus U_{q-1} must be empty and therefore S' is an optimal solution to QBCOP. □

The cost function h satisfying the condition of the above theorem can be constructed

easily by modifying the results for the bottleneck traveling salesman problem discussed in [144]. For instance, let $\gamma_1 = 0$, and for $j = 2, \cdots, p$ let $\gamma_j = m^2\gamma_{j-1} + 1$. Define

$$h(\{e,f\}) = \begin{cases} 0 & \text{if } q(\{e,f\}) \leq z^1 \\ \gamma_j & \text{if } q(\{e,f\}) = z^j,\ j = 2, \cdots, p, \end{cases}$$

then every optimal solution to the QCOP with cost function h is an optimal solution to QBCOP with cost function q.

By Theorem 21, the QBCOP can be solved by solving a single QCOP. However, to satisfy the condition of the theorem, $h(\{e,f\})$ often grows exponentially in the problem size m, and hence it is difficult to use the theorem directly with computational advantage.

4.2 Weak duality and the quadratic threshold algorithm

As mentioned in Chapter 1, a family of subsets of E is called *a clutter* if no member of the family is contained in another member of the family. Thus without loss of generality we assume that $\mathcal{F}$ is a clutter. For the clutter $\mathcal{F}$ on E, its blocking clutter (blocker) is the unique clutter $\mathcal{B} = \{S \subseteq E : |S \cap T| \geq 1\ \forall T \in \mathcal{F} \text{ and } S \text{ is minimal}\}$.

Let $\pi : P[E] \to N$ be a one to one correspondence where $N = \{1, 2, \ldots, r\}$ and $r = \binom{m}{2}$. For each $S \in \mathcal{F}$, consider the subset S' of N defined by $S' = \{\pi(e,f) : \{e,f\} \in P[S]\}$. Let $\mathcal{H} = \{S' : S \in \mathcal{F}\}$. Thus $\mathcal{H}$ is a family of subsets of N. In this way the QBCOP becomes a BCOP on the system $(N, \mathcal{H})$ and hence the duality theorem of Edmonds and Fulkerson [73] is applicable for an appropriate clutter - blocker pair defined for $(N, \mathcal{H})$. However, the structure of this clutter-blocker pair in terms of the original system $(E, \mathcal{F})$ is not clearly understood. Burkard [34] made this observation and stated that "further understanding of the structure of this clutter-blocker pair (in terms of the original clutter-blocker pair) could lead to significant algorithmic developments".

We establish a weak duality result for the QBCOP. For any $S \subseteq E, T \subseteq E$ denote $P[S,T] = \{\{e,f\} : e \in S, f \in T, e \neq f\}$.

Theorem 22. *For any clutter $\mathcal{F}$ and its blocker $\mathcal{B} = b(\mathcal{F})$ on E*

$$\max_{\substack{R,S \in \mathcal{B} \\ R \neq S}} \min_{\{e,f\} \in P[R,S]} q(\{e,f\}) \leq \min_{T \in \mathcal{F}} \max_{\{e,f\} \in P[T]} q(\{e,f\})$$

Proof. By definition, for any $T \in \mathcal{F}$ and any $R, S \in \mathcal{B}$ we have $T \cap R \neq \emptyset$ and $T \cap S \neq \emptyset$. Thus

$$\min_{\{e,f\} \in P[R,S]} q(\{e,f\}) \leq \max_{\{e,f\} \in P[T]} q(\{e,f\}).$$

Since T, R and S are arbitrary, we have

$$\max_{\substack{R,S \in \mathcal{B} \\ R \neq S}} \min_{\{e,f\} \in P[R,S]} q(\{e,f\}) \leq \min_{T \in \mathcal{F}} \max_{\{e,f\} \in P[T]} q(\{e,f\})$$

□

The weak duality theorem discussed above has duality gap as illustrated in the following example. Consider the graph G on four nodes given in Figure 4.1. Choose $\mathcal{F}$ as the collection of all spanning trees of G. Its blocker $\mathcal{B}$ is the collection of all cuts in G.

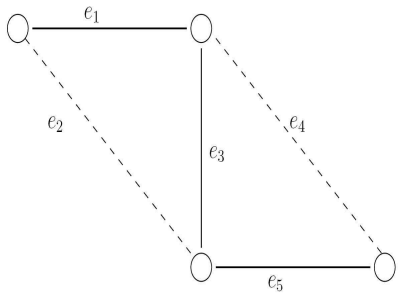

Figure 4.1: Example: duality gap for the QBCOP

Let $e, f \in \{e_1, e_2, e_3, e_4, e_5\}$ and the matrix

$$Q = \begin{bmatrix} 0 & 2 & 0 & 2 & 1 \\ 2 & 0 & 2 & 0 & 2 \\ 0 & 2 & 0 & 2 & 0 \\ 2 & 0 & 2 & 0 & 2 \\ 1 & 2 & 0 & 2 & 0 \end{bmatrix}$$

gives the cost of pairs of edges of G. It may be verified that

$$\min_{T \in \mathcal{F}} \max_{\{e,f\} \in P[T]} q(\{e,f\}) = 1$$

whereas

$$\max_{\substack{R,S \in \mathcal{B} \\ R \neq S}} \min_{\{e,f\} \in P[R,S]} q(\{e,f\}) = 0$$

We now discuss an algorithm to solve the QBCOP, which generalizes the threshold algorithm for the BCOP in Chapter 1. Let $z_1 < z_2 < \cdots < z_p$ be an ascending arrangement of distinct costs $\{q(\{e,f\}) : \{e,f\} \in P[E]\}$. For any non-negative integer k, let $Q(k) = \{\{e,f\} \in P[E] : q(\{e,f\}) > z_k\}$. Thus $Q(p) = \emptyset$. Let $F_k = \{T \in \mathcal{F} : P[T] \cap Q(k) = \emptyset\}$. In other words, F_k contains all solutions in F with the QBCOP objective function value less than or equal to z_k. Note that $F_1 \subseteq F_2 \subseteq \cdots \subseteq F_p$ where $F_p = \mathcal{F}$.

Theorem 23. *For $1 \leq k \leq p$, if k is the largest index such that $F_k = \emptyset$ then any $S \in F_{k+1}$ is an optimal solution to the QBCOP. Further if $F_t = \emptyset$ for some t then $F_k = \emptyset$ for all $k \leq t$.*

The proof of the above theorem is straightforward. The QBCOP feasibility problem (FQBCOP) defined in Chapter 1 can now be described in a slightly different way: "Given $1 \leq k \leq p$, is F_k empty?" i.e. the FQBCOP(k) verifies that for a given $k, 1 \leq k \leq p$ and the family $\mathcal{F}$, does there exist a feasible solution $S \in \mathcal{F}$ which does not contain both e and f for any set $\{e,f\} \in Q(k)$? Note that such an S may contain either e or f or none, but not both.

Then similar to the threshold algorithm for the BCOP, the QBCOP can be solved as a sequence of FQBCOPs by performing binary search over $z_1 < z_2 < \cdots < z_p$. A formal description of our algorithm is given below.

Theorem 24. *The QBCOP can be solved in polynomial time if and only if the the FQBCOP is polynomially solvable. Further, the quadratic threshold algorithm computes an optimal solution to the QBCOP in $O(\phi(m) \log m)$ time, where $O(\phi(m))$ is the complexity of the FQBCOP(k).*

Proof. Suppose that the FQBCOP can be solved in polynomial time. By theorem 23, it can be verified that Algorithm 4.1 correctly solves the QBCOP. To establish the complexity, note that the algorithm tests if $F_k = \emptyset$ by invoking FQBCOP(k). Such a test is performed at most

Algorithm 4.1: Quadratic Threshold Algorithm

1: **Input:** A family $\mathcal{F}$ (in compact form), the cost function q and an oracle $\alpha(.)$ which with input k verifies if $F_k = \emptyset$ and output an 'yes' or 'no' answer along with an $S \in F_k$ whenever $F_k \neq \emptyset$.
2: Construct an ascending arrangement $z_1 < z_2 < \cdots < z_p$ of distinct costs $\{q(\{e, f\}) : \{e, f\} \in P[E]\}$.
3: $\ell = 1$; $u = p$;
4: **while** $u - \ell > 0$ **do**
5: $\quad k = \lfloor \frac{(l+u)}{2} \rfloor$;
6: $\quad$ **if** $F_k = \emptyset$ **then** $\ell = k + 1$; **else** $u = k$;
7: **end while**
8: **Output** any $T \in F_\ell$

$O(\log(p))$ times. Since $p = O(m^2), O(\log(p)) = O(\log(m^2)) = O(\log m)$, the complexity of the algorithm is $O(\phi(m) \log m)$. Thus if $\phi(m)$ is polynomial then the QBCOP is solvable in polynomial time.

Conversely, suppose the QBCOP is solvable in polynomial time. Let S^* be an optimal solution. Choose t such that $z_t = f(S^*)$. Then $F_k = \emptyset$ precisely when $k < t$. Thus the quadratic feasibility problem can be solved in polynomial time □

Although Theorem 24 appears to be promising, even for the case where the family of feasible solutions have a very simple structure, the quadratic feasibility problem (i.e. verifying if $F_k = \emptyset$) may be difficult. We illustrate this by considering the quadratic bottleneck problem with only cardinality restrictions. Suppose $\mathcal{F}$ is the family of all subsets of E with cardinality at least t for some given t. We call the resulting QBCOP, *the cardinality constrained QBCOP* (CQBCOP). The corresponding quadratic feasibility problem is: "Given an instance of the CQBCOP, represented by E and t, verify if there exist an $S \in \mathcal{F}$ such that $g(S) \leq K$ for a given constant K." where $g(S) = \max\{q(\{e, f\}) : \{e, f\} \in P[S]\}$.

Theorem 25. *The CQBCOP feasibility problem is NP-complete.*

Proof. We reduce the maximum independent set problem [95] to the CQBCOP. Let G be a graph on which the maximum independent set problem is defined. Construct an instance of the CQBCOP as follows. Choose the ground set E as the node set of G and

$\mathcal{F} = \{S : S \subseteq V(G), |S| \geq t\}$. Construct the cost $q(\{e, f\})$ for pairs of elements of E as

$$q(\{e,f\}) = \begin{cases} 1 & \text{if the nodes } e \text{ and } f \text{ are adjacent in } G \\ 0 & \text{otherwise} \end{cases}$$

Choose $K = 0$. Now G has an independent set of size greater than or equal to t if and only if there exists an $S \in \mathcal{F}$ such that $g(S) \leq 0$. Since the maximum independent set problem is NP-complete [95], the result follows. □

4.3 Polynomially solvable cases

Although the problem CQBCOP appears to be the simplest non trivial QBCOP, Theorem 25 shows that even this problem is hard. Despite this disappointment, it can be shown that there are non-trivial cases of QBCOP that can be solved in polynomial time.

One way to achieve polynomial solvability is by restricting the structure of $q(\{e, f\})$. Before considering specially structured $q(\{e, f\})$ let us briefly discuss four different variations of QBCOP. The QBCOP we discussed so far is called *uniform symmetric* QBCOP where no cost is considered for single elements of E. In addition to pairs of elements of E, if there is a cost associated for each element $e \in E$, then we have an instance of the *non-uniform symmetric* QBCOP. In this case we represent the cost of $e \in E$ as $q(\{e, e\})$ and the domain of the cost function as $P[E] \cup \{\{e, e\} : e \in E\}$. When the domain of the cost function q is $E \times E$ we have the *non-uniform asymmetric* QBCOP. In this case, to reflect the fact that (e, f) is an ordered pair, we write $q(e, f)$ in place of $q(\{e, f\})$. Finally, when the domain of the cost function is restricted to $E \times E \setminus \{(e, e) : e \in E\}$ we get the *uniform asymmetric* QBCOP. The following lemma shows that the four variations of QBCOP discussed above are equivalent in the sense that given any two variations, one can be reduced to another.

Lemma 15. *The four variations of the QBCOP discussed above are equivalent.*

Proof. Consider a non-uniform asymmetric QBCOP with cost function q_1. Define the non-uniform symmetric cost function q_2 as $q_2(\{e, f\}) = \max\{q_1(e, f), q_1(f, e)\}$. It can be verified that an optimal solution to the non-uniform symmetric QBCOP with cost function q_2 is also an optimal to the asymmetric QBCOP with cost function q_1. Consider the uniform symmetric cost function q_3 defined as $q_3(\{e, f\}) = \max\{q_1(e, f), q_2(f, e), q_1(e, e), q_1(f, f)\}$ for $e \neq f$. It can be verified that an optimal solution to the uniform symmetric QBCOP

with cost function q_3 is also an optimal to the non-uniform asymmetric QBCOP with cost function q_1. Reduction among other pairs of variations can be established by appropriate modifications of the above arguments and hence the details are omitted. □

Based on Lemma 15 without loss of generality one may assume that the cost functions are of uniform symmetric type. We concentrate on uniform cases primarily for notational clarity and continue to consider only uniform symmetric cases in all other sections. However, a distinction between the four versions of QBCOP is maintained in this section which is important for specially structured cost functions that we consider. In particular, maintaining asymmetric problems, without reducing them to the symmetric case is advantageous in some cases, as we illustrate later in this section.

A non-uniform asymmetric cost function q is said to be *decomposable* if there exist real numbers a_e and b_e for each $e \in E$ such that $q(e, f) = a_e + b_f$ for all $e \in E, f \in E$. A non-uniform symmetric cost function q is said to be *decomposable* if there exist a real number a_e for each $e \in E$ such that $q(\{e, f\}) = a_e + a_f$ for all $e, f \in E$.

Theorem 26. *A non-uniform asymmetric QBCOP with a decomposable cost function q is solvable in $O(m\zeta(m))$ time if an associated BCOP can be solved in $O(\zeta(m))$ time.*

Proof. For any feasible solution S

$$\begin{aligned} g(S) &= \max\{q(e, f) \ : \ e \in S, f \in S\} \\ &= \max\{a_e + b_f \ : \ e \in S, f \in S\} \\ &= \max\{a_e \ : \ e \in S\} + \max\{b_f \ : \ f \in S\} \end{aligned}$$

Thus $g(S)$ decomposes into sum of two maximum functions. Then minimizing $g(S)$ over $\mathcal{F}$ is a special case of the ξ-deviation problem [70]. Using the results of [70], minimization of $g(S)$ can be done by solving $O(m)$ linear bottleneck problems of the type

$$\underset{S \in \mathcal{F}}{\text{Minimize}} \ \max\{b_e : e \in S\}.$$

Thus QBCOP can be solved in $O(m\zeta(m))$ time. □

We thus specialize the algorithm given in [70] to solve our QBCOP and the details are described in Algorithm 4.2.

Algorithm 4.2: An Algorithm for the QBCOP with non-uniform, asymmetric, decomposable cost function

Input: A set E with cardinality m, the family $\mathcal{F}$ of feasible solutions, $a_1, \cdots, a_m$, $b_1, \cdots, b_m$;
Let $b_1 \leq b_2 \leq \cdots \leq b_m$ be an ascending arrangement of b's;
Let $p = m, k = m, z^* = \infty$.
Repeat
    Solve the linear bottleneck problem

$$\min_{S \in \mathcal{F}} \{\max a_e : \text{ for } \forall\ e \in S \text{ and } b_e \leq b_p\}$$

    Let S^0 be the optimal solution and $z(S^0) = \max_{e \in S^0} a_e + \max_{e \in S^0} b_e$;
    if $z(S^0) < z^*$, let $S^* = S^0$, $z^* = z(S^0)$;
    Let k be the minimal with $b_k = \max\{b_e : e \in S^0\}$;
    Set $b_e = \infty$ for $e = k$ to p;
    Let $p = k - 1$;
Until $p = 0$ or the linear bottleneck problem is infeasible
Output an optimal solution S^* to the QBCOP with objective function value z^*.

It may be noted that when $b_e = -a_e$, the the non-uniform asymmetric QBCOP with decomposable cost function reduces to *the balanced optimization problem* [45, 174].

To take advantage of Theorem 26 we need a way to test if a given cost function is decomposable or not. We say that an $m \times m$ matrix $Q = (q(e_i, e_j))$ is *decomposable* if and only if q is a decomposable function. Here to simplify the notation we write $q(e_i, e_j)$ as $q(i, j)$. The following lemma allows us to test for decomposability of a cost function.

Lemma 16. *Let Q be an $m \times m$ matrix with $(i, j)^{th}$ entry $q(i, j)$. Let $\hat{Q} = \hat{q}(i, j)$ be defined as $\hat{q}(i, j) = q(i, j) - (a_i + b_j)$, where $a_i = q(i, m) - \frac{q(m,m)}{2}$, $b_j = q(m, j) - \frac{q(m,m)}{2}$. Then Q is decomposable if and only if $\hat{q}(i, j) = 0$ for $i, j = 1, \ldots, m$.*

Proof. Suppose Q is decomposable, then $\exists \alpha_i,\ \beta_i,\ i = 1, \ldots, m$ such that $q(i, j) = \alpha_i + \beta_j$. Now

$$\begin{aligned}
\hat{q}(i, j) = q(i, j) - (a_i + b_j) &= q(i, j) - \left(q(i, m) - \frac{q(m, m)}{2} + q(m, j) - \frac{q(m, m)}{2} \right) \\
&= q(i, j) - q(i, m) - q(m, j) + q(m, m) \\
&= \alpha_i + \beta_j - \alpha_i - \beta_m - \alpha_m - \beta_j + \alpha_m + \beta_m \\
&= 0.
\end{aligned}$$

Conversely, suppose $\hat{q}(i,j) = q(i,j) - (a_i + b_j) = 0$, then clearly $q(i,j) = a_i + b_j$ holds and hence Q is decomposable. □

Conditions similar to Lemma 16 have been studied in the context of traveling salesman problem by various authors [87, 143, 145].

Note that q is decomposable if and only if its associated cost matrix $Q = (q(i,j))$ is decomposable and hence decomposability of q can be tested in $O(m^2)$ time. Based on Lemma 15, one might wonder why asymmetric cost functions need to be considered at all. To answer this, consider the matrices Q^1 and Q^2 given by

$$Q^1 = \begin{bmatrix} 3 & 8 & 3 & 4 & 7 \\ 2 & 7 & 2 & 3 & 6 \\ 8 & 13 & 8 & 9 & 12 \\ 6 & 11 & 6 & 7 & 10 \\ 5 & 10 & 5 & 6 & 9 \end{bmatrix} \text{ and } Q^2 = \begin{bmatrix} 3 & 8 & 8 & 6 & 7 \\ 8 & 7 & 13 & 11 & 10 \\ 8 & 13 & 8 & 9 & 12 \\ 6 & 11 & 9 & 7 & 10 \\ 7 & 10 & 12 & 10 & 9 \end{bmatrix}$$

Now $q^1(i,j) = a_i + b_j$ with $(a_1, a_2, a_3, a_4, a_5) = (2, 1, 7, 5, 4)$ and $(b_1, b_2, b_3, b_4, b_5) = (1, 6, 1, 2, 5)$ and hence Q^1 is decomposable, but Lemma 16 guarantees that Q^2 is not decomposable. The matrix Q^2 is obtained from Q^1 by using the transformation described in Lemma 15. Recall that the non-uniform cost functions include costs for single elements and the cost of element i is denoted as $q^1(i,i)$ $(q^2(i,i))$. By Lemma 15, QBCOP with cost functions q^1 and q^2 are equivalent. But if we are simply presented the QBCOP with cost function q^2, we do not have an obvious easy way to solve it, without probably using the quadratic threshold algorithm, but when presented with cost function q^1, based on Theorem 26 it can be solved as a sequence of QCOPs. Thus, there are instances where it is advantageous to maintain the asymmetry of the cost function, without converting it into a symmetric problem.

Consider an asymmetric(symmetric) cost function q and the associated cost matrix Q. Suppose q is not decomposable. Let L be a lower bound and U be an upper bound on the optimal objective function value of QBCOP. Consider the linear inequality system

$$\begin{array}{lllll} \text{(LI)} & a_i + b_j < L, & \text{if } \max\{q(i,j), q(j,i)\} < L & \text{(r1)} \\ & a_i + b_j > U, & \text{if } \max\{q(i,j), q(j,i)\} > U & \text{(r2)} \\ & a_i + b_j = q(i,j) & \text{if } L \le q(j,i) \le q(i,j) \le U & \text{(r3)} \end{array}$$

$$a_i + b_j \leq q(j,i) \qquad \text{if } L \leq q(i,j) < q(j,i) \leq U \qquad \text{(r4)}$$

where $a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m$ are variables.

Suppose $a^0 = (a_1^0, a_2^0, \ldots, a_m^0)$, $b^0 = (b_1^0, b_2^0, \ldots, b_m^0)$ be a feasible solution to (LI). Let $q^0(i,j) = a_i^0 + b_j^0$

Lemma 17. *The QBCOP with cost functions q^0 and q are equivalent.*

Proof. Note that the QBCOP with cost function q^0 and q have the same solution set. For any feasible solution S, let $Z^1(S) = \max\{q^0(i,j) : i,j \in S\}$ and $Z^2(S) = \max\{q(i,j) : i,j \in S\}$. For any $i,j \in S$ clearly $Z^1(S) \geq \max\{q(i,j), q(j,i)\}$. Thus by construction, for any feasible solution S satisfying $L \leq Z^1(S) \leq U$, we have $Z^1(S) = Z^2(S)$. The result now follows from the fact that L and U are respectively lower and upper bounds on the optimal objective function value of the QBCOP. □

Since q^0 is decomposable, the QBCOP is polynomially solvable whenever the associated BCOP can be solved in polynomial time. In the example of the 5×5 matrix Q^2 given earlier, if we simply take the trivial lower bound $L = 3$ and trivial upper bound $U = 13$ the resulting (LI) is feasible with solution $a^0 = (2,1,7,5,4)$, $b^0 = (1,6,1,2,5)$. This verifies that we can get an equivalent decomposable problem from Q^2. The difference between the lower bound L and upper bound U can affect the number of solution to the system (LI). When L and U are closer, there are less number of equality constraints in (r3) and hence the feasible region of (LI) is likely to be larger. So if we have heuristics to obtain high quality lower and upper bounds, the QBCOP is more likely to be solved in polynomial time, by finding an equivalent decomposable cost function.

Algorithm 4.2 can be easily modified to solve the uniform asymmetric QBCOP with a decomposable cost function, and thus we have the following result.

Corollary 27. *A uniform asymmetric QBCOP with a decomposable cost function q is solvable in $O(m\zeta(m))$ time if an associated BCOP can be solved in $O(\zeta(m))$ time.*

Moreover, the other two QBCOP variations could also be nicely solvable with decomposable cost functions.

Theorem 28. *A non-uniform symmetric QBCOP with a decomposable cost function can be formulated as a BCOP. Further, a uniform symmetric QBCOP with a decomposable cost function can be formulated as a 2-sum optimization problem.*

Proof. Let us first consider the non-uniform case. For any feasible solution S for the QBCOP, from the proof of Theorem 26, we have

$$g(S) = \max\{a_e \ : \ e \in S\} + \max\{a_f \ : \ f \in S\} = 2\max\{a_e \ : \ e \in S\}.$$

Thus minimizing $g(S)$ is equivalent to minimizing $\max\{a_e \ : \ e \in S\}$, which is an BCOP.

When the cost function is uniform, the restriction $e \neq f$ is retained in the definition of $P[E]$ and $q(\{e, f\})$. Thus, in this case, minimizing $g(S)$ is equivalent to finding a solution S where the sum of the largest and second largest (counting multiplicity) value of a_x for $x \in S$ is minimized. This is precisely the 2-sum optimization problem [118, 206]. □

Note that a 2-sum optimization problem can be solved efficiently whenever an associated LCOP problem can be solved efficiently [118, 206].

Similarly, we define a non-uniform asymmetric cost function q as *a multiplicative cost function* (or equivalently, Q as a *multiplicative cost matrix*) if there exist real numbers a_e and b_e for each $e \in E$ such that $q(e, f) = a_e \cdot b_f$ for all $e \in E, f \in E$. A non-uniform symmetric cost function q is said to be *multiplicative* if there exist a real number a_e for each $e \in E$ such that $q(\{e, f\}) = a_e \cdot a_f$ for all $e, f \in E$.

Theorem 29. *A non-uniform asymmetric QBCOP with a multiplicative cost function q is solvable in $O(m\zeta(m))$ time if an associated BCOP can be solved in $O(\zeta(m))$ time.*

Proof. For any feasible solution S to the QBCOP

$$\begin{aligned} g(S) &= \max\{q(e, f) \ : \ e \in S, f \in S\} \\ &= \max\{a_e \cdot b_f \ : \ e \in S, f \in S\} \\ &= \max\{a_e \ : \ e \in S\} \cdot \max\{b_f \ : \ f \in S\} \end{aligned}$$

Thus $g(S)$ decomposes into multiplication of two maximum functions. Algorithm 4.2 can be easily modified to solve the QBCOP with a non-uniform, asymmetric and multiplicative cost function, simply by changing $z(S^0) = \max_{e \in S^0} a_e \max_{e \in S^0} b_e$ in step 5. Then this problem can then be solved as a sequence of $O(m)$ BCOP and thus the QBCOP can be solved in $O(m\zeta(m))$ time. □

Note that a zero matrix is multiplicative. Further, a matrix Q is multiplicative if and

only if $-Q$ is multiplicative. Thus to test the multiplicability of a matrix Q, we can focus our attention simply on nonzero matrices with at least one positive element.

Lemma 18. *Let $Q = (q(i,j))_{m\times m}$ be a nonzero matrix with at least one positive element, say $q(k,l)$, and $\hat{Q} = (\hat{q}(i,j))_{m\times m}$ be defined as $\hat{q}(i,j) = q(i,j) - \alpha_i\beta_j$, where $\alpha_i = \frac{q(i,l)}{\sqrt{q(k,l)}}$, $\beta_j = \frac{q(k,j)}{\sqrt{q(k,l)}}$. Then Q is multiplicative if and only if $\hat{q}(i,j) = 0$ for $i,j = 1,\ldots,m$.*

Proof. Suppose Q is multiplicative, then there exists $a_i,\ b_i,\ i = 1,\ldots,m$ such that $q(i,j) = a_ib_j$ for all i and j. Since $q(k,l) > 0$ and $q(k,l) = a_kb_l$, $a_k, b_l \neq 0$. Now

$$\begin{aligned}\hat{q}(i,j) = q(i,j) - \alpha_i\beta_j &= q(i,j) - \frac{q(i,l)}{\sqrt{q(k,l)}}\frac{q(k,j)}{\sqrt{q(k,l)}} = q(i,j) - \frac{q(i,l)q(k,j)}{q(k,l)} \\ &= a_ib_j - \frac{a_ib_la_kb_j}{a_kb_l} = 0.\end{aligned}$$

Conversely, suppose $\hat{q}(i,j) = q(i,j) - \alpha_i\beta_j = 0$, then clearly $q(i,j) = \alpha_i\beta_j$ holds and hence Q is multiplicative. □

Corollary 30. *If Q is a nonnegative symmetric matrix that is multiplicative, then there exists α_i, $i = 1,\cdots,m$ such that $q(i,j) = \alpha_i\alpha_j$ for $i,j = 1,\cdots,m$.*

Proof. The proof is trivial if Q is a zero matrix. Suppose Q is a symmetric, nonnegative and nonzero matrix that is multiplicative, then there exists $r \in \{1,\cdots,m\}$ such that $q(r,r) > 0$. Choosing $(k,l) = (r,r)$ in Lemma 18 we have $\alpha_i = \beta_i$ and hence $q(i,j) = \alpha_i\alpha_j$ for $i,j = 1,\cdots,m$. □

Corollary 31. *A uniform asymmetric QBCOP with a multiplicative cost function q is solvable in $O(m\zeta(m))$ time if an associated BCOP can be solved in $O(\zeta(m))$ time.*

Theorem 32. *A non-uniform symmetric QBCOP with a multiplicative cost function can be formulated as a BCOP.*

Proof. From Corollary 30, we have

$$g(S) = \max\{a_e\ :\ e \in S\}\cdot\max\{a_f\ :\ f \in S\} = (\max\{a_e\ :\ e \in S\})^2.$$

Thus minimizing $g(S)$ is equivalent to minimizing $\max\{a_e\ :\ e \in S\}$, which is an BCOP. □

If the cost function is uniform, symmetric and multiplicative, minimizing $g(S)$ reduces to find a solution S where the multiplication of the largest and second largest (counting multiplicity) value of a_e for $e \in S$ is minimized. This problem can be solved by modifying the corresponding algorithms for the 2-sum optimization problem.

Corollary 33. *A uniform symmetric QBCOP with a multiplicative cost function q is solvable in $O(m\zeta(m))$ time if an associated BCOP can be solved in $O(\zeta(m))$ time.*

Besides decomposable and multiplicative cases, we could also have a non-uniform, asymmetric cost matrix Q which is "separable" in a way that there exists a_e and b_e for $e \in E$ such that

$$q(e, f) = \max\{a_e, b_f\} \text{ for all } e \in E, f \in E, \tag{4.1}$$

then since

$$\begin{aligned} &\max\{q(e, f) : e \in S, f \in S\} \\ =&\max\{\max\{a_e, b_f\} : e \in S, f \in S\} \\ =&\max\{\max\{a_e, b_e\} : e \in S, f \in S\}, \end{aligned}$$

the QBCOP with such a Q can be reduced to a BCOP with $c_e = \{a_e, b_e\}$ for all e. Similarly, if

$$q(e, f) = \min\{a_e, b_f\} \text{ for all } e \in E, f \in E, \tag{4.2}$$

then the QBCOP can be solved by solving a BCOP with $c_e = \min\{a_e, b_e\}$ for all e. We call a cost matrix (function) satisfying property (4.1) or (4.2) *comparable cost matrix (function))* and it can be easily verify that the above result holds for the QBCOP with non-uniform symmetric, non-uniform asymmetric and non-uniform symmetric comparable cost matrices as well.

To obtain another polynomially solvable case of the QBCOP, let us consider the following bottleneck combinatorial optimization problem:

$$\begin{aligned} \text{BCOP}(e, Q)\text{: Minimize } &\max\{q(e, f) \ : \ f \in S\} \\ \text{Subject to} \\ &S \in \mathcal{F}. \end{aligned}$$

Let z^e be the optimal objective function value of the BCOP(e, Q). Now consider the BCOP:

$$\text{BCOP}(Q)\text{: Minimize } \max\{z^e \ : \ e \in S\}$$
$$\text{Subject to}$$
$$S \in \mathcal{F}.$$

Let $\tilde{L}$ be the optimal objective function value of the BCOP(Q).

Lemma 19. *$\tilde{L}$ is a lower bound for the optimal objective function value of the QBCOP.*

Proof. Let S be any feasible solution to the QBCOP. Note that S is also a feasible solution of the BCOP(Q). Then $\max\{q(e, f) : e \in S, f \in S\} = \max\limits_{e \in S} \max\limits_{f \in S} q(e, f) \geq \max\limits_{e \in S} z^e \geq \tilde{L}$. □

Clearly, if we can find a feasible solution S for the QBCOP with objective function value $\tilde{L}$, it is indeed optimal. This is possible when the cost matrix Q satisfies additional properties.

Let $Q = (q(e, f))$ be an $m \times m$ matrix. Then Q is said to be *row graded* if $q(e, 1) \leq q(e, 2) \leq \cdots \leq q(e, m)$ for all $e = 1, \cdots, m$. Further, Q is said to be *doubly graded* if Q and Q^T are row graded. An optimal solution $S^0 = (x_1^0, x_2^0, \cdots, x_n^0)$ to a BCOP is called a *least-index optimal solution* if S^0 is optimal and $r = \max\{i : x_i^0 = 1\} \leq \max\{i : x_i = 1\}$ for any optimal solution S for the BCOP. The value r is called *the critical index of the BCOP*.

Lemma 20. *Let S^0 be a least-index optimal solution to the BCOP$(1, Q)$ with critical index r. If Q is row graded and $q(1, r) \leq q(2, r) \leq \cdots \leq q(m, r)$, then S^0 is an optimal solution to the QBCOP with cost matrix Q and the corresponding optimal objective function value is $q(r, r)$.*

Proof. Since Q is row graded, S^0 is an optimal solution to BCOP(e, Q) for all $e = 1, \cdots, m$ with corresponding optimal objective function value $z^e = q(e, r)$. Since $q(1, r) \leq q(2, r) \leq \cdots \leq q(m, r)$, S^0 is also an optimal solution to the BCOP(Q) with optimal objective function value $q(r, r)$. By Lemma 19, $q(r, r)$ is a lower bound for the optimal objective function value of QBCOP. The QBCOP objective function value of S^0 can be verified to be $q(r, r)$ and the proof follows. □

Theorem 34. *Given a QBCOP with cost matrix Q and let r be the critical index of the least-index optimal solution to the BCOP$(1, Q)$, if Q is row graded and $q(1, r) \leq q(2, r) \leq \cdots \leq q(m, r)$, then the QBCOP can be solved in polynomial time whenever the corresponding BCOP is polynomially solvable.*

Note that all doubly graded matrices satisfy the conditions of Lemma 20. Further, QBCOP with cost matrices Q and Q^T are equivalent. Thus if the conditions of Lemma 20 are satisfied for Q^T, then also QBCOP can be solved in polynomial time. Lemma 20 can be extended to a larger class of cost matrices where the rows (and hence the columns) can be relabeled to obtain a matrix that satisfies conditions of the lemma.

In fact, a similar lower bound as $\tilde{L}$ for the QBCOP can be established for the QCOP, by solving the following LCOP:

$$\text{LCOP(Q): } \hat{L}=\text{Minimize } \sum_{e_i \in S} z^i$$
$$\text{Subject to}$$
$$S \in \mathcal{F},$$

where

$$\text{LCOP(i,Q): } z^i = \text{Minimize } \sum_{e_j \in S} q(i, j)$$
$$\text{Subject to}$$
$$S \in \mathcal{F}.$$

In the context of the QAP, $\hat{L}$ is precisely the Gilmore-Lawler type lower bound [101, 169]. Accordingly, we define *the critical optimal solution* of the LCOP as an optimal solution of the LCOP, whose elements are selected by following the lexicographic order of the indices. For example, if there are three optimal solutions $S^1 = \{e_2, e_3, e_7\}$, $S^2 = \{e_2, e_4, e_5\}$ and $S^3 = \{e_3, e_5, e_6\}$, then S^1 is the critical minimum spanning tree.

The LCOP-preserving matrix is then described as a row graded matrix Q with the property that the critical optimal solution of LCOP$(1, Q)$ is also optimal to LCOP(Q).

We next show that a similar result as the one stated in Theorem 34 holds for a special

case of the QCOP - *the quadratic minimum weight matroid problem (QMWM)* and the reverse is also true, which then yields a new characterization of matroid.

Given a $E = \{1, 2, \cdots, m\}$, let $\mathcal{F}$ be the family of feasible solutions and $I = \{X \subseteq S : S \in \mathcal{F}\}$. Assume for any $S \in \mathcal{F}$, $|S| = s$ is a constant. The structure (E, I) is a *matroid* [71] if and only if for any $S_1, S_2 \in \mathcal{F}$ and $e_1 \in S_1 \setminus S_2$ there exists $e_2 \in S_2 \setminus S_1$ such that $S_1 \setminus \{e_1\} \cup \{e_2\} \in F$. $\mathcal{F}$ is called *a matroid base*.

Theorem 35. *The following statements are equivalent:*

1. *(E, I) is a matroid.*
2. *If Q is a LCOP-preserving matrix, then the critical optimal solution of LCOP(i, Q) is optimal to the QCOP with E, $\mathcal{F}$ and cost matrix Q.*

Proof. If (E, I) is a matroid, we let S^0 be the critical optimal solution of LCOP(i, Q), then it is a feasible solution of the QCOP. Since Q is row graded and S^0 is a critical optimal, it is also optimal for LCOP(i, Q) for $i = 2, \cdots, m$, with corresponding optimal objective function value $z^i = \sum_{j \in S^0} q(i, j)$. As S^0 is also optimal for LCOP(Q), $z(S^0) = \hat{L}$ and the optimality of S^0 to the QCOP is proved.
To show (2) implies (1), we assume that (E, I) is not a matroid when (2) is true. Then there exists $S_1 \in \mathcal{F},\ S_2 \in \mathcal{F}$ and $e_1 \in S_1 \setminus S_2$ such that $S_1 \setminus \{e_1\} \cup \{e\} \notin \mathcal{F}$ for any $e \in S_2 \setminus S_1$. Now E can be partitioned as $E = \{e_1\} \cup (S_1 \setminus \{e_1\}) \cup (S_2 \setminus S_1) \cup (E \setminus (S_1 \cup S_2))$, and we construct a matrix $Q = (q(i, j))_{m \times m}$ in the following way:

$$q(i,j) = \begin{cases} 1 + \frac{3}{2}\Delta & \text{if } e_i = e_j = e_1; \\ 1 + \Delta & \text{if } e_i = e_1, e_j \in S_2 \setminus S_1 \text{ or } e_i \in S_2 \setminus S_1, e_j = e_1; \\ 1 & \text{if } e_i = e_1, e_j \in S_1 \setminus \{e_1\}, \text{ or } e_i \in S_1 \setminus \{e_1\}, e_j = e_1, \text{ or } e_i, e_j \in S_2 \setminus S_1; \\ 1 - \Delta & \text{if } e_i \in S_1 \setminus \{e_1\}, e_j \in S_2 \setminus S_1 \text{ or } e_i \in S_2 \setminus S_1, e_j \in S_1 \setminus \{e_1\}; \\ 1 - \frac{3}{2}\Delta & \text{if } e_i, e_j \in S_1 \setminus \{e_1\}; \\ M & \text{if } e_i \text{ or } e_j \in E \setminus (S_1 \cup S_2), \end{cases}$$

where M is a large number and $\Delta > 0$ is a small number. Clearly Q is symmetric and row graded, so it is LCOP-preserving. It is easy to see that S_1 is the unique optimal solution of LCOP$(1, Q)$. In the QCOP, let $f(S) = \sum_{e_i \in S} \sum_{e_j \in S} q(i, j)$, then

$$
\begin{aligned}
f(S_1) &= w(1,1) + \sum_{e_j \in S_1 \setminus \{e_1\}} w(1,j) + \sum_{e_i \in S_1 \setminus \{e_1\}} w(i,1) + \sum_{e_i, e_j \in S_1 \setminus \{e_1\}} w(i,j) \\
&= 1 + \frac{3}{2}\Delta + 2|S_1 \setminus \{e_1\}| + (1 - \frac{3}{2}\Delta)|S_1 \setminus \{e_1\}|^2 \\
&= 1 + \frac{3}{2}\Delta + 2(s-1) + (1 - \frac{3}{2})(s-1)^2 \\
&= s^2 + 3\Delta s - \frac{3}{2}\Delta s^2 = s^2 + 3\Delta s(\frac{s}{2} - 1) \geq s^2
\end{aligned}
$$

$$
\begin{aligned}
f(S_2) &= \sum_{\substack{e_i \in S_1 \cap S_2 \\ e_j \in S_1 \cap S_2}} w(i,j) + \sum_{\substack{e_i \in S_1 \cap S_2 \\ e_j \in S_2 \setminus S_1}} w(i,j) + \sum_{\substack{e_i \in S_2 \setminus S_1 \\ e_j \in S_1 \cap S_2}} w(i,j) + \sum_{\substack{e_i \in S_2 \setminus S_1 \\ e_j \in S_2 \setminus S_1}} w(i,j) \\
&= (1 - \frac{3}{2}\Delta)|S_1 \cap S_2|^2 + 2(1-\Delta)|S_1 \cap S_2||S_2 \setminus S_1| + |S_2 \setminus S_1|^2 \\
&= (1 - \frac{3}{2}\Delta)|S_1 \cap S_2|^2 + (2 - 3\Delta)|S_1 \cap S_2||S_2 \setminus S_1| + \Delta|S_1 \cap S_2||S_2 \setminus S_1| \\
&\quad + (1 - \frac{3}{2}\Delta)|S_2 \setminus S_1|^2 + \frac{3}{2}\Delta|S_2 \setminus S_1|^2
\end{aligned}
$$

$$
\begin{aligned}
&= (1 - \frac{3}{2}\Delta)\left(|S_2 \setminus S_1| + |S_2 \setminus S_1|\right)^2 + \Delta|S_2 \setminus S_1|(|S_1 \cap S_2| + \frac{3}{2}|S_2 \setminus S_1|) \\
&= (1 - \frac{3}{2}\Delta)|S_2|^2 + \Delta|S_2 \setminus S_1|(|S_2| + \frac{1}{2}|S_2 \setminus S_1|) \\
&= (1 - \frac{3}{2}\Delta)s^2 + \Delta|S_2 \setminus S_1|(s + \frac{1}{2}|S_2 \setminus S_1|) \\
&< (1 - \frac{3}{2}\Delta)s^2 + \Delta s(s + \frac{1}{2}s) = s^2.
\end{aligned}
$$

Therefore, $f(S_2) < f(S_1)$, i.e. S^1 is not an optimal solution of the QCOP, which contradicts (2). So (E, I) is a matroid. □

4.4 Asymptotic results

We now present an asymptotic result for the QBCOP. Earlier in this chapter, we observed that a QBCOP can in fact be viewed as a BCOP on an extended ground set. This relationship immediately allows to use Theorem 5 in Chapter 1 to provide asymptotic results for

the QBCOP.

Theorem 36. *Let $I_n = (E^n, F^n, f^n)$ be an instance of a QBCOP such that $S \in F^n$ implies $|S| = \alpha_n$ for $n \in \aleph$. Let $f^n(\{e, f\})$ for all $\{e, f\} \in P[E^n]$ be identically distributed random variables in $[0, M]$ and $f^n(\{e, f\})$ for $\{e, f\} \in P[S]$ be independent for every fixed feasible solution $S \in F^n, n \in \aleph$. Let $\gamma(n)$ be a positive function such that*

$$\ln |S| + \alpha_n^2 \ln \mathbb{P}\left(f^n(\{e, f\}) \leq \frac{M}{1 + \gamma(n)}\right) \text{ diverges to } -\infty \text{ as } n \to \infty. \tag{4.3}$$

Then,

$$\mathbb{P}\left(\frac{g^n(S^w) - g^n(S^*)}{g^n(S^*)} \leq \gamma(n)\right) = 1 - o(1), \tag{4.4}$$

where S^* is an optimal solution, S^w is the worst solution for the instance I_n, and $g^n(S) = \max\{f^n(\{e, f\}) : \{e, f\} \in P[S]\}$. Furthermore, if the series

$$\sum_{i=1}^{\infty} |F^n| \left(\mathbb{P}(f^n(\{e, f\}) \leq \frac{M}{1 + \gamma(n)})\right)^{\alpha_n^2} \tag{4.5}$$

converges then $\frac{g^n(S^w) - g^(S^*)}{g^n(S^*)} \leq \gamma(n)$ holds almost surely.

After discussing the general QBCOP in this chapter, we consider three special cases of the QBCOP - Quadratic Bottleneck Spanning Tree Problem, Quadratic Bottleneck Knapsack Problem and Quadratic Bottleneck Assignment Problem sequentially in Chapter 5, 6 and 7. The general purpose results established in this chapter will be demonstrated by exploiting special problem structures and for each problem, we also develop lower bounding schemes, exact and heuristic algorithms and report experimental results. Note that without loss of generality, we simply consider the non-uniform, asymmetric cost matrix in later chapters and use notation $q(e, f)$ or $q(i, j)$ to denote the quadratic cost.

Chapter 5

The Quadratic Bottleneck Spanning Tree Problem

5.1 Introduction

In this chapter we study a special case of the QBCOP - *quadratic bottleneck spanning tree problem (QBST)*. Given an undirected graph $G = (V, E)$, a family $\mathcal{T}$ of all the spanning trees of G and a cost $q(e, f)$ for $e \in E, f \in E$, the QBST can be formulated as follows:

$$\begin{aligned} &\text{Minimize } \max\{q(e,f) : e, f \in T\} \\ &\text{Subject to} \\ &\qquad T \in \mathcal{T}. \end{aligned}$$

In a QBST, if $q(e, f) = 0$ whenever e and f are adjacent, then the problem is called *the adjacent-only quadratic bottleneck spanning tree problem (AQBST)*.

The QBST feasibility problem, denoted by FQBST, can be stated as follows: "Given a graph G with cost function q for pairs of edges of G and a constant K, does there exist a spanning tree T of G such that $g(T) \leq K$?" where $g(T) = \max\{q(e, f)) : e \in T, f \in T\}$.

5.2 Computational complexity

Although the bottleneck spanning tree problem can be solved in linear time [44], QBST seems more difficult.

Theorem 37. *The FQBST is NP-complete on bipartite graphs even if q takes 0-1 values only.*

Proof. The FQBST is clearly in NP. We now reduce the Hamiltonian path problem (HPP) on a directed graph to the FQBST. The HPP can be stated as follows: "Given a directed graph D, does there exist a Hamiltonian path in D?" From an instance of the HPP, we now construct an instance of the FQBST. Let $V(D) = \{1, 2, \ldots, n\}$ be the node set of D and $E(D)$ be its arc set. From D, let us construct an undirected graph G on $2n$ nodes $\{1, 2, \ldots 2n\}$. For each arc (i, j) of D create an edge $(i, n + j)$ in G. Also introduce edges $(i, n + i)$ for $i = 1, 2, \ldots n$. For $i = 1, 2, \ldots n$, let $A(i) = \{(i, n + j) : (i, n + j) \in E(G), i \neq j\}$ and $A(n + i) = \{(j, n + i) : (j, n + i) \in E(G), i \neq j\}$. Note that the definition of $A(i)$ and $A(n + i)$ excludes the edge $(i, n + i)$ from it for all $i = 1, 2, \ldots, n$. Let q be the cost function for pairs of edges of G where

$$q(e, f) = \begin{cases} 1 & \text{if } e \in A(i), f \in A(i) \text{ for } i = 1, 2, \ldots 2n \\ 0 & \text{otherwise.} \end{cases}$$

Thus $q(e, f) = 0$ if and only if $e \in A(i), f \in A(j), i \neq j$. Choose $K = 0$. Note that edges in G incident on node i (other than the edge $(i, n + i)$) represent arcs going out of node i in D for $1 \leq i \leq n$. Similarly, edges in G incident on node $n + j$ (other than the edge $(j, n + j)$) represent arcs coming into node j in D, for $1 \leq j \leq n$. Suppose that G contains a spanning tree T such that $g(T) \leq 0$. Such a tree can choose at most one edge from $A(i)$ for $i = 1, 2, \ldots, 2n$. It can be verified that this is possible only if T is a Hamiltonian path in G and from this path, a Hamiltonian path of D can easily be recovered. Similarly, if D contains a Hamiltonian path, we can construct a spanning tree T of G from it by splitting each node i of the path using the edge $(i, n + i)$. Thus $g(T) = 0$. The proof follows from the NP-completeness of the Hamiltonian path problem on a directed graph [95]. □

Corollary 38. *For any $\epsilon > 1$ computing an ϵ-optimal solution for the QBST is NP-hard.*

Proof. The proof of this corollary follows from the same construction as in the proof above using cost function

$$q(e,f) = \begin{cases} 1+\epsilon & \text{if } e \in A(i), f \in A(i) \text{ for } i = 1, 2, \dots 2n \\ 1 & \text{otherwise.} \end{cases}$$

It can be verified that an ϵ-optimal solution and an optimal solution to the QBST instance constructed cannot use any edge pair with cost more than 1 when D is Hamiltonian. If D is not Hamiltonian, they must use an edge pair of weight $1+\epsilon$. Thus the QBST instance have an ϵ optimal solution with objective function value 1 if and only if D contains a Hamiltonian path and hence the proof follows. □

In Chapter 1 we have shown the relation between the general QBCOP and the LCOPC by Lemma 3, when specifying the feasible solutions as spanning trees, the relation still holds for the QBST and MSTC.

Corollary 39. *The FQBST is equivalent to the feasibility version of the MSTC.*

Then naturally the AQBST feasibility problem is equivalent to the feasibility version of the MSTAC (FSTAC). From Lemma 6 in Chapter 2, we immediately get the following complexity results.

Corollary 40. *AQBST is NP-complete on a fan, a wheel and a (k,n)-ladder.*

Corollary 41. *QBST is NP-complete on a fan, a wheel and a (k,n)-ladder.*

Corollary 42. *QBST is NP-complete on a ladder.*

5.3 Polynomially solvable special cases and an exact algorithm

Despite the QBST is shown to be NP-hard even on many sparse graphs, we are still able to obtain some polynomially solvable special cases of the QBST by specifying the polynomially solvable QBCOP cases identified in Chapter 4 in the context of QBST. Let $|V| = n$, $|E| = m$ and according to the proof in [202], linear bottleneck spanning tree problem can be solved in $O(m)$ time and thus we have the following results:

Corollary 43. *The QBST with a non-uniform, asymmetric, decomposable/multiplicative cost function can be solved in $O(m^2)$ time.*

Corollary 44. *The QBST with a uniform, asymmetric, decomposable/multiplicative cost function can be solved in $O(m^2)$ time.*

Corollary 45. *The QBST with a non-uniform, symmetric, decomposable/multiplicative cost function can be solved in $O(m)$ time.*

Corollary 46. *The QBST with a uniform, symmetric, decomposable/multiplicative cost function can be solved in $O(m)$ time.*

Corollary 47. *The QBST with a comparable cost function can be solved in $O(m)$ time.*

Note that the algorithm for QBST with a non-uniform, asymmetric, decomposable cost function can be improved to run in $O(m \log n)$ time using dynamic tree data structure along the lines of [91, 205]. The QBST with a uniform asymmetric decomposable cost function reduces to the 2-sum spanning tree problem, which can be solved by solving just one minimum spanning tree problem [118].

If we define

$$\text{BSTP}(e, Q)\text{: Minimize } \max\{q(e, f) \ : \ f \in T\}$$
$$\text{Subject to}$$
$$T \in \mathcal{T},$$

then Theorem 34 in Chapter 4 directly reduces to the following:

Corollary 48. *Given a QBST with cost matrix Q and let r be the critical index of the least-index optimal solution to the BSTP$(1, Q)$, if Q is row graded and $q(1, r) \leq q(2, r) \leq \cdots \leq q(m, r)$, then the QBST can be solved in $O(m)$ time.*

Now let us specialize the general quadratic threshold algorithm proposed in Chapter 4, which is an exact algorithm to solve the QBST. For the rest of this chapter, we assume that without loss of generality, $q(e, f) = q(f, e)$ for $e \in E, f \in E$, and a tree T is represented as its edge set. The crucial step in the quadratic threshold algorithm is solving the FQBST. Given a constant K, let $Q(K) = \{\{e, f\} : q(e, f) > K\}$. In the context of QBST, we can rephrase the FQBST as follows: "Does there exist a spanning tree T of G such that

$\{e, f\} \notin Q(k)$ for $e \in T, f \in T$?" In other words we want a spanning tree T of G that uses at most one edge out of any pair of edges $\{e, f\}$ in $Q(K)$.

Algorithm 5.1: Quadratic Spanning Tree Threshold Algorithm

1: **Input:** A graph G, a cost function q defined on $E \times E$ and an oracle $\alpha(.)$ which with input K and verifies if there exists a spanning tree T of G such that $\{e, f\} \notin Q(K)$ for $e \in T, f \in T$ and output an 'yes' or 'no' answer along with T whenever the answer is 'yes'.
2: Construct an ascending arrangement $z_1 < z_2 < \cdots < z_p$ of distinct costs $\{q(e, f) : e \in E, f \in E\}$.
3: $\ell = 1$; $u = p$;
4: **while** $u - \ell > 0$ **do**
5: $k = \lfloor \frac{(l+u)}{2} \rfloor$;
6: **if** there exists a spanning tree T such that $\{e, f\} \notin Q(k)$ for $e \in T, f \in T$ **then** $\ell = k + 1$; **else** $u = k$;
7: **end while**
8: **Output** T;

5.4 Heuristic algorithms

Even though an exact algorithm has been established for the QBST, the FQBST is still NP-hard in view of Theorems 3 and 37, thus the Quadratic Spanning Tree Threshold Algorithm would not help too much on practical computations. Therefore we now consider modifying it to obtain an efficient heuristic algorithm.

Note that any spanning tree of G can be represented by a 0-1 vector $x = (x_1, x_2, \ldots, x_m)$, called the incidence vector, where m is the number of edges of G and $x_e = 1$ if and only if $e \in T$. We next formulate the FQBST as a MSTC:

$$\text{MSTC: Minimize } \sum_{e \in T} c_e$$

$$\text{Subject to}$$

$$x \in \mathcal{T},$$

$$x_e + x_f \leq 1 \text{ for all } \{e, f\} \in Q(K)$$

$$x_e, x_f \in \{0, 1\}$$

where $c_e = 0$ for all e. Then the MSTC has a feasible solution if and only if the FQBST has an 'yes' answer. We have discussed the MSTC in Chapter 3 and developed various heuristic algorithms based on tabu search, tabu thresholding, etc. Thus, these algorithms can be employed to solve the FQBST within the quadratic threshold algorithm. If we use an exact algorithm to solve the MSTC, the quadratic threshold algorithm computes an optimal solution to the QBST. If a heuristic algorithm is used to solve the MSTC, it provides a heuristic solution.

When an MSTC heuristic is used to solve the FQBST, and the answer is 'yes', we make a correct decision in the quadratic threshold algorithm. But if the answer is 'no', it is not guaranteed that the answer to the FQBST is indeed 'no' and it is possible that the heuristic may have produced a 'suboptimal' solution leading to a 'no' answer. Since we are interested in just a feasible solution, we can try to run the MSTC heuristic again using a randomly generated cost vector c_e for $e \in E(G)$ with the hope that it may take the algorithm out of the current infeasible spanning tree for the MSTC. We call this a *shake* operation. The shake operation can be repeated for a fixed number of iterations and if a 'no' answer emerges consistently, we can conclude with high probability that the answer is indeed 'no', especially when our MSTC heuristic is powerful. The performance of the heuristic algorithm (and the quadratic threshold algorithm) can be improved by computing good lower and upper bounds on the optimal objective function value and thereby reducing the search range.

For any two subsets R and S of E, let $P[R, S] = \{\{e, f\} : e \in R, f \in S, e \neq f\}$. A quick lower bound for QBST can be obtained as follows. Any spanning tree must have at least one edge incident on each node. For any node i, let $N(i)$ be the set of edges incident on i. For any two distinct nodes i and j, $\min\{q(e, f) : \{e, f\} \in P[N(i), N(j)]\}$ is a lower bound for the optimal objective function value of QBST. Thus

$$\max_{i,j \in V(G), i \neq j} \min\{q(e, f) : \{e, f\} \in P[N(i), N(j)]\} \tag{5.1}$$

is a lower bound for the optimal objective function value of QBST. Note that for the family of spanning trees of the graph G the blocker is the family of cuts in G. Let $\mathcal{D}$ be the collection of all cuts in G. Then by weak duality theorem,

$$\max_{\substack{R,S \in \mathcal{D} \\ R \neq S}} \min\{q(e, f) : \{e, f\} \in P[R, S]\} \tag{5.2}$$

where $P[R,S] = \{\{e,f\} : e \in R, f \in S, e \neq f\}$ and R and S are two distinct cuts (represented as edge sets), gives another lower bound for the optimal objective function value of QBST. Let L_1 be the lower bound given in equation (5.1) and L_2 be the lower bound given equation (5.2). Since $[N(i), V(G) \setminus N(i)]$ is a cut, it can be verified that the L_2 is no worse than L_1. However it is easy to compute L_1 and hence we used it in our computational experiments.

The lower bound L_1 could be arbitrarily bad compared to the L_2 bound, in worst case. Consider graph in Figure 5.1 formed by three triangles joined together by two bridges.

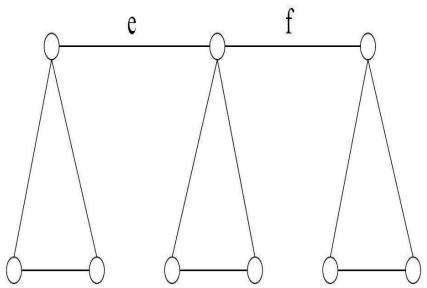

Figure 5.1: QBST: comparison of lower bounds (the number e and f are edge labels)

Set the cost of the edge pair $\{e,f\}$ to w and cost of all other pairs to be zero. Then the L_1 bound is zero whereas L_2 bound is w. Nevertheless, L_1 does not perform so bad in practice, as suggested by our experimental results.

We can speed up our main heuristic algorithm by constructing good upper bounds as well. For this we employ a simple heuristic: first generate a random tree T and $U_1 = g(T)$ is an obvious upper bound. To get a possibly improved upper bound U_2, we use binary search on distinct costs $\{z_1, z_2, \cdots, z_t\}$ of the edge-pairs $\{e,f\}$ such that $q(e,f) \leq U_1$. Initially set $\ell = 1,\ u = t$ and these values are updated appropriately in each iteration. Choose $k = \lfloor \frac{(\ell+u)}{2} \rfloor$ and construct $Q(k)$ as the set of edge pairs $\{e,f\}$ where $q(e,f) > z_k$. Then from the original graph we delete all the edges associated with $Q(k)$ and denote $\tilde{E}$ the set of the remaining edges and $\tilde{G}$ as the resulting graph. If $\tilde{G}$ is connected then z_k is clearly an upper bound and hence we update u as k. If $\tilde{G}$ is disconnected then z_k may or may not

be an upper bound. We proceed as if z_k is not an upper bound update ℓ as $k+1$. This procedure is repeated until $\ell = u$ and we have the upper bound $U_2 = z_l$. This heuristic is summarized in Algorithm 5.2.

Algorithm 5.2: The upper bound algorithm

1: **Input:** A graph G and cost function q for pairs of arcs in G.
2: Randomly generate a spanning tree T, let $U_1 = \max_{e \in T, f \in T}(e, f)$;
3: Construct an ascending arrangement $z_1 < z_2 < \cdots < z_t$ of distinct costs $\{q(e,f) : e \in E, f \in E, q(e,f) \leq U_1\}$;
4: Let $\ell = 1$; $u = t$;
5: **while** $u - \ell > 0$ **do**
6: $\quad k = \lfloor \frac{(l+u)}{2} \rfloor$;
7: $\quad Q(k) = \{e \in E, f \in E, q(e,f) > z_k\}$
8: $\quad \tilde{E} = \{\alpha : \alpha \in E, \alpha \notin \{e,f\}$ for any $\{e,f\} \in Q(k)\}$
9: $\quad \tilde{G}$ be the spanning subgraph of G with edge set $\tilde{E}$;
10: $\quad$ **if** $\tilde{G}$ is connected **then** $u = k$ **else** $\ell = k+1$;
11: **end while**
12: **Output** $U_2 = z_\ell$

Since connectivity of a graph can be tested in linear time and sorting of edge-pair costs can be done in $O(m^2 \log n)$ the upper bound algorithm runs in $O(m^2 \log n)$ time. We now present our main heuristic algorithm (Algorithm 5.3), called *the approximate quadratic spanning tree threshold algorithm (AQST-heuristic)*, to solve QBST.

Note that in step 15 of the algorithm, the MSTC heuristic may or may not find a feasible solution. If it failed to compute a feasible solution, we "heuristically" conclude that no feasible solution exist with the current threshold value. This imprecise step makes our algorithm 5.3, a heuristic. If this step is done by using an exact algorithm for MSTC, then algorithm 5.3 becomes an exact algorithm for QBST.

5.5 Computational results

The AQST-heuristic was coded in C++ and tested on a Dell PC with 3.40 GHz processor and 2.0 GB memory running on Windows XP operation system. We used the public domain compiler Dev C++ for all compilation work. No standard benchmark test problems are available for the QBST. Thus we have generated random test instances as follows: 17 random connected graphs with the number of nodes ranging from 10 to 50 and the number

Algorithm 5.3: AQST-heuristic Algorithm

Input: A graph G and cost function q for pairs of edges in G.
Compute a lower bound L for the optimal objective function value. (Probably using equation (5.1));
Compute an upper bound U for the optimal objective function value. (Probably obtained from the upper bound algorithm);
Construct an ascending arrangement $z_1 < z_2 < \cdots < z_p$ of distinct costs $\{q(e,f) : e \in E, f \in E\}$;
Choose ℓ and u such that $z_\ell = L,\ z_u = U$;
while $u - \ell > 0$ **do**
  $k = \lfloor \frac{(l+u)}{2} \rfloor$; count =0;
  $Q(k) = \{\{e,f\} : e \in E, f \in E, q(e,f) > z_k\}$
  repeat
    if count = 0 **then**
      $d(e) = 0$ for all $e \in E$
    else
      generate random costs $d(e)$ for $e \in E(G)$;
    end if
    Apply a MSTC heuristic on the minimum spanning tree problem with set $Q(k)$of conflict pairs and edge costs $d(e)$. Let T^* be the output tree and $P[T^*]$ be the set of unordered pairs of edges of T^*;
    count = count + 1;
  until $P[T^*] \cap Q(k) = \emptyset$ or count > max-count
  if $P[T^*] \cap Q(k) = \emptyset$ **then** $u = k$; Tree $= T^*$; **else** $\ell = k + 1$;
end while
Output Tree

of edges ranging from 20 to 500 are generated. Note that for instances with 500 edges, there are 124750 variables (pairs of edges), and hence our problems are of reasonably large size. For each graph generated, we have 10 sets of costs, resulting in 10 different problems on the same graph. Costs $q(e,f)$ are random integers in the interval [0, r] where $r = \binom{m}{2}$ and m is the number of edges in the graph. In addition to these instances, we also generated special test problems where the optimal objective function value is known *a priori*. To construct these instances, we again used our 17 graphs generated for the problem set discussed earlier. For each graph G an arbitrary spanning tree T is selected and assigned costs for pairs of its edges $\{e,f\}$ a random integer in the interval [0,n]. All other edge pairs have cost a random integer in $[n+1, r]$. Here n is the number of nodes in G. By construction, T is an optimal solution to the resulting instance of the QBST.

The MSTC heuristic we used in the algorithm is a tabu search heuristic developed in [235]. The number of iterations in the tabu search is set to $\min\{100, \frac{m}{2}\}$ and the length of the tabu list (tabu size) is selected as 7. (We have compared the tabu search algorithm with tabu size 5, 7 and 10. It turns out that the solution is better when the tabu size is larger. With tabu size 7, the quality of the solution obtained was similar to that with tabu size 10, but the former takes less time. Therefore we set the tabu size at 7.) The maximum number of "shake" operations, i.e. max-count, is set to 5, but to save the computational time, if the number of violated conflict pairs in the MSTC returned by the tabu search is larger (say, greater than 10), then we assume it is not likely to have any feasible solutions and hence the rest of the shake operations were skipped.

Table 5.1 and 5.2 shows computational results for the general instances. For each problem, there are $\binom{m}{2}$ cost elements and we use the index rather than actual costs to record the lower bound L. The initial upper bound U is selected as the value returned by the upper bound algorithm and the objective function value 'heu-obj' is selected as that returned by the Quadratic Threshold Heuristic. For example, $L = 87$ means the lower bound is z_{87} in the ordered list of distinct costs. The CPU time was measured in seconds corrected to three decimal digits. Also the number of times a lower bound was updated ('l-update') and the number of times an upper bound was updated ('u-update') are recorded. The column 'shake' shows how many upper bound updates are caused by the shake operations. The 'min', 'max', and 'average' of these values are calculated based on the 10 problems for each graph topology we considered.

To evaluate the quality of the lower bound, the initial upper bound and the heuristic solution, we define the relative bounds as $r_L = \frac{L}{p}\%$, $r_U = \frac{U}{p}\%$, and relative objective function value as $r_{obj} = \frac{heu-obj}{p}\%$, where p is the number of distinct costs. In each binary search iteration of the algorithm, we used tabu search together with shake operations to solve the MSTC on step 16. The ratio $\frac{u-update}{\ell-update}$ provides us some insight into the effectiveness of our MSTC heuristic on step 15. Further, the ratio $\frac{shake}{u-update}$ measures in percentage the effectiveness of shake operations. We calculated these ratios for the test problems by using the average values of L, U, heu-obj, u-update, ℓ-update and shake from Table 5.1 and 5.2, and the results are summarized in Table 5.3. From Tables 5.1, 5.2 and 5.3 it can be observed that, in general, our heuristics run in a reasonable amount of CPU time with acceptable solution quality. The lower bound L tends to get worse as the size of the problems get larger and it is affected significantly by the number of edge pairs. The quality of the initial upper

n	m	L			U			heu-obj			CPU time		
		min	max	av.	min	max	av.	min	max	av.	min	max	av.
10	20	38	87	51.9	109	119	114.2	90	101	97.3	0.110	0.250	0.181
	30	25	40	34	238	261	253.4	173	210	193	0.297	0.844	0.517
	40	25	48	35	448	478	466.2	293	323	308.8	0.547	0.922	0.736
	45	22	68	33.8	597	621	606.1	366	406	391.2	0.625	1.156	0.838
30	60	599	946	711	1070	1129	1099.3	1023	1083	1056.4	3.094	5.859	4.505
	80	523	959	705.4	1918	2000	1966.6	1799	1859	1826.6	6.078	8.672	7.466
	100	525	1684	926.5	3031	3120	3063.8	2774	2843	2805.6	5.422	8.328	7.080
	200	456	660	554.3	12004	12154	12092.6	9883	10285	10096.9	50.782	80.517	65.881
	300	420	1156	626.6	28021	28194	28090.7	23310	23749	23468.8	208.097	1254.110	645.713
	435	227	354	297.5	40232	40375	40276.8	31435	32504	31974.2	902.699	5270.080	2693.204
50	100	1723	2670	2055	3053	3137	3082	2987	3049	3011.5	18.407	37.125	29.600
	150	1940	4419	2519.1	6928	7061	7001.7	6614	6711	6669.8	74.735	121.064	97.867
	200	1859	4002	2646.7	12030	12180	12131.2	11173	11440	11335.4	126.033	185.675	143.282
	250	1490	5431	2430.1	19258	19479	19378.6	17863	18073	17988.2	224.347	385.755	290.841
	300	659	1152	884.6	10954	10993	10975.6	10072	10302	10173.5	306.926	1125.080	646.438
	450	1222	2565	1725.4	43146	43402	43245.6	38772	39298	39104.8	947.325	5614.320	2796.142
	500	1178	2553	1665.1	53552	53785	53699.2	48358	49004	48640.3	1197.140	9551.810	5052.977

Table 5.1: QBST: general problems (part I)

n	m	l-update			u-update			shake		
		min	max	av.	min	max	av.	min	max	av.
10	20	2	4	3.1	2	4	2.80	0	0	0
	30	3	6	4.5	2	5	3.3	0	1	0.3
	40	3	6	4.7	0	3	0.9			
	45	3	6	4.6	3	6	4.5	0	2	0.6
	60	4	6	5.2	1	5	3.2	0	1	0.7
	80	4	6	4.9	4	7	5.6	0	2	0.7
30	100	4	8	6	4	7	5.2	0	4	2.2
	200	5	10	7.1	5	9	6.3	1	8	3.7
	300	5	11	8.3	3	10	6.3	2	6	3.9
	4355	12	8.6	3	11	6.7	1	7	3.9	
50	100	4	9	6.1	1	6	3.8	0	4	1.7
	150	4	10	8.6	2	9	3.4	1	8	2.6
	200	4	9	7.7	4	9	5.4	2	5	3.6
	250	6	12	9	2	8	4.9	0	6	3.1
	300	5	10	7.6	3	9	5.8	2	6	3.6
	450	8	11	9.9	4	8	5.3	2	7	3.4
	500	4	11	8.4	4	12	7.2	3	6	4.3

Table 5.2: QBST: general problems (part II)

bound U appears not very good, and it turns worse as n and m increases. Tabu search heuristic for the MSTC combined with "shake" operations performed consistently and the ratio $\frac{u-update}{\ell-update}$ averaged around 0.8. The scheme worked relatively well for larger problems, since the quality of the heuristic objective function value appears to become better as the number of edge pairs is increased. From $\frac{shake}{u-update}\%$ it can be seen that the shake operation works fairly well and it contributed to more than 60% of all the u-updates for large problems.

The results on the special problems are summarized in Table 5.4. 'opt-gap' represents the gap between the objective function value of the heuristic solution and that of the optimal solution.

It is interesting to note that our heuristic algorithm identified an optimal solution for all special problems generated with known optimal solution. This indicates the power of the tabu search heuristic in solving the MSTC and the power of the "shake" operations in search diversifications. But it is possible that the special problems we constructed may be "too special" that they do not capture the real complexity of the QBST. Nevertheless, the test results on these problems are very impressive.

5.6 Asymptotic results

In the end of this chapter, we adapt the asymptotic result established in Chapter 4 to the QBST.

Consider an instance $ST(n)$ of the QBST defined on the graph G^n with cost function q^n where $q^n(e,f)$s are uniformly distributed random numbers in the interval $[0,1]$. Let T^* be an optimal solution to $ST(n)$ and T^w is a worst solution. Specializing Theorem 36 to QBST, we get the following bound.

Theorem 49. $\frac{g^n(T^w)}{g^n(T^*)} \leq n^{\frac{n}{(n-1)^2}}$ *almost surely. Equivalently,*

$$\mathbb{P}\left(\frac{g^n(T^w)}{g^n(T^*)} > n^{\frac{n}{(n-1)^2}} \text{ infinitely often}\right) = 0.$$

Proof. Let F^n be the family of all spanning trees of G^n on n nodes. By Cayley's theorem [48], $|F^n| \leq n^{n-2}$. Each spanning tree contains $n-1$ arcs. Since $q^n(\{e,f\})$s are uniformly and independently distributed random numbers in the interval $[0,1]$, $\mathbb{P}(q^n(\{e,f\}) \leq \kappa) = \kappa$. Note that

$$\sum_{n=1}^{\infty} |F^n| \left(\mathbb{P}(q^n(\{e,f\}) \leq \frac{1}{n^{\frac{n}{(n-1)^2}}})\right)^{(n-1)^2} \leq \sum_{n=1}^{\infty} n^{n-2}\frac{1}{n^n} \tag{5.3}$$

$$= \sum_{n=1}^{\infty} \frac{1}{n^2} \text{ which coverges to } \frac{\pi}{6}. \tag{5.4}$$

The result now follows from Theorem 36 by choosing $\gamma(n) = n^{\frac{n}{(n-1)^2}} - 1$, $\alpha_n = n-1$ and $M = 1$. □

n	m	$r_L(\%)$	$r_U(\%)$	$r_{obj}(\%)$	$\frac{u-update}{l-update}$	$\frac{shake}{u-update}\%$
10	20	43.3	95.2	81.1	0.9	0.0
10	30	12.5	93.2	71.0	0.7	9.1
10	40	7.2	95.5	63.3	1.1	19.1
10	45	5.4	96.7	62.4	1.0	13.3
30	60	63.8	98.7	94.8	0.6	21.9
30	80	35.5	98.8	91.8	1.1	12.5
30	100	29.8	98.6	90.3	0.9	42.3
30	200	4.5	98.8	82.5	0.9	58.7
30	300	2.2	99.2	82.9	0.8	61.9
30	435	0.7	99.3	78.8	0.8	58.2
50	100	66.1	99.2	96.9	0.6	44.7
50	150	35.7	99.2	94.5	0.4	76.5
50	200	21.6	99.1	92.6	0.7	66.7
50	250	12.5	99.5	92.4	0.5	63.3
50	300	8.0	99.4	92.1	0.8	62.1
50	450	4.0	99.5	90.0	0.5	64.2
50	500	3.1	99.5	90.2	0.9	59.7

Table 5.3: QBST: relative bounds and obj of general problems

n	m	opt-gap (%)	CPU time		
			min	max	av.
10	20	0	0.031	0.156	0.067
10	30	0	0.015	0.172	0.040
10	40	0	0.031	0.203	0.072
10	45	0	0.046	0.250	0.102
30	60	0	0.078	0.125	0.102
30	80	0	0.188	0.797	0.295
30	100	0	0.391	1.016	0.478
30	200	0	5.234	6.328	5.634
30	300	0	25.233	28.311	26.835
30	435	0	109.813	119.388	112.391
50	100	0	0.390	0.437	0.409
50	150	0	1.719	2.078	1.825
50	200	0	5.125	6.016	5.375
50	250	0	12.235	13.453	12.505
50	300	0	25.328	31.891	27.389
50	450	0	125.892	130.220	127.395
50	500	0	191.940	214.393	198.801

Table 5.4: QBST: special problems

Chapter 6

The Quadratic Bottleneck Knapsack Problems

6.1 Introduction

Let $E = \{1, 2, \cdots, n\}$ be a finite set and $q(i, j)$ be a cost prescribed for each $(i, j) \in E \times E$. In Chapter 1 we have introduced the *unconstrained 0-1 quadratic programming problem (UQP)*, which is generally represented as the following 0-1 quadratic programming problem

$$\text{Minimize } \sum_{i=1}^{n}\sum_{j=1}^{n} q(i,j)x_i x_j$$

Subject to

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

For any $X = (x_1, \cdots, x_n) \in \{0, 1\}^n$, let $S(X) = \{j : x_j = 1\}$, then the bottleneck version of UQP, called the *bottleneck unconstrained 0-1 quadratic programming problem (BUQP)*, can be stated as

$$\text{Minimize } \max\{q(i,j) : (i,j) \in S(X) \times S(X), S(X) \neq \emptyset\}$$

Subject to

$$X \in \{0, 1\}^n.$$

As illustrated in Chapter 1, the UQP is NP-hard and extensive literature is available on the problem. However, the BUQP is not well studied in literature. In fact, it is not difficult to see that the BUQP is solvable in polynomial time. Choose $(i, j) \in E \times E$ such that $max\{q(i, j), q(j, i), q(i, i), q(j, j)\}$ is minimized. Let (r, s) be a pair for which this minimum is attained. Then

$$x_j^* = \begin{cases} 1 & \text{if } j = r \text{ or } j = s \\ 0 & \text{otherwise} \end{cases}$$

is an optimal solution to BUQP. It may be noted that r could be equal to s.

Interestingly, by introducing an additional linear constraint to BUQP, the complexity status is completely changed. We call the resulting problem *the quadratic bottleneck knapsack problem (QBKP)*. Let $w_j \geq 0$ be a prescribed weight of element $j \in E$ and $c > 0$ be a given capacity. Then QBKP can be formally defined as

$$\begin{aligned} &\text{Minimize } \max\{q(i, j) : (i, j) \in S(X) \times S(X)\} \\ &\text{Subject to} \\ &\qquad \sum_{j=1}^{n} w_j x_j \geq c \\ &\qquad x_j = 0 \text{ or } 1 \text{ for } j = 1, \cdots, n, \end{aligned}$$

where $X = (x_1, \cdots, x_n)$. Also for QBKP we assume $q(i, j) \geq 0$ without loss of generality. The QBKP is another special case of the QBCOP discussed in Chapter 4, by letting $\mathcal{F} = \{I : I \subset E, \sum_{j \in I} w_j \geq c\}$

6.2 Computational complexity

The simplicity in the solution approach for the BUQP does not translate into QBKP, although the bottleneck knapsack problem (BKP) and BUQP admits linear time solution procedures.

Theorem 50. *The QBKP is NP-hard even if all w_j's are 1 and $q(i, j)$'s are 0 or 1 only. Further, unless P=NP, there does not exist a polynomial time ϵ-approximation algorithm for QBKP for any $\epsilon \geq 0$.*

Proof. Let us first establish the NP-hardness when $q(i, j)$ takes values 0 or 1 only. We

reduce the maximum weight independent set problem to QBKP. Let $G = (V, E)$ be a graph on which the maximum independent set problem is defined and $V = \{1, 2, \ldots, n\}$. Construct an instance of QBKP as follows: choose $w_j = 1$ for $j = 1, \ldots, n$ and

$$q(i,j) = \begin{cases} 0 & \text{if } i \text{ is not adjacent to } j \text{ in } G \text{ or } i = j \\ 1 & \text{otherwise.} \end{cases}$$

Then the resulting QBKP has optimal objective function value 0 if and only if G has an independent set of size greater than or equal to c. Since the maximum independent set problem is NP-hard, QBKP is also NP-hard. To establish the non-approximability result, we again use a reduction from the maximum independent set problem. Define

$$q(i,j) = \begin{cases} 1 & \text{if } i \text{ is not adjacent to } j \text{ in } G \text{ or } i = j \\ 1 + 2\epsilon & \text{otherwise} \end{cases}$$

and choose $w_j = 1$ for $j = 1, \ldots, n$. Suppose that there is a polynomial time approximation algorithm α for QBKP which produces an ϵ-optimal solution X^* and let X^0 be an optimal solution for the constructed instance of QBKP. If there exists an independent set in G of size greater than or equal to c, then $f(X^0) = \max\{q(i,j) : (i,j) \in S(X^0) \times S(X^0)\} = 1$. Let $f(X^*) = \max\{q(i,j) : (i,j) \in S(X^*) \times S(X^*)\}$. Also we know that $\frac{f(X^*)}{f(X^0)} \leq 1 + \epsilon$. This happens happens precisely when $f(X^*) = 1$ and hence G has an independent set of size c. If $f(X^0) = 1 + 2\epsilon$ then G has no independent of size of at least c and in this case $f(X^*)$ must also be $1 + 2\epsilon$. Thus whenever α is polynomial for any $\epsilon > 0$ we can solve the maximum independent set problem on G in polynomial time. Thus unless P=NP, there cannot be a polynomial time ϵ-approximation algorithm for QBKP. □

6.3 Polynomially solvable special cases and an exact algorithm

In Chapter 4 we discussed various nontrivial QBCOP solvable cases where the polynomial solvability was linked to that of an associated optimization problem. We now specialize such oracle based algorithm to the QBKP. Note that here the dimension of the cost matrix Q is $n \times n$ and the complexity of solving the bottleneck knapsack problem is $O(n)$ by using the

binary serach version of the threshold algorithm along with a reduction strategy.

Corollary 51. *The QBKP with a non-uniform, asymmetric, decomposable/multiplicative cost function can be solved in $O(n^2)$ time.*

Corollary 52. *The QBKP with a uniform, asymmetric, decomposable/multiplicative cost function can be solved in $O(n^2)$ time.*

Corollary 53. *The QBKP with a non-uniform, symmetric, decomposable/multiplicative cost function can be solved in $O(n)$ time.*

Corollary 54. *The QBKP with a uniform, symmetric, decomposable/multiplicative cost function can be solved in $O(n^2)$ time.*

Corollary 55. *The QBKP with a comparable cost function can be solved in $O(n)$ time.*

Corollary 56. *Given a QBKP with cost matrix Q and let r be the critical index of the least-index optimal solution to the BSTP$(1, Q)$, which is defined as*

$$\begin{aligned} BKP(i,Q)\text{: } &\textit{Minimize } \max\{q(i,j) \ : \ j \in S(X)\} \\ &\textit{Subject to} \\ &\sum_{j=1}^{n} w_j x_j \geq c \\ &x_j = 0 \textit{ or } 1 \textit{ for } j = 1, \ldots, n. \end{aligned}$$

then if Q is row graded and $q(1,r) \leq q(2,r) \leq \cdots \leq q(m,r)$, then the QBKP can be solved in $O(n)$ time.

Hereafter we assume without loss of generality that Q is symmetric and we store only the elements of Q on or above the diagonal to represent Q.

Similar to the FQBST in Chapter 5, we define *the feasibility version of the QBKP (FQBKP)* as "given a constant K, does there exist a subset I of E such that $\sum_{j \in I} w_j \geq c$ and $\max\{q(i,j) : i,j \in I\} \leq K$"? Interestingly, the FQBKP can be formulated as a *maximum weight independent set problem (MWIP)*. Consider the following graph $\bar{G} = (\bar{V}, \bar{E})$ with node set $\bar{V} = \{i : i \in \{1, 2, \cdots, n\},\ q(i,i) \leq K\}$ and edge set $\bar{E} = \{(i,j) : i,j \in \bar{V},\ q(i,j) > K\}$. Let w_j be the weight of node $j \in \bar{V}$. Then we formulate the maximum weight independent set problem as follows.

$$\text{MWIP: Maximize } \sum_{j \in \bar{V}} w_j x_j$$

Subject to

$$x_i + x_j \leq 1 \quad \forall\ (i,j) \in \bar{E}$$

$$x_j = 0 \text{ or } 1 \text{ for } j \in \bar{V}.$$

Let $X^0 = (x_2^0, x_2^0, \cdots, x_n^0)$ be an optimal solution to MWIP with optimal objective function value w^0. Define $X^* = (x_1^*, x_2^*, \cdots, x_n^*)$ as

$$x_j^* = \begin{cases} x_j^0 & \text{if } j \in \bar{V} \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Note that $\sum_{j \in \bar{V}} w_j x_j^0 = \sum_{j=1}^{n} w_j x_j^* = w^0$.

Theorem 57. *There exists a feasible solution to QBKP with objective function value no more than K if and only if $w^0 \geq c$. Further, X^* is such a feasible solution if $w^0 \geq c$.*

Proof. If $w^0 \geq c$, then X^* is a feasible solution to the QBKP. For $(i,j) \in \bar{E}$, either $x_i^* = 0$ or $x_j^* = 0$. Thus $x_i^* x_j^* = 0$ for $\forall (i,j) \in \bar{E}$ and hence the QBKP objective function value of X^* is no more than K. If $w^0 < c$, then X^* is not a feasible solution to QBKP. Suppose there exists a solution $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n)$ of QBKP with objective function value no more than K. Then for any $(i,j) \in \bar{E}$, either $\tilde{x}_i = 0$ or $\tilde{x}_j = 0$. Thus $\tilde{X}$ determines an independent set of G and feasibility of $\tilde{X}$ for QBKP implies $\sum_{j=1}^{n} w_j \tilde{x}_j \geq c$. Thus $\tilde{X}$ is a better solution to MWIP, contradicting the optimality of X^0. This completes the proof. □

The Quadratic Knapsack Threshold Algorithm is then obtained by adapting the general algorithm in Chapter 4, and solving a MWIP in each iteration. Algorithm 6.1 gives the detailed description.

Theorem 58. *The quadratic knapsack threshold algorithm solves the QBKP in $O(\phi(n) \log n)$ time, where $\phi(n)$ is the complexity of the maximum weight independent set problem.*

Let L be a lower bound on the optimal objective function value of QBKP. Consider the

Algorithm 6.1: The Quadratic Knapsack Threshold Algorithm

Input: A QBKP with cost function q, capacity c and weight w_j for each $j \in E$;
Construct an ascending arrangement $z_1 < z_2 < \cdots < z_p$ of distinct $q(i,j)$'s;
$\ell = 1$; $u = p$;
while $u - \ell > 0$ **do**
  $k = \lfloor \frac{(l+u)}{2} \rfloor$;
  Solve the MWIP on $G = (\bar{V}, \bar{E})$, where $\bar{V} = \{i : i \in \{1, \cdots, n\},\ q(i,i) \leq z_k\}$, $\bar{E} = \{(i,j) : i,j \in \bar{V},\ q(i,j) > z_k\}$, let X^0 be the solution to MWIP with objective function value w^0 and X^* is the solution obtained from X^0 as given by equation (6.1).
  if $w^0 \geq c$ **then**
    $u = k$;
  else
    $\ell = k + 1$;
  end if
end while
Output: z_l as the optimal objective function value and the corresponding optimal solution.

graph $\hat{G} = (\hat{V}, \hat{E})$ where $\hat{V} = \{1, 2, \cdots, n\}$ and $(i,j) \in \hat{E}$ if and only if $q(i,j) \geq L$. We call $\hat{G}$ the *L-support graph* of Q.

Based on Theorem 57, if the maximum weight independent set problem on all subgraphs of the L-support graph of Q can be solved in polynomial time and L can be identified in polynomial time, then QBKP can be solved in polynomial time. Then since the MWIP on perfect graph is polynomially solvable [61, 114, 115, 223], we immediately obtain the following result:

Corollary 59. *If the L-support graph of a given QBKP is a perfect graph, then the QBKP can be solved in polynomial time.*

Note that series-parallel graph and bipartite graph are both special cases of perfect graph, therefore the QBKP is polynomially solvable on them.

6.4 Heuristic algorithms

Our first heuristic algorithm is a variation of the exact algorithm (the quadratic knapsack threshold algorithm) discussed in Section 3. Here, instead of solving the maximum weight independent set problem optimally, we solve it using a heuristic algorithm and the heuristic

solution produced by this algorithm is used to determine the binary search direction. The resulting algorithm is called *approximate quadratic knapsack threshold heuristic* or AQKT-heuristic. The quality and running time of this heuristic depends primarily on the quality and running time of the heuristic for the maximum weight independent set problem we use. In the section on computational results, we discuss this with additional details.

6.4.1 Semi-greedy Algorithm

Let us now consider another heuristic using the semi-greedy strategy discussed in [130, 212]. Semi-greedy algorithms are originally developed to solve vehicle routing problems. Such algorithms are also used in various implementations of GRASP algorithms [80]. However, to the best of our knowledge, semi-greedy algorithms are not tested for any quadratic bottleneck problems. We are interested in semi-greedy algorithms because these are simple heuristics that provide reasonable solutions very quickly and the inherent randomization allows diversification in the construction process. Further, standard improvement-type algorithms seem not very efficient for quadratic bottleneck problems, as discussed later in this section. The algorithm builds a solution X by selecting elements of $S(X)$ one at a time. The algorithm maintains a subset $S(X)$ of $\{1, 2, \ldots, n\}$ in each iteration. Let $\bar{S} = \{1, 2, \ldots, n\} - S(X)$. For each $j \in \bar{S}$ let $p_j = \max\{q(i, j) : i \in S(X)\}$. Given a parameter γ, generate a random integer r from $[1, \gamma]$ and choose a subset I of $\bar{S}$ such that $j \in I$ implies p_j is no larger than the r^{th} smallest element of the set $\{p_j : j \in \bar{S}\}$ for $r \leq \gamma$ (counting multiplicity). The algorithm then selects a j from I randomly and update $S(X)$ as $S(X) \cup \{j\}$. The process is continued until a feasible solution $S(X)$ is reached. This whole process is further repeated for a fixed number of times and we output the overall best solution obtained. A formal description of the algorithm is given in Algorithm 2.

When $\gamma = 1$ the above algorithm reduces to a greedy algorithm. The objective function value of the solution obtained by the semi-greedy algorithm can be used as an upper bound to speed up the quadratic threshold algorithm and the AQKT-heuristic.

6.4.2 Improving a solution

Let X^* be any solution to the QBKP and z^* be its objective function value. Clearly X^* is a feasible solution to the integer program

Algorithm 6.2: The Semi-Greedy Algorithm

Input: A QBKP with cost function q, capacity c and weight w_j for each $j \in E$;
γ_iter=0;
Let $z^* = \infty,\ S^* = \emptyset$;
while γ_iter < max_γ_iter **do**
  Choose a value of γ.
  iter=0;
  while iter<max-iter **do**
    Randomly choose $i^0 \in E$, let $S(X) = \{i^0\}$, $\bar{S} = \{1, \cdots, n\} \setminus S(X)$;
    while $\sum_{i \in S(X)} w_i < c$ **do**
      For each $j \in \bar{S}$ let $p_j = \max\{q(i,j) : i \in S(X)\}$;
      Generate a random integer $r \in [1, \min\{\gamma, |\bar{S}|\}]$;
      Let p_{j^*} be the r-th smallest element in $\{p_j : j \in \bar{S}\}$;
      $S(X) = S(X) \cup \{j^*\}$, $\bar{S} = \bar{S} \setminus \{j^*\}$;
    end while
    Let $z^0 = \max\{q(i,j) : (i,j) \in S(X) \times S(X)\}$;
    if $z^0 < z^*$ **then**
      $z^* = z^0$;
      $S^* = S(X)$;
    end if
    iter=iter+1;
  end while
  γ_iter=γ_iter+1;
end while
Output S^* and z^*;

IP: Maximize $0X$

Subject to

$$\sum_{j=1}^{n} w_j x_j \geq c$$

$$x_i + x_j \leq 1 \text{ if } q(i,j) > z^*$$

$$x_j = 0 \text{ or } 1.$$

Let $z_1 < z_2 < \cdots < z_p$ be an ascending arrangement of distinct $q(i,j)$ values and let $z^* = z_r$ for some $r \leq p$. Let $E_t = \{(i,j) : q(i,j) = z_t\}$. Then X^* can be improved if and only if the

problem IP with the additional constraints

$$x_i + x_j \leq 1 \text{ for } (i,j) \in E_{r-1} \tag{6.2}$$

is feasible. Obviously, finding an improved solution is NP-hard. Since the modified IP is obtained from the original IP by adding the cutting planes described by (6.2), the resulting improvement scheme is called a cutting plane algorithm. If the modified IP is solved by an exact algorithm, the resulting cutting plane algorithm is the same as using a top-down sequential search in the quadratic threshold algorithm. If the modified IP is solved using a heuristic, the cutting plane algorithm can be viewed as the top-down sequential search version of the AQKT-heuristic. In practice, it may be better to explore possible improvements by more than one step (say k steps, for small values of k). In this case we are adding cutting planes

$$x_i + x_j \leq 1 \text{ for } (i,j) \in E_{k,r} = \cup_{t=1}^{k} E_{r-t}.$$

As soon as the modified IP becomes infeasible, the algorithm is terminated. The modified IP can be solved in many ways. For example by solving a MWIP [95] or a knapsack problem with conflict pairs [196]. This cutting plane approach could work better if we have a good quality solution to start with. A formal description of the cutting plane algorithm is given in Algorithm 3.

6.5 Computational results

The lower bound algorithm, heuristics and the exact algorithm were coded in C++ and tested on a Linux workstation with Intel Xeon E5410 CPU (2.33GHz) and 8GB RAM running Red Hat Enterprise Linux (kernel 2.6.18). There are no standard benchmark problems available for QBKP. Thus we have generated the test instances randomly. The test instances are generated using two parameters n and Δ, where n is the number of variables and Δ is the percentage of the number of $q(i,j)$'s values that are non-zero. When Δ is small, the matrix Q is sparse. We selected $n = 100, 200, 300, \cdots, 700$ and $\Delta = 50, 75$ and 100 for our experiments. Note that for $n = 700$ and $\Delta = 100$, there are 245350 $q(i,j)$ values and hence the problems considered are of reasonably large size. For each problem size (n, Δ), 5 different problems ($q(i,j)$ values) were generated and solved using the quadratic threshold algorithm where the MWIP is solved using CPLEX. Some of these problems were solved

Algorithm 6.3: The Cutting Plane Algorithm

1: **Input:** A QBKP with cost function q, capacity c and weight w_j for each $j \in E$;
2: Solve the QBKP using the semi-greedy algorithm, let X^* be the solution produced and z^* be the optimal objective function value.
3: Construct an ascending arrangement $z_1 < z_2 < \cdots < z_p$ of distinct $q(i,j)$ such that $q(i,j) \leq z^*$;
4: Let $\delta = p$;
5: Consider the IP

$$\begin{aligned} &\text{Maximize } 0X \\ &\text{Subject to} \\ &\quad \sum_{j=1}^{n} w_j x_j \geq c \\ &\quad x_i + x_j \leq 1 \text{ if } q(i,j) > z^* \\ &\quad x_j = 0 \text{ or } 1. \end{aligned}$$

Let X^* be an optimal solution to the IP.
6: Choose the value of the parameter k;
7: **repeat**
8: Introduce the cutting planes

$$x_i + x_j \leq 1 \quad \text{for} \quad (i,j) \in E_{k,\delta} = \cup_{t=1}^{k} E_{\delta - t}.$$

Designate IP as IP together with the cutting planes and re-solve the IP with additional constraints. Let X^0 be the resulting solution if IP is feasible.
9: **if** IP is feasible **then**
10: $X^* = X^0$;
11: $\delta = \delta - k$;
12: **end if**;
13: **until** IP is infeasible or $\delta < k$;
14: Output X^*;

quickly and produced optimal solutions whereas some took more time while the remaining were not solved even after running for a couple of days. From each problem size (n, Δ) we have selected one of the 'hard' problems (problems that took more time to solve by the quadratic threshold algorithm and problems that could not be solved) to include in our test bed and named the problems using the convention "QBKPxxx-y", where xxx represents n and y=1, 2, 3 respectively represent $\Delta = 50,\ 75,\ 100$.

For semi-greedy heuristics, all experiments are conducted using the following values of the parameters: max_γ_iter=3,γ=1, 3, 5 and max_iter=5.

Let us now discuss the implementations of the quadratic knapsack threshold algorithm

(QKT_algorithm) and the AQKT_heuristic. Let $z_1 < z_2 < \cdots < z_p$ be the distinct values of $q(i,j)$. Let U be the objective function value of the semi-greedy heuristic and L be a lower bound as developed in Chapter 4, Lemma 19. The values of l and u are selected such that $z_l = L$ and $z_u = U$. Depending on the way the MWIP is solved, we have tested two variations of the AQKT_heuristic, called "AQKT_KPC" and "AQKT_UQP".

In AQKT_KPC, we solve the MWIP using CPLEX 12.1.0 with an added inequality $\sum_{j=1}^{n} w_j x_j \geq c$. Thus we essentially solved *the knapsack problem with conflict pairs (KPC)*

$$\begin{aligned} \text{KPC: Maximize } & \sum_{j=1}^{n} w_j x_j \\ \text{Subject to} & \\ & \sum_{j=1}^{n} w_j x_j \geq c \\ & x_i + x_j \leq 1 \text{ if } q(i,j) > z_k \\ & x_j = 0 \text{ or } 1. \end{aligned}$$

We used a time limit of 60 seconds for each call of CPLEX and if no feasible solution is obtained, we heuristically considered the KPC to be infeasible. CPLEX is terminated as soon as a feasible solution is identified.

In AQP_UQP variation, we formulated the MWIP as an UQP with costs d_{ij}, where

$$d_{ij} = \begin{cases} w_j & \text{if } i = j \\ -M & \text{if } i \neq j \text{ and } q(i,j) > z_k \\ 0 & \text{if } i \neq j \text{ and } q(i,j) \leq z_k, \end{cases}$$

where M is a large number. If the resulting UQP has a solution with objective function value no less than c, then the MWIP has a feasible solution with weight greater than or equal to c. We solved the UQP using a heuristic algorithm reported in [159] which is found to be very effective in solving several related optimization problems [111, 228].

In the cutting plane algorithm, we again used CPLEX to solve the IP with a cpu time limit of 60 seconds. To set a proper value for the parameter k, we have tested k =1, 3, 5, 10, 20 and selected $k = 10$ in the final experiments.

Problem Name	n	p		opt	lb		Greedy		Semi-Greedy	
		value	cpu	position	position	cpu	position	cpu	position	cpu
QBKP100-1	100	2211	0.02	198	0	0	1195	0	825	0
QBKP100-2	100	3082	0	1653	0	0.01	2583	0	2036	0
QBKP100-3	100	3851	0	1866	240	0	2924	0	2806	0
QBKP200-1	200	8579	0.2	5749	0	0.04	6895	0	6788	0
QBKP200-2	200	12127	0.05	7883	0	0.03	10715	0	9867	0
QBKP200-3	200	14960	0	11591	1631	0.03	13237	0	12965	0
QBKP300-1	300	16313	0.98	14540	0	0.13	15345	0	15344	0.01
QBKP300-2	300	21055	0.26	14140	0	0.09	17818	0	17087	0.01
QBKP300-3	300	24434	0.01	21949	4040	0.08	23416	0	23037	0.01
QBKP400-1	400	23158	0.11	11451	0	0.31	17640	0	15897	0.01
QBKP400-2	400	27465	0.8	-	0	0.22	25522	0	25152	0.02
QBKP400-3	400	29872	0.02	-	2819	0.17	27683	0.01	27525	0.02
QBKP500-1	500	27956	7.55	-	0	0.59	26560	0	26319	0.03
QBKP500-2	500	30903	1.99	-	0	0.41	22965	0	20949	0.01
QBKP500-3	500	32004	0.03	-	267	0.32	16858	0	14750	0.01
QBKP600-1	600	30688	15.62	-	0	1.02	28987	0	28965	0.04
QBKP600-2	600	32233	3.93	-	0	0.68	31079	0.01	31078	0.05
QBKP600-3	600	32631	0.06	-	5289	0.55	31776	0	31594	0.04
QBKP700-1	700	32011	29.96	-	0	1.61	29943	0	29643	0.05
QBKP700-2	700	32654	7.42	-	0	1.09	31178	0.01	31033	0.05
QBKP700-3	700	32754	0.08	-	3856	0.85	31578	0	31511	0.04

Table 6.1: Experimental results: I

Table 6.1 and 6.2 summarize our main experimental results. Here n is the number of variables and p is the number of distinct costs. The remaining columns respectively record information about the lower bound, the optimal objective function value, and test results for the 5 heuristics considered. Note that since our problem has a bottleneck objective function, the exact values of the $q(i, j)$'s are not important but their orders are. Therefore, we reported the positions instead of the exact values. For example, "0" means the objective function value returned equals the smallest $q(i, j)$. The column "cpu" under AQKT_KPC is the time for completing the "while loop" in the AQKT_heuristic, so the total cpu time for AQKT_KPC should actually be reported as this time plus the cpu time of calculating p, lb and running time for the semi-greedy heuristic. But these extra time components are negligible. For AQKT_UQP and the cutting plane algorithm, similar information is reported.

For the problems with $n \geq 400$, it took days to run the exact algorithm and hence we aborted these experiments. From table 1 we can see that the lower bound algorithm takes very little time but does not consistently provide high quality bounds. The Greedy and

Problem Name	n	AQKT_KPC		AQKT_UQP		cutting plane	
		position	cpu	position	cpu	position	cpu
QBKP100-1	100	**198**	6.69	**198**	306.94	205	7.02
QBKP100-2	100	**1653**	6.48	**1653**	493.18	1656	5.41
QBKP100-3	100	**1866**	7.09	**1866**	184.05	**1866**	7.1
QBKP200-1	200	**5749**	1034.07	5782	507.49	5758	287.72
QBKP200-2	200	**7883**	974.81	**7883**	507.71	7887	339.75
QBKP200-3	200	**11591**	1582.95	11593	455.71	11595	468.58
QBKP300-1	300	14564	2450.36	14586	600.49	14584	842.52
QBKP300-2	300	14148	2914.92	14143	516.73	14167	1387.79
QBKP300-3	300	**21949**	2199.25	22083	602.52	21987	1022.62
QBKP400-1	400	11885	2015.23	11614	512.83	15397	3248.35
QBKP400-2	400	22514	2412.11	22352	673.25	22412	1858.6
QBKP400-3	400	25312	1921.81	25122	497.65	25285	1579.05
QBKP500-1	500	24881	1910.58	24824	527.7	24729	2324.64
QBKP500-2	500	17591	1006.99	12311	663.28	20759	3084.39
QBKP500-3	500	13813	1397.63	8177	534.08	14705	600.02
QBKP600-1	600	27440	2272.2	27212	586.01	27365	753.47
QBKP600-2	600	29930	2968.35	29873	656.28	29938	2530.15
QBKP600-3	600	30932	2549.76	30766	776.06	31004	1725.54
QBKP700-1	700	27748	2840.4	27416	584.8	28083	3043.85
QBKP700-2	700	30314	3310.29	29657	564.9	30173	3012.9
QBKP700-3	700	30631	2840.77	30272	581.24	30981	3229.07

Table 6.2: Experimental results: II (numbers in bold represent the cases where the heuristics produced an optimal solution)

Semi-greedy algorithms can obtain a heuristic solution quite fast but the quality of these solutions is much worse than the quality provided by the other three heuristics. AQKT_UQP outperforms other heuristics by returning solutions with better objective function values and less cpu time. AQKT_KPC can sometimes provided better solutions than the cutting plane algorithm but the latter is faster in 15 out of 21 problems.

6.6 Variations of the QBKP

Let us now consider some variations of the QBKP. Our first variation is *the quadratic bottleneck 0-1 programming problem (QB0-1)*, which can be formulated as follows

QB0-1: Minimize $\max\{q(i,j) : (i,j) \in S(X) \times S(X)\}$

Subject to

$$AX \geq b$$

$$X \in \{0,1\}^n,$$

where A is a $m \times n$ matrix and $b \in \mathcal{R}^m$. Note that QBKP is a special case of QB0-1 when $m = 1$. The quadratic threshold algorithm can be modified to solve QB0-1, where instead of solving a MWIP in each iteration, we try to find if there exists a solution $X = (x_1, x_2, \cdots, x_n)$ such that $AX \geq b$ and $x_i + x_j \leq 1$ for $(i,j) \in \bar{E}$. Also, we can have heuristics similar to the AQKT variations.

The maxmin version of QBKP can be defined as

QBKP1: Maximize $\min\{q(i,j) : (i,j) \in S(X) \times S(X)\}$

Subject to

$$\sum_{j=1}^{n} w_j x_j \geq c$$

$$x_j = 0 \text{ or } 1.$$

The problems QBKP and QBKP1 are equivalent. To see this, consider an instance of QBKP1 with costs $q(i,j)$ and weights w_j. Let $F_{\geq} = \{X = (x_1, \cdots, x_n) \in \{0,1\}^n : \sum_{j=1}^{n} w_j x_j \geq c\}$. Now

$$\max_{X \in F_{\geq}} \min\{q(i,j) : (i,j) \in S(X) \times S(X)\} = - \min_{X \in F_{\geq}} \max\{-q(i,j) : (i,j) \in S(X) \times S(X)\}. \tag{6.3}$$

Let M be a large number. Since $-q(i,j)$ and $M - q(i,j)$ have the same ordering, an optimal solution of $\min_{X \in F_{\geq}} \max\{-q(i,j) : (i,j) \in S(X) \times S(X)\}$ and $\min_{X \in F_{\geq}} \max\{(M - q(i,j)) : (i,j) \in S(X) \times S(X)\}$ are the same. Thus, the algorithms derived in the previous sections can be used to solve QBKP1. However, the reduction from QBKP1 to QBKP discussed above does not preserve performance ratios of approximation algorithms. Theorem 50 extends to the

QBKP1 as summarized below.

Theorem 60. *The QBKP1 is NP-hard even if all w_j's are 1 and $q(i,j)$'s are 0 or 1 only. Further, unless P=NP, there does not exist a polynomial time ϵ-approximation algorithm for QBKP1 for any $\epsilon \geq 0$.*

The proof of this theorem can be constructed following the proof of Theorem 50 and hence is omitted. By reversing the direction of the constraint of the QBKP and QBKP1 we get two more variations of the QBKP which are stated below.

$$\text{QBKP2: Minimize } \max\{q(i,j) : (i,j) \in S(X) \times S(X)\}$$
$$\text{Subject to}$$
$$\sum_{j=1}^{n} w_j x_j \leq c$$
$$x_j = 0 \text{ or } 1.$$

$$\text{QBKP3: Maximize } \min\{q(i,j) : (i,j) \in S(X) \times S(X)\}$$
$$\text{Subject to}$$
$$\sum_{j=1}^{n} w_j x_j \leq c$$
$$x_j = 0 \text{ or } 1.$$

When $q(i,j) \geq 0$, the QBKP2 is trivial since $S(X) = \emptyset$ gives an optimal solution. However, its "max-sum" counterpart is NP-hard and studied by various authors [24, 51, 92, 127, 133, 182, 209]. Let us now consider the QBKP3. Clearly there exists an optimal solution to the QBKP3 where at most two x_j's are one and all other x_j's have value zero. Such an optimal solution can be identified in $O(n^2)$ time by a simple sequential search algorithm as discussed in Algorithm 6.4.

Algorithm 6.4: QBKP3

Input: A QBKP3 with cost function q, capacity c and weight w_j for each $j \in E$;
Let $S^* = \emptyset,\ \bar{S} = E \times E,$ and $z^* = -\infty$;
while $\bar{S} \neq \emptyset$ **do**
  Take $(r, t) \in \bar{S}$;
  if $r = t,\ q(r, r) > z^*$ and $w_r \leq c$ **then**
    $S^* = \{r\}$;
    $z^* = q(r, r)$;
  end if
  if $r \neq t,\ \min\{q(r, r), q(t, t), q(r, t), q(t, r)\} > z^*$ and $w_r + w_t \leq c$ **then**
    $S^* = \{r, t\}$;
    $z^* = \min\{q(r, r), q(t, t), q(r, t), q(t, r)\}$;
  end if
  $\bar{S} = \bar{S} \setminus \{(r, t) \cup (t, r)\}$;
end while
Output: S^* and z^*.

Chapter 7

The Quadratic Bottleneck Assignment Problem

7.1 Introduction

In this chapter we study *quadratic bottleneck assignment problem (QBAP)*. Let $G = (V_1, V_2, E)$ be a bipartite graph with $|V_1| = |V_2| = n$. For each $e, f \in E$, a real valued cost $q(e, f)$ is prescribed. Then *the quadratic bottleneck assignment problem (QBAP)* can be formulated as follows:

$$\begin{aligned} &\text{Minimize } \max_{\substack{e\in M\\ f\in M}} q(e,f) \\ &\text{Subject to} \\ &\qquad M \in \mathcal{M}, \end{aligned}$$

where $\mathcal{M}$ is the family of all perfect matchings in G. Without loss of generality, we can assume that G is a complete bipartite graph. Let $N = \{1, 2, \cdots, n\}$ and $\mathcal{P}_n$ be the family of all permutations of N, take any matching $M \in \mathcal{M}$, there is a unique permutation $\pi \in \mathcal{P}_n$ such that $M = \{(i, \pi(i)) : i \in N\}$. Therefore the QBAP has an alternative formulation:

$$\text{Minimize } \max_{i,j \in N} q_{i\pi(i)j\pi(j)}$$

Subject to

$$\pi \in \mathcal{P}_n,$$

which is consistent with the definition of QBCOP given in Chapter 1. As discussed in Chapter 1, the QBAP has applications in backboard wiring [219] and it generalizes the well-known bandwidth minimization problem [55, 177]. But to the best of our knowledge, no computational study on the QBAP is available in literature.

Let $A = (a_{ij})_{n \times n}$ and $B = (b_{ij})_{n \times n}$, then *the Koopmans-Beckmann QBAP (QBAP-KB)* is formulated as:

$$\text{Minimize } \max_{i,j \in N} a_{ij} b_{\pi(i)\pi(j)}$$

Subject to

$$\pi \in \mathcal{P}_n.$$

By choosing $q_{ijkl} = a_{ik} b_{jl}$ for all $i, j, k, l = 1, \cdots, n$, it can be verified that the QBAP-KB is case of the QBAP. Note that the input size of QBAP-KB is $O(n^2)$ whereas that of the QBAP is $O(n^4)$.

The NP-hardness of the QBAP can be easily obtained since its special cases - the QBAP-KB and the bandwidth minimization problem are well known NP-hard problems [55, 219].

7.2 Polynomially solvable cases and an exact algorithm

As a special case of the QBCOP, QBAP could adapt the polynomially solvable cases discussed in Chapter 4. We now give the following corollaries obtained by exploiting the structure of the QBAP, given that the bottleneck assignment problem can be solved in $O(n^{1.5}\sqrt{m})$ time [203].

Corollary 61. *The QBAP with a non-uniform, asymmetric, decomposable/multiplicable*

cost function can be solved in $O(n^{1.5}m^{1.5})$ *time.*

Corollary 62. *The QBAP with a uniform, asymmetric, decomposable/multiplicable cost function can be solved in* $O(n^{1.5}m^{1.5})$ *time.*

Corollary 63. *The QBAP with a non-uniform, symmetric, decomposable/multiplicable cost function can be solved in* $O(n^{1.5}\sqrt{m})$ *time.*

Corollary 64. *The QBAP with a uniform, symmetric, decomposable/multiplicable cost function can be solved in* $O(n^{1.5}m^{1.5})$ *time.*

Corollary 65. *The QBAP with a comparable cost function can be solved in* $O(n^{1.5}\sqrt{m})$ *time.*

Corollary 66. *Given a QBAP with cost matrix* Q *and let* r *be the critical index of the least-index optimal solution to the BAP(*$1, Q$*), which is defined as*

$$\begin{gathered} BAP(e,Q)\text{: } \textit{Minimize } \max\{q(e,f) \ : \ f \in M\} \\ \textit{Subject to} \\ M \in \mathcal{M}. \end{gathered}$$

then if Q *is row graded and* $q(1,r) \leq q(2,r) \leq \cdots \leq q(m,r)$*, then the QBAP can be solved in* $O(n^{1.5}\sqrt{m})$ *time.*

Algorithm 7.1 gives an exact method called *the quadratic assignment threshold algorithm* to solve the QBAP. It is adapted from the quadratic threshold algorithm discussed in Chapter 4 and in order to use this algorithm effectively, we need to solve the feasibility version of the QBAP (FQBAP) which is described as "given a constant K, does there exist a perfect matching M in G such that $\max_{e,f\in M} q(e,f) \leq K$? Note that ever since now, when discussing the QBAP, we assume without loss of generality, the cost matrix is symmetric.

According to Lemma 3 and Lemma 4 in Chapter 1, the FQBAP can be solved as an *assignment problem with conflict constraints (APC)*, or a QAP. Note that the APC is also a special case of the QAP and hence can be solved using any algorithms for the QAP. However, it may be solved efficiently by exploiting the problem structure.

Algorithm 7.1: Quadratic Assignment Threshold Algorithm

Input: A bipartite graph $G = (V_1 \cup V_2, E)$, $q(e, f)$ for $e, f \in E$, and an oracle $\alpha(.)$ which with input μ verifies if the FQBAP had a 'yes' or 'no' answer along with a feasible solution when the answer is 'yes'.
Construct an ascending arrangement $z_1 < z_2 < \cdots < z_p$ of distinct $q(e, f)$;
Let $l = 1,\ u = p$;
while $u - l > 0$ **do**
    $k = \lfloor \frac{(l+u)}{2} \rfloor$;
    if the FQBAP with threshold z_k return 'yes'
    then $l = k + 1$;
    else $u = k$;
end while
Output: z_l as the optimal objective function value and the corresponding optimal solution.

7.3 Assignment problem with conflict constraints

The general LCOPC is introduced in Chapter 1 and let us now specify the definition in the context of the APC. Given a bipartite graph $G = (V_1 \cup V_2, E)$, a cost c_e for every edge $e \in E$ and the conflict set $P \subseteq \{\{e, f\} : e, f \in E, e \neq f\}$, the APC is to:

$$\text{Minimize} \sum_{e \in E} c_e x_e$$

Subject to

$$M \in \mathcal{M}$$

$$x_e + x_f \leq 1 \text{ for } \forall \{e, f\} \in P$$

$$x_e = \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{otherwise ,} \end{cases}$$

Note that for the APC, the underlying graph G is not necessarily a complete bipartite graph. *The conflict graph* $\hat{G} = (\hat{V}, \hat{E})$ associated with the APC is then constructed as an undirected graph containing m nodes $n_1, n_2, \cdots, n_m$ corresponding to the m edges in G, and $(n_e, n_f) \in \hat{E}$ if and only if $\{e, f\} \in P$.

7.3.1 Computational complexity of the APC

It is proved by Darmann et al. [63] that the APC is strongly NP-hard even if the conflict graph is a collection of single edges. This result rules out (unless P=NP) the possibility of getting polynomially solvable special cases of the APC by restricting the structure of the conflict graph to reasonable nontrivial graphs. We then try to answer the question if the APC is solvable in polynomial time by restricting G instead of the conflict graph $\hat{G}$. Again the result is not very encouraging.

Theorem 67. *The APC is NP-hard even if G is a collection of disjoint 4-cycles.*

Proof. We reduce the maximum independent set problem to an APC on a collection of 4-cycles.

Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be a given graph with $\tilde{V} = \{1, 2, \ldots, n\}$. For each node $i \in \tilde{V}$, create a 4-cycle $\alpha_i - \beta_i - \gamma_i - \delta_i - \alpha_i$. Let the cost of (α_i, β_i) and (γ_i, δ_i) be 0 and the cost of (β_i, γ_i), (δ_i, α_i) be 1/2. The resulting graph G is shown in Figure 7.1. Note that any perfect matching in G must select either both edges (α_i, β_i) and (γ_i, δ_i), or both edges (α_i, δ_i) and (β_i, γ_i).

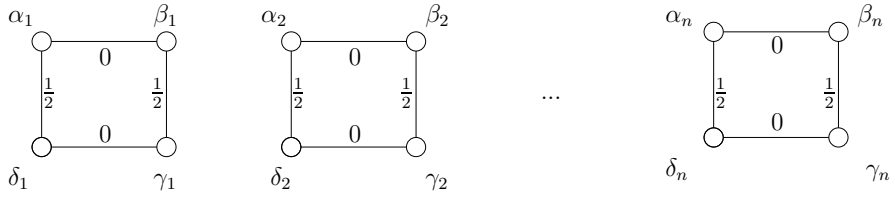

Figure 7.1: G: a collection of 4-cycles constructed from $\tilde{G}$

Let

$$P = \left\{ \{(\alpha_i, \beta_i), (\gamma_j, \delta_j)\}, \{(\alpha_i, \beta_i), (\alpha_j, \beta_j)\}, \{(\gamma_i, \delta_i), (\alpha_j, \beta_j)\}, \{(\gamma_i, \delta_i)(\gamma_j, \delta_j)\} : (i, j) \in \tilde{E} \right\}$$

be the conflict set. Thus we have constructed an instance of the APC. It is easy to verify that the maximum independent set in $\tilde{G}$ is of size k if and only if the APC has optimal objective function value $n - k$. Therefore, the APC on a collection of disjoint 4-cycles is NP-hard due to the NP-hardness of the maximum independent set problem [95]. □

Corollary 68. *(1) The maximization version of the APC on a collection of disjoint 4-cycles is NP-hard.*
(2)The APC(minimization/maximization) cannot be approximated within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$, even if G is a collection of disjoint 4-cycles.

Proof. By switching edge costs 1/2 and 0 and following the same way as in the proof of Theorem 67, (1) holds. The proof of (2) is also straight forward since the reduction discussed in the proof of Theorem 67 and Corollary 68(1) is approximation preserving and it is well known that the maximum independent set problem cannot be approximated within a factor of of $n^{1-\epsilon}$ for any $\epsilon > 0$ [95]. □

The MSTAC and AQBST have been discussed in Chapter 2 and 5. We have shown that they are NP-hard even on many sparse graphs. Let us consider corresponding adjacent-only version of the APC, in which e and f are adjacent in G for all $\{e, f\} \in P$.

Lemma 21. *The adjacent-only APC can be solved in $O(n^{1.5}\sqrt{m})$ time.*

Proof. Given an adjacent-only APC, $\{e, f\} \in P$ implies that e and f are adjacent in G. Since any APC feasible solution is a perfect matching in G, it does not allow adjacent edges. Therefore, all the conflict pairs in P can be dropped and the APC becomes a standard assignment problem and thus the result follows. □

Then by defining *the adjacent-only QBAP* as the QBAP where $q(e, f) = 0$ if e and f are adjacent, we have the following result.

Theorem 69. *The adjacent-only QBAP can be solved in $O(n^{1.5} \log n\sqrt{m})$ time.*

Next we consider a graph $G = (V, E)$ with the following properties:

(1) There exists subgraphs $G_1, G_2, \ldots, G_k$ such that any perfect matching M in G is a union of perfect matchings M_i in G_i, i.e. $M = \bigcup_{i=1}^{k} M_i$.

(2) Each G_i has a specified edge e_i such that e_i does not conflict with any edges of G_i, and the edges in the conflict graph $\hat{G}$ cover only $\{e_1, e_2, \ldots, e_k\}$, i.e. there are no conflict pairs containing edges in $E \setminus \{e_1, e_2, \ldots, e_k\}$. Figure 7.2 gives an example of a graph satisfying properties (1) and (2).

Theorem 70. *If G and the associated conflict graph satisfy properties (1) and (2), then the APC on G can be solved in polynomial time if and only if the maximum independent set problem on $\hat{G}$ can be solved in polynomial time.*

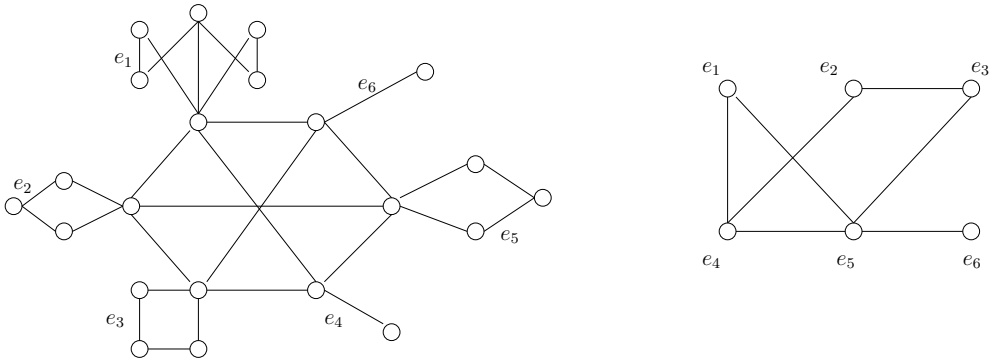

Figure 7.2: A graph G and an associated conflict graph satisfying properties (1) and (2)

Proof. Let w_i^0 be the cost of the smallest perfect matching M_i^0 in G_i that contains edge e_i and w_i^* be the cost of the minimum cost perfect matching M_i^* in G_i that does not contain e_i. By property (1) and (2), the APC reduces to the problem of deciding from each G_i, whether M_i^0 or M_i^* should be selected. If M_i^0 is selected, the conflict edge pairs should be avoided. Thus finding an optimal solution to the APC is equivalent to find an independent set S in $\hat{G}$ such that $\sum_{i \in S} w_i^0 + \sum_{i \notin S} w_i^*$ is minimized. This can be further stated as a standard maximum independent set problem on $\hat{G}$, where node i has weight $w_i^* - w_i^0$ for $i = 1, 2, \cdots, k$. If $\hat{S}$ is an optimal solution to this problem., then $M = \left(\bigcup_{i \in \hat{S}} M_i^0 \right) \cup \left(\bigcup_{i \notin \hat{S}} M_i^* \right)$ is an optimal solution to the APC with optimal objective function value $\sum_{i \in \hat{S}} (w_i^0 - w_i^*) + \sum_{i=1}^{k} w_i^*$. □

It may be noted that the maximum independent set problem can be solved in polynomial time on bipartite graphs and the more general perfect graph [61, 115, 114, 223].

Theorem 71. *If the conflict graph is collection of a fixed number of disjoint cliques, then the APC can be solved in polynomial time.*

Proof. Note that each node of the conflict graph represent an edge of G. If the conflict graph is a collection of disjoint cliques, from the edges representing each clique, at most one edge can be chosen in the matching. Assume there are k cliques, if $k < n$, then the APC is infeasible. If $k \geq n$, let the cardinality of the cliques be $q_1, q_2, \ldots, q_k$. Then from the ith clique, there are $1 + q_i$ choices, corresponding to selecting each node or selecting no nodes.

Thus, in total we have $\prod_{i=1}^{k}(1+q_i) = O(n^k)$ choices for selecting at most one node from each clique. Each such selection yields a subgraph of G, on which a minimum weighted perfect matching problem can be solved in polynomial time. Choosing the best solution so obtained provides an optimal solution to the APC. □

Corollary 72. *After dropping the adjacent conflict pairs from $\hat{G}$, if the remaining conflict graph is composed of a fixed number of disjoint cliques, then the APC can be solved in polynomial time.*

Note that in the above theorem and corollary, although we are fixing the number of cliques, the edges in each clique could be arbitrary.

7.3.2 Heuristics of the APC

Local Search (LS). Similar to the local search heuristic we developed for the MSTC in Chapter 2, the 2-exchange neighborhood is employed for the APC: given a perfect matching M in G, let π be the permutation corresponding to M, i.e. $M = \{(i, \pi(i) : i = 1, 2, \cdots, n\}$. Then for any $i, j = 1, 2, \cdots, n,\ i \neq j$, $M(i,j) = M - (i,\pi(i)) - (j,\pi(j)) + (i,\pi(j)) + (j,\pi(i))$ is also a perfect matching in G and the 2-exchange neighborhood is defined as $N^2(M) = \cup_{\substack{i,j \in M \\ i \neq j}} M(i,j)$. To take the conflict constraints into account, we let the objective function $\Pi(M) = \sum_{e \in M} c_e + \alpha \cdot g(M)$, where $g(M) = |\{\{e,f\} : \{e,f\} \in P \text{ and } e,f \in M\}|$ and α is a large number, so it acts as the penalty of having conflict pairs in the matching. *The k-exchange neighborhood* is a generalization of the 2-exchange neighborhood and used in developing local search algorithms for many hard combinatorial optimization problems. Clearly, a larger k number results in a wider search space but for the sake of efficiency, $N^2(M)$ usually provides good solutions in a reasonable CPU time.

Then starting from a perfect matching M, the local search algorithm searches the neighborhood $N^2(M)$, move to the best improving solution and check the neighborhood of the new solution for the best improving solution again. When there are no improving solutions in the neighborhood, a local optimum is reached and we update the global solution. This procedure is continued until the global solution is not improved for consecutive ρ times. Then we randomly move to a new solution and repeat the process. The LS algorithm will stop when the total number of iterations reaches a limit "max_iter". Note that the resulting solution M^* may not be always feasible and LS returns the best obtained solution and

$g(M^*)$ is the number of violated conflict constraints in M^*.

Tabu Search (TS). Like we discussed in Chapter 2, Tabu Search is a more advanced algorithm which conducts a series of neighborhood searches in a more systematic way, and it avoids moving between repeated neighborhoods. To do this, a tabu list consisting of at most θ 2-exchange pairs is maintained in a first-in-first-out way, where θ is a parameter. Initially the tabu list is empty and in each iteration, the algorithm searches the 2-exchange neighborhood of the current solution M, move to the best "non-tabu" solution $M(i,j)$ and add the 2-exchange pair (i,j) to the tabu list; or move to the best solution in the neighborhood, which is in the tabu list but satisfies the aspiration criteria. The process is continued until a prescribed number of iterations is completed and the output is the best solution obtained. During the algorithm, whenever the tabu list is full and a new 2-exchange pair occurs, the very first pair in the tabu list will be removed so the new pair can be added to the end of the list.

To develop variations of Tabu Search, we consider build a subgraph of the original graph G (and hence a subgraph of the conflict graph $\hat{G}$), so that the problem size is reduced and at the same time, well qualified edges are retained. Based on this idea, we construct two matrices in the tabu search algorithm: *the frequency matrix (F)*, which records the number of times a particular edge appeared in the local optimal solutions, and *the recency matrix (R)*, which tracks the recent D local optimal solutions and D is a predefined parameter. For example, if $F[1][3] = 5$, then in **all** the local optimum obtained, edge (1,3) is included for 5 times. While if $R[1][3] = 2$, then in **the recently** D local optimal solutions, edge (1,3) occurred twice.

Note that in each iteration of Tabu Search, we intend to accommodate the effect of having repeated edges in the solutions, so in each iteration k, the objective function $\tilde{\Pi}(\bar{M})$ is defined as

$$\tilde{\Pi}(\bar{M}) = \begin{cases} \Pi(\bar{M}) + \phi \cdot \Omega \cdot \Pi(M), & \text{if } \Pi(\bar{M}) \geq \Pi(M) \\ \Pi(\bar{M}), & \text{otherwise,} \end{cases}$$

where ϕ is a parameter and $\Omega = \sqrt{N}/k \cdot \sum_{(i,j)\in(\bar{M}\setminus M)} F[i][j]$. The square root factor $\sqrt{N}$, which was first proposed by Taillard [221] in the context of the vehicle routing problem, is used to compensate for instance size.

Frequency Guided Tabu Search (Tabu_F). Tabu_F uses the frequency matrix to capture the "most wanted" edges from the past few local optimal solutions. In this algorithm, we first perform tabu search for ψ iterations, then consider the frequency matrix F.

As mentioned in the previous example, each entry of F corresponds to an edge of G, we sort the non-zero entries in a non-increasing order, store them in a list and select up to ω elements, i.e. ω edges using the following rule: starting from the first element in the sorted list, whenever an edge e is selected, the edges in the list who are in conflict with or incident to e will be eliminated and we move to the next element. Let M_s be the set of selected edges, it represents a partial perfect matching in G. Then we remove M_s and the corresponding nodes from G to get a graph G_r. The APC on G_r is then formulated as an Integer Linear Programming (ILP) problem. Note that even though G_r is a subgraph of G, we need to take all conflict constraints into account. The ILP is solved by CPLEX and if it is feasible, a matching M_r in G_r is returned. Then $M_s \cup M_r$ is a feasible solution of the APC on G.

As an illustrative example, let us consider an APC on $G = (V_1, V_2, E)$, where $|V_1| = |V_2| = 4$. The conflict set

$$P = \{(1,1),(3,2);(1,1),(3,3);(2,2),(4,4);(1,3),(4,2);(2,3),(1,2);(3,2),(4,4);(2,3),(3,2)\}.$$

Assume that in the sorted list from the frequency matrix, the first few elements are (2,3), (2,4), (1,2) and (1,1), $\omega = 2$. Then $M_s = \{(2,3),\ (1,1)\}$, $G_r = (V_{1r}, V_{2r}, E_r)$ where $V_{1r} = \{3,4\}$, $V_{2r} = \{2,4\}$ and $E_r = \{(3,2),(3,4),(4,2),(4,4)\}$. The constraints of the ILP problem constructed on G_r are:

$$\begin{aligned}
&x_{32} + x_{34} = 1\\
&x_{42} + x_{44} = 1\\
&x_{32} + x_{42} = 1\\
&x_{34} + x_{44} = 1\\
&x_{32} + x_{44} \leq 1\\
&x_{32} \leq 0\\
&x_{ij} \in \{0,1\} \text{ for } i = 3,4 \text{ and } j = 2,4.
\end{aligned}$$

By solving the ILP we obtain $M_r = \{(3,4),\ (4,2)\}$. Then

$$M_s \cup M_r = \{(1,1),\ (2,3),\ (3,4),\ (4,2)\}$$

is a feasible solution of the APC on G.

After having $M_s \cup M_r$, we apply Tabu Search from there. This procedure is repeated and the algorithm will stop when a fixed number of iterations is reached. Algorithm 7.2 summarizes the details of the Tabu_F.

Algorithm 7.2: Tabu_F Algorithm

1: **Input:** A bipartite graph $G = (V_1 \cup V_2, E)$ with conflict set P and edge cost c_e for every $e \in E$;
2: Generate an initial solution M^0;
3: Let $i = 0,\ M^* = M^0$;
4: **while** $i \leq$ max_iter **do**
5: i++;
6: Let M^i be the best solution obtained by performing Tabu Search on M^{i-1};
7: Update the frequency matrix F;
8: **if** there are no improved solution for consecutive ψ_f steps
9: Construct G_r;
10: Use CPLEX to solve the APC on G_r as an ILP;
11: **if** the ILP is feasible and M_r is the returned solution, let $M' = M_r \cup M_s$;
12: **if** $\Pi(M') < \Pi(M^i)$, let $M^i = M'$;
if the ILP is infeasible, then go to line 4;
13: **if** $\Pi(M^i) < \Pi(M^*)$, let $M^* = M^i$;
14: **end while**;
15: **Output:** M^*;

Genetic Mutation Guided Tabu Search (Tabu_GM). Tabu_GM builds a subgraph of G by considering the recency matrix R. During Tabu Search, R is updated in the way that each time when TS obtains a local optimal solution M, $R[i][j]$ is increased by 1 if edge (i, j) appears in M. After having ψ_g local minimums, we construct a subgraph G_s of G so that G_s has all nodes of G, but only the edges corresponding to the non-zero entries of R.

Based on the concern that the current edges of G_s may not be enough for good mutations, we sort the entries of the frequency matrix F and add the first one third of edges to G_s, plus the edges in the best solution so far. Now G_s has a good level of density and we consider the APC on it, with conflict set $P_s = \{\{e, f\} \in P : e, f \in G_s\}$. The problem is formulated as an ILP and solved by CPLEX. If the solution returned is better than the best solution so far, we move to it and continue Tabu Search. Otherwise, another subset of the sorted list of F is selected and an alternative G_s, equivalently, an alternative APC is then constructed and solved. In case of getting no improvement after a certain number of iterations, the algorithm stops. We give a formal description of Tabu_GM in Algorithm 7.3.

Algorithm 7.3: Tabu_GM Algorithm

1: **Input:** A bipartite graph $G = (V_1 \cup V_2, E)$ with conflict set P and edge cost c_e for every $e \in E$;
2: Generate an initial solution M^0;
3: Let $i = 0, \lambda = 0$, $M^* = M^0$;
4: **while** $i <$ max_iter **do**
5: $i = i + 1$;
6: Let M^i be the best solution obtained by performing Tabu Search on M^{i-1};
7: Update the frequency matrix F and the recency matrix R;
8: $\lambda = \lambda + 1$;
9: **if** $\lambda = \psi_g$
10: iter=0;
11: **while** iter$< \chi$
12: Construct G_s;
13: Use CPLEX to solve the APC on G_s as an ILP;
14: **if** the ILP is feasible, let M' be the returned solution;
15: **if** $c(M') < c(M^i)$, let $M^i = M'$, go to line 4;
16: **else** iter = iter +1, go to line 11;
17: **if** the ILP is infeasible, go to line 4;
18: **end while;**
19: **if** $c(M^i) < c(M^*)$, let $M^* = M^i$;
20: **end while**;
21: **Output:** M^*;

7.3.3 Computational results with the APC algorithms

Besides the heuristics, we have considered two lower bounding algorithms for the APC, obtained by formulating it as a QAP and then directly implement the Gilmore-Lawler type lower bound [101, 169] and the Assad and Xu lower bound [11]). To test the performance of the lower bounding schemes and heuristics algorithms in terms of solution accuracy and computational efficiency, we code them in C++ and on a Pentium IV PC with 3 GHz CPU and 2 GB RAM.

The test problems are generated in the following way: let n range from 5 to 100, we generate complete bipartite bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2| = n,\ |E| = n^2$. The edge costs are random numbers uniformly distributed in [100, 200]. Then for the graphs, p distinct conflict pairs are randomly selected, where p value ranges from 5 to 9950. To avoid the redundant conflict pairs, we delete from the conflict set the pairs $\{e, f\}$ where e and f are adjacent. In total, there are 200 test instances.

As to the parameter settings, we have $max\_iter = 50n$, $\rho = 2n$, $\alpha = 200n$ in Local Search; $\theta = 7$, $max\_iter = 3000$, $\phi = 0.05$ in Tabu Search; $\omega = \min\{10, \lceil n \rceil\}$, $\psi_f = 2n$, $max\_iter = 20n$ in Tabu_FGC,; And $\psi_g = 3$, $\chi = \max\{2, \lceil n/15 \rceil\}$, $max\_iter = 20n$ in Tabu_GM.

To get optimal solutions, CPLEX with default options is used to solve the APC instances. Moreover, the APC linear programming relaxation is considered as another type of lower bounds, i.e. we directly ease the integer constraints to $0 \leq x_e \leq 1$ for any $e \in E$ in the APC formulation.

The algorithms are tested on all the 200 APC instances and 20 of the results are summarized in Table 7.1[1]. n is the number of nodes in V_1 where $G = (V_1, V_2, E)$ is a complete bipartite graph. p is the number of conflict pairs. "OPT" records the optimal solution, including the objective function value and CPU time taken by running CPLEX. The rest of the columns sequentially represent the 3 lower bounding schemes (LB): Gilmore-Lawler bound (GX), Assad and Xu lower bound (AX), linear programming relaxation (LP) and 4 heuristics (UB): LS, TS, TS_FGC, TS_GM we have discussed, each with the relative deviation and CPU time. The deviation is calculated as

$$100 \times \frac{|z - z^*|}{z^*},$$

where z is the returned value of the lower or upper bounds and z^* is the optimal objective function value of the problem.

From Table 7.1 we observe that AX returns better lower bounds than GL but takes much more time, which is intuitively reasonable since the latter can be viewed as a special case of AX. LP outperforms the other two types of lower bounds in terms of both accuracy and running time. However, for instances sets with (n, p) sizes (5, 5), (5, 15) and (10, 40), AX obtained better than the LP relaxation of the APC formulation. By comparing the upper bounds obtained by the heuristics, Tabu_FGC and Tabu_GM yield quite promising results, they behaved better than the CPLEX solver in terms of CPU time for 118 out of 200 instances and 113 out of 200 instances, respectively. Tabu_GM returns solutions with slightly larger deviations than that by Tabu_FGC but obviously takes less CPU time, and in fact this advantage grows significantly as the problem size increases.

In Figure 7.3 the improvement steps of the FCG are illustrated for an instance with

[1]The experiments reported in this table are carried out by my collaborator Dr. Temel Öncan.

n	p	OPT		LB						UB							
				GL		AX		LP		TS		LS		Tabu_FGC		TABU_GM	
		value	CPU	dev	CPU	dev	CPU	dev	CPU	dev	CPU	dev	CPU	dev	CPU	dev	CPU
5	5	601.8	0.0203	0.133	0.000	0.000	0.000	0.000	0.044	0.0000	0.0027	0.0000	0.0015	0.0000	0.0188	0.0000	0.0157
5	10	610.5	0.0000	1.016	0.000	0.293	0.002	0.246	0.008	0.0039	0.0008	0.0062	0.0016	0.0039	0.0062	0.0000	0.0141
5	15	607.5	0.0094	1.136	0.000	0.352	0.000	0.403	0.006	0.0000	0.0015	0.0000	0.0000	0.0000	0.0078	0.0000	0.0125
10	20	1114.8	0.0079	0.027	0.000	0.000	0.013	0.000	0.008	0.0000	0.0015	0.0005	0.0031	0.0000	0.0064	0.0000	0.0109
10	40	1123.4	0.0062	0.000	0.000	0.000	0.016	0.000	0.099	0.0000	0.0031	0.0000	0.0030	0.0000	0.0062	0.0000	0.0078
10	60	1124.6	0.0062	0.178	0.000	0.103	0.011	0.000	0.034	0.0000	0.0032	0.0000	0.0016	0.0000	0.0078	0.0000	0.0156
10	80	1152.2	0.0092	0.851	0.000	0.639	0.000	0.240	0.006	0.0030	0.0015	0.0025	0.0032	0.0000	0.0093	0.0002	0.0141
20	350	2135.9	0.0265	0.066	0.010	0.054	0.067	0.002	0.011	0.0010	0.0188	0.0013	0.0064	0.0001	0.0237	0.0008	0.0140
30	850	3137.2	0.0560	0.172	0.040	0.150	0.280	0.020	0.028	0.0027	0.1126	0.0028	0.0141	0.0002	0.0530	0.0027	0.0188
40	1550	4133.8	0.0906	0.022	0.100	0.021	0.795	0.002	0.044	0.0005	0.4407	0.0005	0.0377	0.0000	0.0891	0.0005	0.0347
50	500	5133.3	0.0548	0.010	0.210	0.009	1.952	0.000	0.034	0.0002	0.1374	0.0003	0.0563	0.0000	0.0766	0.0002	0.0501
50	1000	5132.9	0.0765	0.039	0.230	0.036	1.802	0.000	0.045	0.0006	0.3997	0.0007	0.0657	0.0000	0.1110	0.0006	0.0482
50	1500	5135.5	0.0922	0.012	0.280	0.011	2.001	0.002	0.052	0.0006	0.6887	0.0006	0.1078	0.0000	0.1141	0.0006	0.0531
50	2450	5137.5	0.1390	0.029	0.230	0.027	2.116	0.000	0.067	0.0010	0.7610	0.0026	0.1453	0.0000	0.1783	0.0008	0.0562
100	10	10114.3	0.3125	0.000	3.290	0.000	37.641	0.000	0.211	0.0000	0.2142	0.0000	0.3952	0.0000	0.3751	0.0000	0.3185
100	100	10109.8	0.2924	0.000	3.300	0.000	41.514	0.000	0.211	0.0000	0.2063	0.0000	0.3705	0.0000	0.3827	0.0000	0.3564
100	2000	10112.0	0.3968	0.000	3.360	0.000	49.827	0.000	0.252	0.0000	2.7408	0.0000	0.6081	0.0000	0.4672	0.0000	0.4422
100	5000	10112.9	0.7031	0.013	3.060	0.012	45.345	0.001	0.339	0.0004	8.8079	0.0006	0.9920	0.0000	1.0843	0.0008	0.4233
100	8000	10108.0	1.2532	0.004	3.130	0.004	44.759	0.000	0.414	0.0001	12.5157	0.0004	1.2124	0.0000	1.0031	0.0001	0.4296
100	9950	10116.5	1.3548	0.020	5.250	0.019	41.550	0.000	0.479	0.0007	18.2531	0.0009	1.2638	0.0000	1.5406	0.0005	0.4376

Table 7.1: APC: lower and upper bounds

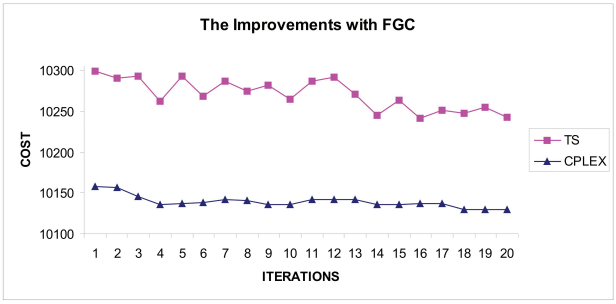

Figure 7.3: The improvements of the FGC

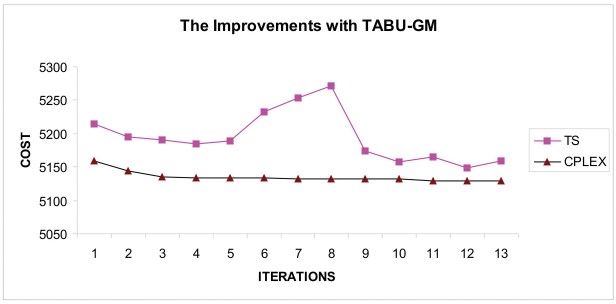

Figure 7.4: The improvements of the TABU-GM

$n = 100$ and $p = 10$ and similarly, we plot the improvement steps of GM in Figure 7.4, where the problem size is $n = 50$ and $p = 500$.

7.4 Heuristics for the QBAP

After discussing the APC algorithms, now let us get back to the QBAP.

In Chapter 6 a semi-greedy algorithm has been derived to solve the quadratic bottleneck knapsack problem. It is easily implementable and through the experimental results in Section 6.5, we have shown that the algorithm could quickly provides fair quality solutions. Thus for the QBAP, we keep using the semi-greedy algorithm to generate upper bounds. After adapting the details in context of the assignment problem, the steps are summarized in Algorithm 7.4.

Algorithm 7.4: The Semi-Greedy Algorithm (QBAP)

Input: A bipartite graph $G = (V_1 \cup V_2, E)$ with $|V_1| = |V_2| = n$, a cost $q(e, f)$ for every $e, f \in E$;
γ_iter=0;
Let $z^* = \infty,\ M^* = \emptyset$;
while γ_iter $<$ max_γ_iter **do**
  Choose a value of γ.
  iter=0;
  while iter<max-iter **do**
    Randomly choose $e^0 \in E$, let $M = \{e^0\}$;
    while $|M| < n$ **do**
      Let $\bar{E} = \{f : f \in (E \setminus M), f$ is not adjacent with any $e \in M\}$;
      For each $f \in \bar{E}$ let $p_f = \max\{q(e, f) : e \in M)\}$;
      Generate a random integer $r \in [1, \min\{\gamma, |\bar{E}|\}]$;
      Let p_{f^*} be the r-th smallest element in $\{p_f : f \in \bar{E}\}$;
      $M = M \cup \{f^*\}$;
    end while
    Let $z^0 = \max\{q(e, f) : e \in M, f \in M\}$;
    if $z^0 < z^*$ **then**
      $z^* = z^0$;
      $M^* = M$;
    end if
    iter=iter+1;
  end while
  γ_iter=γ_iter+1;
end while
Output M^* and z^*;

Another heuristic for the QBAP - *the approximate quadratic assignment threshold algorithm* (or the AQAT-heuristic) is obtained based on the the quadratic assignment threshold algorithm presented in Section 7.2, where in each iteration, the FQBAP is formulated as an APC and thus we could use the heuristics discussed in Section 7.3.2. To enhance the AQAT-heuristic, the search range can be reduced: note that any feasible solution of the QBAP do not contain adjacent edges, i.e. $q(e, f)$ does not contribute to the QBAP objective function value if e and f are adjacent. Therefore, instead of considering all the $q(e, f)$'s, we take only the costs corresponding to non-adjacent edges, and sort the distinct values in

an increasing order: $z_1 < z_2 < \cdots < z_p$. Moreover, a lower bound $\tilde{L}$ is obtained by solving

$$\begin{aligned} \tilde{L} = \ & \text{Minimize } \max\{z^e \ : \ e \in M\} \\ & \text{Subject to} \\ & M \in \mathcal{M}, \end{aligned}$$

where

$$\begin{aligned} z^e = \ & \text{Minimize } \max\{q(e,f) \ : f \in M\} \\ & \text{Subject to} \\ & e \in M \\ & M \in \mathcal{M}, \end{aligned}$$

and then used together with the objective function value U_s returned by the semi-greedy algorithm to further speed up the AQAT-heuristic. We give details in Algorithm 7.5.

Algorithm 7.5: AQAT-heuristic Algorithm

1: **Input:** A bipartite graph $G = (V_1 \cup V_2, E)$, $q(e, f)$ for $e, f \in E$.
2: Construct an ascending arrangement $z_1 < z_2 < \cdots < z_p$ of distinct $q(e, f)$'s, where e and f are not adjacent;
3: Obtain $\tilde{L}$ and U_s, find l^* and u^* such that $z_{l^*} = \tilde{L}$, $z_{u^*} = U_s$;
4: Let $l = l^*$, $u = u^*$;
5: **while** $u - l > 0$ **do**
6: $k = \lfloor \frac{(l+u)}{2} \rfloor$;
7: Construct an APC on G where the conflict graph $P = \{\{e, f\} : q(e, f) = z_\mu \text{ and } \mu > k\}$;
8: Call the APC heuristic to solve the FQBAP;
9: **if** the FQBAP has an "YES" answer
10: **then** $l = k + 1$;
11: **else** $u = k$;
12: **end while**
13: **Output:** z_l as the objective function value and the corresponding solution.

7.5 Computational results for the QBAP

For the QBAP, test instances have been generated for the general QBAP and its two special cases: the QBAP-KB and the bandwidth minimization problem (BM). Again we consider complete bipartite graph $G = (V_1, V_2, E)$, where $|V_1| = |V_2| = n$. For each G, we assume the cost matrix $Q = (q(i,j))$ is symmetric and $q(i,i) = 0$ for $i = 1, 2, \cdots, n^2$. A parameter *pct* is used to control the percentage of non-zero entries of Q.

In the general QBAP, we let $n = 10$, 15, 20, 25, 30 and *pct* = 50, 75, 100. Note that for $n = 30$ and *pct* = 100, there are 404550 $q(i,j)$ values and thus the problems are of reasonably large size, and the memory needed for storing the problem is $O(n^4)$. For each combination $(n,\ pct)$, we generate 10 problems i.e. 10 cost matrices where $q(i,j)$ values are randomly selected integers ranging from 0 to $n^4/4$. So overall there are 150 instances for the general QBAP.

In the QBAP-KB, since $Q = A \times B$ where A and B are $n \times n$ matrices, the memory required for the problem is reduced from $O(n^4)$ to $O(n^2)$. Therefore we generated larger sized problems by having $n = 10$, 20, 30, 40, 50 and *pct* = 50, 75, 100. For each $(n,\ pct)$, 10 symmetric matrices $A = (a(i,j))$ and $B = (b(i,j))$ are generated where $a(i,i) = b(i,i) = 0$ for $i = 1, 2, \cdots, n$, and $a(i,j),\ b(i,j)$ are random integers in $[0, n^2/2]$ for $i, j = 1, 2, \cdots, n, i \neq j$.

To construct instances for the BM, we let $n = 50$, 100, 150, 200 and *pct* = 2, 5, 10. For each $(n,\ pct)$, 10 different 0-1 matrices $A = (a(i,j))_{n \times n}$ are generated such that *pct*% of the non-diagonal A entries have value 1.

On all the 450 instances, we apply the lower bound, the semi-greedy algorithm and two versions of the AQAT_heuristics: the *AQAT_GM* uses the Tabu_GM algorithm as the APC solver; the AQAT_CPX solves the APC using CPLEX with a time limit τ. Note that for the QBAP-KB and the BM, we do not execute the sorting step (step 2 and 3) in Algorithm 7.5 but simply let $u = U_s$ and $l = \tilde{L}$ in step 4, and the construction of the APC conflict set in step 7 can be replaced by "$P = \{\{e,f\} : e = (i,j),\ f = (t,h),\ a(i,t)b(j,h) > k$" or "$P = \{\{e,f\} : e = (i,j), f = (t,h),\ a(i,t) = 1$ and $|j - h| > k$". Accordingly, the "while loop" stop when $u - l > 1$ because the objective function values are integers.

In the semi-greedy algorithm, we set the parameters max_γ_iter = 3,γ = 1, 3, 5 and max_iter = 5. When selecting the parameters for the AQAT_GM, we run a few experiments with different options and based on the trade off between the accuracy and computational

n	pct	p		LB		Semi-Greedy		AQAT_GM			AQAT_CPX	
		value	cpu	value	cpu	value	cpu	value	cpu	GM	value	cpu
10	50	1376.6	0	0	0.004	1052.6	0	564.3	4.26	17	562.8	15.915
	75	1736.1	0	0	0.011	1401.9	0	1065.9	4.371	23.2738	1051.6	19.913
	100	1992	0.002	409.3	0.038	1722.6	0.002	1389	5.378	26.881	1388.8	16.327
15	50	7240.4	0.002	0	0.018	5887.2	0.002	3929.3	29.784	25.9167	3631	5073.97
	75	9104.4	0.006	0	0.034	8042.4	0.006	6132.1	29.694	26.8571	5927.1	4855.61
	100	10272.1	0.007	1452.4	0.287	9212.7	0.007	7603	32.49	17.5	7356.3	5309.35
20	50	21888.3	0.01	0	0.066	19655.1	0.01	14547.2	53.772	31	14657.1	6192.55
	75	26465.7	0.014	0	0.073	24652.1	0.014	20798.1	62.267	23.9722	20755.7	6202.41
	100	29140.8	0.019	4204.2	1.244	27548.2	0.019	24223.7	49.169	19.8809	24393	5798.91
25	50	30671.1	0.028	0	0.196	27860.9	0.028	22533.7	49.544	19.6667	23190.9	6629.87
	75	32231.8	0.034	0	0.142	30071.2	0.034	26638.3	77.422	10.0404	27042.8	6555.84
	100	32631.6	0.046	4044.6	3.841	31103.8	0.046	28307.7	78.448	16.0516	28747.6	6397.11
30	50	32662	0.053	0	10.489	30556.2	0.053	25442.3	75.377	17.7222	26283.9	7018.06
	75	32762.8	0.077	0	21.652	31457	0.077	27882.5	68.008	10.0559	28603.4	6410.28
	100	32768.7	0.103	7694.1	36.755	31629.2	0.103	29089.4	62.428	7.78968	29547	6415.52

Table 7.2: QBAP: lower and upper bounds

time, the number of iterations in Tabu_GM is set to be 15 and in each iteration the iteration limits inside the tabu search is 200, $\psi_g = 3$. In AQAT_GM again the instances are first tested by trying $\tau = 1, 2, 3, 4, 5, 6$ inside CPLEX and it turns out that if we let CPLEX stop after 1 minute, the algorithm returns a slightly worse (averagely 10%) solution than that obtained by setting other time limits but the CPU time is much less (down to 14%). Therefore we let $\tau = 1$ in the following computations.

Table 7.2, 7.3 and 7.4 sequentially summarize the results of running the algorithms on the QBAP, QBAP-KB and BM, for each (n, pct), we report the average values and CPU time among the 10 instances. Column p in Table 7.2 represents the number of distinct costs in the QBAP, which is obtained after detecting and sorting the distinct costs, and the CPU time this step took. In the column "AQAT_GM", we use variable "GM" to reflect the number of times the genetic mutation worked, so GM is calculated as t_1/t_2, where t_1 is the number of times a feasible solution is obtained by running CPLEX on G_s and t_2 is the total number of times a feasible solution is obtained during the binary search. Therefore, the higher GM value is, the better genetic mutation acts. If $GM = 0$ we conclude that Tabu_GM reduces to the original tabu search algorithm. Table 7.3 and 7.4 do not have column p as the sorting step has been skipped and in Table 7.4 we omitted column "LB" since it returned value 0 all the time. Note that to accommodate the sorting step, the column "value" in Table 7.2 reflects positions of the objective values, while the column "value" in Table 7.3 and 7.4 records real objective function values of the returned solutions.

n	pct	LB		Semi-Greedy		AQAT_GM			AQAT_CPX	
		value	cpu	value	cpu	value	cpu	GM	value	cpu
10	50	0	0.012	968	0	420.5	1.598	28.9127	419.3	0.983
	75	0	0.026	1380.4	0.002	738.1	1.388	10.2976	734.7	0.916
	100	0	0.037	1523.2	0	951	1.343	5.77381	949.7	1.093
20	50	0	0.598	26031.6	0.019	13600.3	8.466	16.671	12910.5	1408.84
	75	0	1.513	28617.6	0.018	17201.6	9.946	33.5793	16393.3	1592.1
	100	3861.5	2.552	30392.5	0.02	21025.7	8.97	14.2399	20200.2	1537.25
30	50	0	7.326	151723	0.09	89017.5	27.24	13.4828	89218	1819.36
	75	0	19.031	158726	0.09	107815	24.764	10.645	108110	1805.24
	100	31526.2	31.918	163334	0.089	120004	28.855	21.303	121489	1663.11
40	50	0	42.736	505539	0.276	317990	88.58	8.05429	337480	2046.76
	75	0	122.628	525766	0.27	376659	94.872	10.6928	388710	2341.49
	100	120007	189.664	537290	0.31	408154	111.583	12.5565	421025	2176.42
50	50	0	198.832	1239420	0.662	823749	285.817	7.66667	927665	2803.34
	75	0	493.291	1331020	0.65	955263	278.806	10.8263	1039280	2742.08
	100	294619	769.454	1228280	0.648	99480.9	251.772	18.7544	1039280	2621.23

Table 7.3: QBAP-KB: lower and upper bounds

n	pct	Semi-Greedy		AQAT_GM			AQAT_CPX	
		value	cpu	value	cpu	GM	value	cpu
50	2	46.2	3.207	5.9	24.89	0	2.4	124.361
	5	46.9	2.682	13.5	29.116	0	9.6	409.679
	10	47.9	2.127	20.1	38.503	8.33333	18.7	399.08
100	2	97.1	67.703	29.5	435.688	0	31.5	481.409
	5	97.8	48.716	51.6	483.306	0	55.7	478.305
	10	98.5	36.144	66.1	611.29	0	69.4	504.631
150	2	147.7	390.415	112.6	2450.68	0	-	-
	5	148.5	220.939	124.7	3452.53	0	-	-
	10	148.6	169.438	131.9	3805.25	0	-	-
200	2	198.1	1374.25	172.3	9284.91	0	-	-
	5	197.6	1298.36	179	4458.25	0	-	-
	10	198.5	1306.98	185	7368.18	0	-	-

Table 7.4: BM: lower and upper bounds

The tables implies that the lower bounds and semi-greedy upper bounds can be quickly computed but are relatively weak by comparing with the objective function values returned by the AQAT_GM and AQAT_CPX. The lower bounds are non-trivial only when the cost matrix Q have the least number of zero entries. The AQAT_GM heuristic has been superior to the AQAT_CPX in terms of CPU time in 36 out of 42 records and when considering the objective function values, we see that the AQAT_GM solutions have slightly larger objective function values than that of the AQAT_CPX solutions in the beginning but as the problem size grew, the former became closer to and then smaller than the latter. In the bandwidth minimization problem, CPLEX failed to solve the constructed APC problem when $n = 150$ and 200. Genetic mutation seems worked better on smaller sized QBAP and QBAP-KB but it did not show much potential in the bandwidth minimization problem, this is probably due to the selection of the parameters. For example, the maximum number of iterations in the algorithm could be increased so the AQAT_GM would keep the advantage on CPU time and in the meanwhile, enhance the effect brought by genetic mutation.

Chapter 8

Conclusion

With its various extensively studied special cases such as assignment problem, minimum spanning tree problem, traveling salesman problem, knapsack problem, etc., linear combinatorial optimization problems (LCOP) is probably the best-known type of optimization problem. Besides having a cost for each single element, if a quadratic cost is introduced for each pair of elements, then the LCOP is generalized to the quadratic combinatorial problem (QCOP). The QCOP has also been very well investigated in the past few decades in the context of quadratic assignment problem (QAP), unconstrained quadratic problem (UQP), quadratic minimum spanning tree problem (QMST) and quadratic knapsack problem (QKP). If the "min-sum" objective function of the QCOP is replaced by a "min-max" format, then we obtain the quadratic bottleneck combinatorial optimization problem (QBCOP). The QBCOP is a generalization of many well know problems such as the quadratic bottleneck assignment problem, the balanced problem and the bottleneck combinatorial optimization problem, and most of the QCOP applications have a natural interpretation in the context of the QBCOP, since the former measures the "average" behavior while the latter can be viewed as a measure of the "worst-case" behavior.

In this thesis we have studied the general QBCOP, its special cases - quadratic bottleneck spanning tree problem (QBST), quadratic bottleneck knapsack problem (QBKP) and quadratic bottleneck assignment problem (QBAP). Also as closely related problems, minimum spanning tree problem with conflict constraints (MSTC), edge clique partitioning problem (Max-ECP) and assignment problem with conflict constraints (APC) are investigated and we have implemented the results on these problems into the QBCOPs.

We first considered the minimum spanning tree problem with conflict constraints (MSTC)

since it plays an important role on solving the QBST. Various NP-hard and polynomially solvable special cases have been identified, high quality lower bounds, heuristic algorithms and feasibility tests have been derived, followed by promising experimental results.

During the development of MSTC lower bound, the Max-ECP is arisen and its solution directly effects that of the MSTC. Thus in Chapter 3, we have discussed the Max-ECP and shown that if the maximum clique problem can be solved by a polynomial time ϵ-approximation algorithm, then the Max-ECP can be solved by a polynomial time $\frac{2(p\epsilon-1)}{p-1}$-approximation algorithm for any fixed integer $p \geq 2$. As a consequence a polynomial time approximation algorithm for the Max-ECP is obtained, with performance ratio bounded by $O\left(\frac{2}{p-1}(p\frac{n(\log\log n)^2}{(\log n)^3}-1)\right)$ where $p \geq 2$ is a fixed integer and n is the number of nodes in the associated graph. This improves the best known bound on the performance ratio of an approximation algorithm for the Max-ECP. The maximum edge bi-clique partitioning problem has also been considered and we have shown that the greedy algorithm computes $4-\frac{4}{n}$ optimal solution if the only balanced bi-cliques are counted and for the case of general bi-cliques, the greedy algorithm can be arbitrarily bad.

The general QBCOP is discussed in Chapter 4. We have illustrated the relation between the QBCOP and the QCOP by showing that the QBCOP can be formulated as a single QCOP with exponentially increasing cost coefficients, a weak duality theorem and an asymptotic result have been proved. Various nontrivial polynomially solvable cases have been identified whenever an associated linear bottleneck problem can be solved in polynomial time, and an exact algorithm to solve the QBCOP is then developed, the complexity of which depends on the answer of the feasibility version of the QBCOP. Therefore, for the rest of the thesis where the QBCOP special cases are considered, those general purpose results can be adapted by exploiting special problem structures. As a byproduct, we have obtained a new matroid characterization.

Then consequentially, we have investigated three QBCOP special cases - the QBST, QBKP and QBAP. Each problem has been studied in terms of computational complexity, polynomially solvable cases, exact and heuristic algorithms, and followed by computational results. Various heuristics are developed including local search, semi-greedy algorithm, tabu thresholding, tabu search, etc., and in order to solve the FQBST, FQBKP and FQBAP, we have formulated them into the MSTC, the maximum weight independent set and the assignment problem with conflict constraints (APC) and called the heuristics for these problems as a subroutine in the approximate quadratic threshold algorithm. Especially, the

APC has not been studied computationally in literature. In Chapter 7, complexity results, polynomially solvable cases and heuristic algorithms for the APC are discussed in a separate section. Two variations of Tabu Search - the frequency guided tabu search (Tabu_FG) and the genetic mutation guided tabu search (Tabu_GM) are developed for the APC and later on we used the Tabu_GM in the algorithm for the QBAP.

Overall, we have examined the QBCOP and its variations, generated plenty of theoretical and computational results, and found several new research challenges.

As future work, it will be interesting to further investigate the effect of logical constraints on a given sets of pairs $\{e, f\}$ of elements of E such as (a) "either e or f", (b) "either both e, f or none", (c) "at least one of e or f", etc. and discuss the impact of logical constraints on some special LCOPs in the context of

1. Polyhedral characterization and valid inequalities identification
2. Polynomially solvable special cases that exploiting underlying graph structures or special properties of the cost matrix
3. Identify the "boundary" between hard and easy cases
4. Approximation algorithms
5. Study inter relationship between graphs defining logical constraints and graphs on which the original problems are defined

We believe that this investigation will provide deeper understanding of the structure of these problems, which will lead to development of efficient exact and approximation algorithms, thus enhance the applicability of the model in solving some real world problems.

Bibliography

[1] E. Aarts and J. K. Lenstra. Local search in combinatorial optimization. 1997.

[2] W. P. Adams and T. A. Johnson. Improved linear programming-based lower bounds for the quadratic assignment problem. In *Quadratic assignment and related problems: DIMACS Workshop, May 20-21, 1993*, volume 16, page 43. American Mathematical Society, 1994.

[3] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber. The angular-metric traveling salesman problem. *SIAM Journal on Computing*, 29:697, 2000.

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

[5] E. Amaldi, G. Galbiati, and F. Maffioli. On minimum reload cost paths, tours, and flows. *Networks*, 57:254–260, 2011.

[6] A. Amberg, W. Domschke, and S. Voß. Capacitated minimum spanning trees: algorithms using intelligent search. *Combinatorial Optimization: Theory and Practice*, 1:9–40, 1996.

[7] M. M. Amini, B. Alidaee, and G .A. Kochenberger. A scatter search approach to unconstrained quadratic binary programs. In *New ideas in optimization*, pages 317–330. McGraw-Hill Ltd., UK, 1999.

[8] Y. P. Aneja, V. Aggarwal, and K. P. K. Nair.

[9] G. C. Armour, E. S. Buffa, and T. E. Vollman. Allocating facilities with craft. *Harvard Business Review*, 42:136–58, 1964.

[10] B. Aspvall, F. M. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing letters*, 8:121–123, 1979.

[11] A. Assad and W. Xu. The quadratic minimum spanning tree problem. *Naval Research Logistics*, 39:399–417, 1992.

[12] E. Balas and J. B Mazzola. Quadratic 0-1 programming by a new linearization. In *Joint ORSA/TIMS National Meeting*, 1980.

[13] E. Balas and J. B. Mazzola. Nonlinear 0–1 programming: I. linearization techniques. *Mathematical Programming*, 30:1–21, 1984.

[14] E. Balas and J.B. Mazzola. Nonlinear 0–1 programming: Ii. dominance relations and algorithms. *Mathematical Programming*, 30:22–45, 1984.

[15] E. Balas and C. S. Yu. On graphs with polynomially solvable maximum-weight clique problem. *Networks*, 19:247–253, 1989.

[16] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87:385–399, 2000.

[17] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36:493–513, 1988.

[18] F. Barahona, M. Juenger, and G. Reinelt. Experiments in quadratic 0–1 programming. *Mathematical Programming*, 44:127–137, 1989.

[19] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on computing*, 6:126–126, 1994.

[20] M. S. Bazaraa and O. Kirca. A branch-and-bound-based heuristic for solving the quadratic assignment problem. *Naval Research Logistics Quarterly*, 30:287–304, 1983.

[21] M. S. Bazaraa and H. D. Sherali. Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. *Naval Research Logistics Quarterly*, 27:29–41, 1980.

[22] M. S. Bazaraa and H. D. Sherali. On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. *Journal of the Operational Research Society*, pages 991–1003, 1982.

[23] J. A. Bennell and K. A. Dowsland. A tabu thresholding implementation for the irregular cutting problem. *International Journal of Production Research*, 37:4259–4275, 1999.

[24] A. Billionnet and F. Calmels. Linear programming for the 0-1 quadratic knapsack problem. *European Jounral of Operational Research*, 92:310–325, 1996.

[25] A. Billionnet and A. Sutter. Minimization of a quadratic pseudo-boolean function. *European Journal of Operational Research*, 78:106–115, 1994.

[26] H. Bodlaender and K. Jansen. On the complexity of scheduling incompatible jobs with unit-times. *Mathematical Foundations of Computer Science 1993*, pages 291–300, 1993.

[27] H. L. Bodlaender, K. Jansen, and G. J. Woeginger. Scheduling with incompatible jobs. *Discrete Applied Mathematics*, 55:219–232, 1994.

[28] A. Bookhold. A contribution to quadratic assignment problems. *Optimization*, 21:933–943, 1990.

[29] E. Boros and P. L. Hammer. The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33:151–180, 1991.

[30] E. Boros, P. L. Hammer, M. Minoux, and D. J. Rader. Optimal cell flipping to minimize channel density in vlsi design and pseudo-boolean optimization. *Discrete applied mathematics*, 90:69–88, 1999.

[31] C. Brezovec, G. Cornu*éjols*, and F. Glover. Two algorithms for weighted matroid intersection. *Mathematical Programming*, 36:39–53, 1986.

[32] C. Brezovec, G. Cornu*éjols*, and F. Glover. A matroid algorithm and its application to the efficient solution of two optimization problems on graphs. *Mathematical Programming*, 42:471–487, 1988.

[33] A. Brüngger, A. Marzetta, J. Clausen, and M. Perregaard. Joining forces in solving large-scale quadratic assignment problems in parallel. In *ipps*, pages 418–427. Published by the IEEE Computer Society, 1997.

[34] R. E. Burkard. Quadratische bottleneckprobleme. *Operations Research Verfahren*, 18:26–41, 1974.

[35] R. E. Burkard. Locations with spatial interactions: the quadratic assignment problem. *Discrete Location Theory*, pages 387–437, 1991.

[36] R. E. Burkard and T. Bönniger. A heuristic for quadratic boolean programs with applications to quadratic assignment problems. *European Journal of Operational Research*, 13:374–386, 1983.

[37] R. E. Burkard, E. Çela, P. M. Pardalos, and L. S. Pitsoulis. The quadratic assignment probelm.

[38] R. E. Burkard, E. Çela, G. Rote, and G. J. Woeginger. The quadratic assignment problem with an anti-monge and toeplitz matrix: Easy and hard cases. *Mathematical Programming*, 82:125–158, 1998.

[39] R. E. Burkard, J. Krarup, and P. M. Pruzan. Efficiency and optimality in minsum and minmax 0-1 programming problems. *Journal of the Operational Research Society*, 33:137–151, 1982.

[40] R.E. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. Society for Industrial Mathematics, 2009.

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

[42] R.E. Burkard and J. Offermann. Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme. *Zeitschrift für Operations Research*, 21:B121–B132, 1977.

[43] Y. Cai, J. Wang, J. Yin, and Y. Zhou. Memetic clonal selection algorithm with eda vaccination for unconstrained binary quadratic programming problems. *Expert Systems With Applications*, 2010.

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

[45] P. M. Camerini, F. Maffioli, S. Martello, and P. Toth. Most and least uniform spanning trees. *Discrete Applied Mathematics*, 15:181–197, 1986.

[46] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11:125–137, 1999.

[47] D. Castelino and N. Stephens. A surrogate constraint tabu thresholding implementation for the frequency assignment problem. *Annals of Operations Research*, 86:259–270, 1999.

[48] A. Cayley. A theorem on trees. *Quarterly Journal of Mathematics*, 23:376–378, 1889.

[49] E. Çela. *The quadratic assignment problem: theory and algorithms*, volume 1. Kluwer Academic Pub, 1998.

[50] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45:41–51, 1985.

[51] P. Chaillou, P. Hansen, and Y. Mahieu. Best network flow bounds for the quadratic knapsack problem. *Combinatorial Optimization*, pages 225–235, 1989.

[52] J. Chakrapani and J. Skorin-Kapov. Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, 41:327–341, 1993.

[53] J. Chakrapani and J. Skorin-Kapov. A constructive method to improve lower bounds for the quadratic assignment problem. In *Quadratic assignment and related problems: DIMACS Workshop, May 20-21, 1993*, volume 16, pages 161–171. American Mathematical Society, 1994.

[54] B. Chen. Special cases of the quadratic assignment problem. *European Journal of Operational Research*, 81:410–419, 1995.

[55] P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs. The bandwidth problem for graphs and matricesa survey. *Journal of Graph Theory*, 6:223–254, 1982.

[56] N. Christofides and M. Gerrard. A graph theoretic analysis of bounds for the quadratic assignment problem. *Studies on graphs and discrete programming*, pages 61–68, 1981.

[57] J. Clausen and M. Perregaard. Solving large quadratic assignment problems in parallel. *Computational Optimization and Applications*, 8:111–127, 1997.

[58] D. T. Connolly. An improved annealing scheme for the qap. *European Journal of Operational Research*, 46:93–100, 1990.

[59] K. Conrad. *Das quadratische Zuweisungsproblem und zwei seiner Spezialfälle*. Mohr Siebeck, 1971.

[60] R. Cordone and G. Passeri. Heuristic and exact approaches for the quadratic minimum spanning tree problem. In *CTW 2008 seventh cologne twente workshop on graphs and combinatorial optimization, Gargano, University of Milan, Italy*, pages 52–55, 2008.

[61] B. Courcelle, J. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33:125–150, 2000.

[62] A. Darmann, U. Pferschy, and J. Schauer. Minimal spanning trees with conflict graphs. *Optimization online*, 2009.

[63] A. Darmann, U. Pferschy, S. Schauer, and G.J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Optimization online*, 2009.

[64] T. Öncan and A. Punnen. The quadratic minimum spanning tree probelm: A lower bounding procedure and an efficient search algorithm. *Computers & Operations Research*, 37:1762–1773, 2010.

[65] V. G. Deineko and G. J. Woeginger. A solvable case of the quadratic assignment problem. Technical report, Institute of Mathematics, Technical University Graz, Austria, 1996.

[66] U. Derigs. On three basic methods for solving bottleneck transportation problems. *Naval Research Logistics Quarterly*, 29:505–515, 1982.

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

[68] A. Dessmark, J. Jansson, A. Lingas, E.M. Lundell, and M. Persson. On the approximability of maximum and minimum edge clique partition problems. *International Journal of Foundations of Computer Science*, 18:217–226, 2007.

[69] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26:29–41, 1996.

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

[71] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971.

[72] J. Edmonds. Matroid intersection. *Annals of Discrete Mathematics*, 4:39–49, 1979.

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

[74] L. Epstein and A. Levin. On bin packing with conflicts. *Approximation and Online Algorithms*, pages 160–173, 2007.

[75] G. Erdoğan. *Quadratic Assignment Problem: Linearizations And Polynomial Time Solvable Cases*. PhD thesis, BILKENT UNIVERSITY, 2006.

[76] T. Feder. Network flow and 2-satisfiability. *Algorithmica*, 11:291–319, 1994.

[77] U. Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18:219–225, 2005.

[78] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[79] E. Fernandez, R. Garfinkel, and R. Arbiol. Mosaicking of aerial photographic maps via seams defined by bottleneck shortest paths. *Operations Research*, 46:293–304, 1998.

[80] P. Festa and M. G. C. Resende. Grasp: An annotated bibliography. *Essays and surveys in metaheuristics*, pages 325–367, 2002.

[81] A. Figueroa, A. Goldstein, T. Jiang, M. Kurowski, A. Lingas, and M. Persson. Approximate clustering of fingerprint vectors with missing values. *In Proc. 11th Computing: The Australasian Theory Symposium (CATS)*, 41:57–60, 2005.

[82] G. Finke, R. E. Burkard, and F. Rendl. Quadratic assignment problems. *Surveys in combinatorial optimization*, pages 61–82, 1987.

[83] A. Fischer and C. Helmberg. The symmetric quadratic traveling salesman problem. *Optimization online*, 2011.

[84] A. M. Frieze and J. Yadegar. On the quadratic assignment problem. *Discrete applied mathematics*, 5:89–98, 1983.

[85] A. M. Frieze, J. Yadegar, S. El-Horbaty, and D. Parkinson. Algorithms for assignment problems on an array processor. *Parallel computing*, 11:151–162, 1989.

[86] R. Fulkerson, I. Glicksberg, and O. Gross. A production line assignment problem. Technical report, The Rand Corporation, Sta. Monica, CA, USA, 1953.

[87] E. Y. Gabovich. Constant discrete programming problems on subsititution sets. *Kibernetika*, 5:128–134, 1976.

[88] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6:109–122, 1986.

[89] H. N. Gabow and R. E. Tarjan. Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9:411–147, 1988.

[90] G. Galbiati, S. Gualandi, and F. Maffioli. On minimum reload cost cycle cover. *Electronic Notes in Discrete Mathematics*, 36:81–88, 2010.

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

[92] G. Gallo, P. L. Hammer, and B. Simeone. Quadratic knapsack problems. *Combinatorial Optimization*, pages 132–149, 1980.

[93] L. M. Gambardella, E. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the operational research society*, 50:167–176, 1999.

[94] J. Gao and M. Lu. Fuzzy quadratic minimum spanning tree problem. *Applied Mathematics and Computation*, 164:773–788, 2005.

[95] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman & Co. New York, NY, USA, 1979.

[96] R. S. Garfinkel. An improved algorithm for the bottleneck assignment problem. *Operations Research*, 19:1747–1751, 1971.

[97] R. S. Garfinkel and K. C. Gilbert. The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. *Journal of the ACM*, 25:435–448, 1978.

[98] J. W. Gavett and N. V. Plyter. The optimal assignment of facilities to locations by branch and bound. *Operations Research*, pages 210–232, 1966.

[99] A. M. Geoffrion and G. W. Graves. Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/lp approach. *Operations Research*, pages 595–610, 1976.

[100] L. Georgiadis. Arborescence optimization problems solvable by edmonds' algorithm. *Theoretical Computer Science*, 301:427–437, 2003.

[101] P. C. Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the society for industrial and applied mathematics*, 10:305–313, 1962.

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

[103] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533–549, 1986.

[104] F. Glover. Tabu search: part i. *ORSA Journal on Computing*, 1:190–206, 1989.

[105] F. Glover. Tabu search: part ii. *ORSA Journal on Computing*, 2:4–32, 1990.

[106] F. Glover. Tabu thresholding: improved search trajectories by non-monotonic search trajectories. *ORSA Journal on Computing*, 7:426–442, 1995.

[107] F. Glover, B. Alidaee, C. Rego, and G. Kochenberger. One-pass heuristics for large-scale unconstrained binary quadratic problems. *European Journal of Operational Research*, 137:272–287, 2002.

[108] F. Glover and G. Kochenberger. Solving quadratic knapsack problems by reformulation and tabu serch, single constraint case. In P. M. Pardalos, A. Migdalas, and R. E. Burkard, editors, *Combinatorial and global optimization*, volume 14, 2002. World scientific, Singapore.

[109] F. Glover, G.A. Kochenberger, and B. Alidaee. Adaptive memory tabu search for binary quadratic programs. *Management Science*, 44:336–345, 1998.

[110] F. Glover and M. Laguna. *Tabu search*, volume 1. Kluwer Academic Pub, 1998.

[111] F. Glover, Z. Lü, and J.K. Hao. Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR-Quarterly Journal of Operations Research*, 8:239–253, 2010.

[112] L. Gourvès, A. Lyra, C. Martinhon, and J. Monnot. The minimum reload s-t path, trail and walk problems. *Discrete Applied Mathematics*, 158:1404–1417, 2010.

[113] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7:43–57, 1985.

[114] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. *Topics on perfect graphs*, 21:325–356, 1984.

[115] M. Grötschel, L. Lovász, and A. Schrijver. Geometric algorithms and combinatorial optimization. *Algorithms and Combinatorics*, 1988.

[116] V. P. Gulati, S. K. Gupta, and A. K. Mittal. Unconstrained quadratic bivalent programming problem. *European Journal of Operational Research*, 15:121–125, 1984.

[117] S. K. Gupta and A. K. Mittal. A minmax problem as a linear programming problem. *Opsearch*, 19:49–53, 1982.

[118] S. K. Gupta and A. Punnen. k-sum optimization problems. *Operations Research Letters*, 9:121–126, 1990.

[119] S. K Gupta and A. Punnen. Minmax linear knapsack problem with grouped variables and gub constraints. *Optimization*, 28:85–94, 1993.

[120] G. Gutin and A. Punnen. *The traveling salesman problem and its variations*, volume 12. Kluwer Academic Pub, 2002.

[121] S. Hadley, F. Rendl, and H. Wolkowicz. Bounds for the quadratic assignment problem using continuous optimization techniques. 1990.

[122] S. W. Hadley. *Continuous optimization approaches for the quadratic assignment problem*. PhD thesis, University of Waterloo, 1989.

[123] S. W. Hadley, F. Rendl, and H. Wolkowicz. A new lower bound via projection for the quadratic assignment problem. *Mathematics of Operations Research*, pages 727–739, 1992.

[124] P. Hahn and T. Grant. Lower bounds for the quadratic assignment problem based upon a dual formulation. *Operations Research*, pages 912–922, 1998.

[125] P. L. Hammer. Plant locationa pseudo-boolean approach. *Israel Journal of Technology*, 6:330–332, 1968.

[126] P. L. Hammer and R. Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Mathematical Methods of Operations Research*, 36:3–21, 1992.

[127] P. L. Hammer and D. J. Rader Jr. Efficient methods for solving quadratic 0-1 knapsack problems. *Infor-Information Systems and Operational Research*, 35:170–182, 1997.

[128] P. L. Hammer and E. Shliffer. Applications of pseudo-boolean methods to economic problems. *Theory and Decision*, 1:296–308, 1971.

[129] P.L. Hammer, S. Rudeanu, and B. Gyires. Boolean methods in operations research and related areas. 1968.

[130] J. P. Hart and A. W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.

[131] C. Helmberg and F. Rendl. Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82:291–315, 1998.

[132] C. Helmberg, F. Rendl, and R. Weismantel. Quadratic knapsack relaxations using cutting planes and semidefinite programming. *Integer Programming and Combinatorial Optimization*, pages 175–189, 1996.

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

[134] M. Hifi and M. Michrafy. A reactive local search-based algorithm for the disjunctively constrained knapsack problem. *Journal of the Operational Research Society*, pages 718–726, 2006.

[135] A. J. W. Hilton. Spanning trees and fibonacci and lucas numbers. *The Fibonacci Quaterly*, 12:259–262, 1974.

[136] J. H. Holland. Adaptation in natural andartificial systems. *Ann Arbor, MI: University of Michigan Press*, 1975.

[137] H. W. Huhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[138] F. K. Hwang, D. S. Richards, and P. Winter. The steiner tree problem, annals of discrete mathematics. *Amsterdam: NorthHolland, The Netherlands*, 1992.

[139] K. Jansen. An approximation scheme for bin packing with conflicts. *Journal of combinatorial optimization*, 3:363–377, 1999.

[140] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *JCSS*, 37:79–100, 1988.

[141] C. Jorgensen and S. Powell. Solving 0-1 minimax problems. *Journal of the Operational Research Society*, pages 515–522, 1987.

[142] M. Jünger, A. Martin, G. Reinelt, and R. Weismantel. Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits. *Mathematical programming*, 63:257–279, 1994.

[143] S. Kabadi. *The Traveling Salesman Problem and Its Variations*, chapter The Polynomially Solvable Cases of the TSP, pages 489–493. Kluwer Academic Publishers, 2002.

[144] S. Kabadi and A. Punnen. *The Traveling Salesman Problem and Its Variations*, chapter The Bottleneck TSP, pages 697–734. Kluwer Academic Publishers, 2002.

[145] S. Kabadi and A. Punnen. Weighted graphs with all hamiltonian cycles of the same length. *Discrete Mathematics*, 271:129–139, 2003.

[146] S. Kabadi and A. Punnen. Qap linearization problem is in p. Technical report, Simon Fraser University, 2011.

[147] V. Kaibel. *Polyhedral combinatorics of the quadratic assignment problem*. PhD thesis, Universität zu Köln, Germany, 1997.

[148] B. Kalantari and A. Bagchi. An algorithm for quadratic zero-one programs. *Naval Research Logistics (NRL)*, 37:527–538, 1990.

[149] S. E. Karisch. *Nonlinear approaches for quadratic assignment and graph partition problems*. PhD thesis, Technical University Graz, Austria, 1995.

[150] S. E. Karisch, E. Cela, J. Clausen, and T. Espersen. A dual framework for lower bounds of the quadratic assignment problem based on linearization. *Computing*, 63:351–403, 1999.

[151] S. E. Karisch and F. Rendl. Lower bounds for the quadratic assignment problem via triangle decompositions. *Mathematical Programming*, 71:137–151, 1995.

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

[153] K. Katayama and H. Narihisa. Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. *European Journal of Operational Research*, 134:103–119, 2001.

[154] F. Kaufman. An algorithm for the quadratic assignment problem using bender's decomposition. *European Journal of Operational Research*, 2:207–211, 1978.

[155] A. Kaveh and A. Punnen. Randomized local search and improved solutions for the microarray qap. Department of Mathematics, Simon Fraser University, 2008.

[156] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer Verlag, 2004.

[157] H. Kellerer and G. Wirsching. Bottleneck quadratic assignment problems and the bandwidth problem. *Asia Pacific Journal of Operational Research*, 15:169–178, 1998.

[158] S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 600–609. IEEE, 2001.

[159] G. Kochenberger, J.K. Hao, Z. Lü, H. Wang, and F. Glover. Solving large scale max cut problems via tabu search. Technical report, 2011.

[160] D. König. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

[161] T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.

[162] J. Krarup and P. M. Pruzan. Computer-aided layout design. *Mathematical programming in use*, pages 75–94, 1978.

[163] J. Krarup and P. M. Pruzan. Reducibility of minimax to minisum 0-1 programming problems. *European Journal of Operations Research*, 6:125–132, 1981.

[164] J. B. Kruskal. On the shortest spannign tree of graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[165] A. M. Land. A problem of assignment with interrelated costs. *Operations Research Quarterly*, 14:185–198, 1963.

[166] G. Laporte and H. Mercure. Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35:378–381, 1988.

[167] D. J. Laughhunn. Quadratic binary programming with application to capital-budgeting problems. *Operations Research*, pages 454–461, 1970.

[168] D. J. Laughhunn and D. E. Peterson. Computational experience with capital expenditure programming models under risk. *J. Business Finance*, 3:43–48, 1971.

[169] E. L. Lawler. The quadratic assignment problem. *Management science*, pages 586–599, 1963.

[170] Y. Li, P. M. Pardalos, K. G. Ramakrishnan, and M. G. C. Resende. Lower bounds for the quadratic assignment problem. *Annals of Operations Research*, 50:387–410, 1994.

[171] S. V. Listrovio and V. I. Khrin. Parallel algorithm to find maximum capacity paths. *Cybernetics and Systems Analysis*, 34:261–268, 1998.

[172] T. L. Magnanti and L. A. Wolsey. Optimal trees. *Handbooks in operations research and management science*, 7:503–615, 1995.

[173] V. Maniezzo and A Colorni. The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering, IEEE Transactions*, 11:769–778, 1999.

[174] S. Martello, W. Pulleyblank, P. Toth, and D. de. Werra. Balanced optimization problems. *Operations Research Letters*, 3:275–278, 1984.

[175] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[176] S. Martello and P. Toth. The bottleneck generalized assignment problem. *European Journal of Operational Research*, 83:621–638, 1995.

[177] R. Martí, V. Campos, and E. Piñana. A branch and bound algorithm for the matrix bandwidth minimization. *European Journal of Operational Research*, 186:513–528, 2008.

[178] A. Marzetta and A. Brüngger. A dynamic-programming bound for the quadratic assignment problem. *Computing and Combinatorics*, pages 339–348, 1999.

[179] T. Mautor and C. Roucairol. A new exact algorithm for the solution of quadratic assignment problems. *Discrete Applied Mathematics*, 55:281–293, 1994.

[180] R. D. McBride and J. S. Yormark. An implicit enumeration algorithm for quadratic integer programming. *Management Science*, pages 282–296, 1980.

[181] P. Merz and B. Freisleben. Genetic algorithms for binary quadratic programming. 1:417–424, 1999.

[182] P. Michelon and L. Veilleux. Lagrangean methods for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 95:671–682, 1996.

[183] P. B. Mirchandani and T. Obata. *Locational Decisions with Interactions Between Facilities: The Quadratic Assignment Problem: a Review*. Laboratory for Computer Methods in Public Systems, 1979.

[184] J. W. Moon. Counting labelled trees. Canadian Mathematical Congress, 1970.

[185] H. Müller-Merbach. *Optimale Reihenfolgen*. Springer-Verlag, 1970.

[186] S. C. Narula and C. A. Ho. Degree-constrained minimum spanning tree. *Computer and Operation Reserch*, 7:239–249, 1980.

[187] C. E. Nugent, T. E. Vollmann, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, pages 150–173, 1968.

[188] M. W. Padberg and M. P. Rijal. *Location, scheduling, design, and integer programming*, volume 3. Springer, 1996.

[189] P. M. Pardalos and J. V. Crouse. A parallel algorithm for the quadratic assignment problem. In *Proceedings of the 1989 ACM/IEEE conference on Supercomputing*, pages 351–360. ACM, 1989.

[190] P. M. Pardalos and G. C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. In *Quadratic assignment and related problems: DIMACS Workshop, May 20-21, 1993*, volume 16, page 237. American Mathematical Society, 1994.

[191] P. M. Pardalos and G. P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45:131–144, 1990.

[192] P. M. Pardalos and G. P. Rodgers. A branch and bound algorithm for the maximum clique problem. *Computers & operations research*, 19:363–375, 1992.

[193] PM Pardalos, KG Ramakrishnan, MGC Resende, and Y. Li. Implementation of a variance reduction-based lower bound in a branch-and-bound algorithm for the quadratic assignment problem. *SIAM Journal on Optimization*, 7:280–294, 1997.

[194] C. C. Petersen. A capital budgeting heuristic algorithm using exchange operations. *AIIE Transactions*, 6:143–150, 1974.

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

[196] U. Pferschy and J. Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13:233–249, 2009.

[197] A. T. Phillips and J. B. Rosen. A quadratic assignment formulation of the molecular conformation problem. *Journal of Global Optimization*, 4:229–241, 1994.

[198] J. C. Picard and H. D. Ratliff. Minimum cuts and related problems. *Networks*, 5:357–370, 1975.

[199] J. C. Picard and H. D. Ratliff. A cut approach to the rectilinear distance facility location problem. *Operations Research*, pages 422–433, 1978.

[200] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[201] A. Punnen. On combined minmax-minsum optimization. *Computers and Operations Research*, 21:707–716, 1994.

[202] A. Punnen. A fast algorithm for a class of bottleneck problems. *Computing*, 56:397–401, 1996.

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

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

[205] A. Punnen and K. P. K. Nair. An $o(m \log n)$ algorithm for the max+sum spanning tree problem. *European Journal of Operational Research*, 89:423–426, 1996.

[206] A. Punnen and Aneja Y. P. On k-sum optimization. *Operations Research Letters*, 18:233–236, 1996.

[207] A. Punnen and R. Zhang. Quadratic bottleneck problems. *Naval Research Logistics*, 58:153–164, 2011.

[208] M. Queyranne. Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Operations Research Letters*, 4:231–234, 1986.

[209] D. Rader Jr. and W. Woeginger. The quadratic 0-1 knapsack problem with series-parallel support. *Operations Research Letters*, 30:159–166, 2002.

[210] F. Rendl. Ranking scalar products to improve bounds for the quadratic assignment problem. *European journal of operational research*, 20:363–372, 1985.

[211] F. Rendl and H. Wolkowicz. Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem. *Mathematical Programming*, 53:63–78, 1992.

[212] M. Resende and C. Ribeiro. Greedy randomized adaptive search procedures. *Handbook of metaheuristics*, pages 219–249, 2003.

[213] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the Association of Computing Machinery*, 23:555–565, 1976.

[214] J. Sedlacek. On the number of spanning trees of finite graphs. *Casopis pro pestovani matematiky*, 95:217–221, 1969.

[215] N. Z. Shor, K. C. Kiwiel, and A. Ruszcaynski. *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc., 1985.

[216] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on computing*, 2:33–45, 1990.

[217] S. M. Soak, D. W. Corne, and B. H. Ahn. A new evolutionary algorithm for spanning tree based communication network design. *IEICE Transaction on Communication*, E88-B:4090–4093, 2005.

[218] S. M. Soak, D. W. Corne, and B. H. Ahn. The edge-window-decoder representation for tree-based problems. *IEEE transactions on Evolutionaly Computation*, 10:124–144, 2006.

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

[220] W. Szpankowski. Combinatorial optimization problems for which almost every algorithm is asymptotically optimal. *Optimization*, 33:359–367, 1995.

[221] *D*.D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673, 1993.

[222] E. Taillard. Robust tabu search for the quadratic assignment problem. *Parallel computing*, 17:443–455, 1991.

[223] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of the ACM*, 29:623–641, 1982.

[224] R. E. Tarjan. A simple version of karzanov's blocking flow algorithm. *Operations Research Letters*, 2:265–268, 1984.

[225] D. M. Tate and A. E. Smith. A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, 22:73–83, 1995.

[226] I. Ugi, J. Bauer, J. Brandt, J. Friedrich, J. Gasteiger, C. Jochum, and W. Schubert. Neue anwendungsgebiete für computer in der chemie. *Angewandte Chemie*, 91:99–111, 1979.

[227] V. Vassilevska, R. Williams, and R. Yuster. All-pairs bottleneck paths for general graphs in truly sub-cubic time. *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 585–589, 2007.

[228] H. Wang, G. Kochenberger, and Y. Xu. A note on optimal solutions to quadratic knapsack problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1:344–351, 2010.

[229] A. Warszawski. Pseudo-boolean solutions to multidimensional location problems. *Operations Research*, pages 1081–1085, 1974.

[230] H. M. Weingartner. Capital budgeting of interrelated projects: survey and synthesis. *Management Science*, pages 485–516, 1966.

[231] D. J. A. Welsh. *Matroid theory*, volume 8. Academic Press, 1976.

[232] M. R. Wilhelm and T. L. Ward. Solving quadratic assignment problems by simulated annealing. *IIE transactions*, 19:107–119, 1987.

[233] H. C. Wirth and J. Steffan. On minimum diameter spanning trees under reload costs. In *Graph-Theoretic Concepts in Computer Science*, pages 78–89. Springer, 1999.

[234] T. Yamada, S. Kataoka, and K. Watanabe. Heuristic and exact algorithms for disjunctively constrained knapsack problem. *IPSJ Journal*, 43:2864–2870, 2002.

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

[236] R. Zhang and A. Punnen. Quadratic bottleneck knapsack problems. *Journal of Heuristics*, 2011.

[237] Q. Zhao. *Semidefinite programming for assignment and partitioning problems*. PhD thesis, University of Waterloo, Canada, 1996.

[238] Q. Zhao, S. E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2:71–109, 1998.

[239] G. Zhou and M. Gen. An effective genetic algorithm approach to the quadratic minimum spanning tree problem. *Computers & Operations Research*, 25:229–237, 1998.

[240] G. Zhou and M. Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114:141–152, 1999.

Vita

NAME OF CANDIDATE: Ruonan Zhang

DATE AND PLACE OF BIRTH: January 13, 1981, Jiayuguan, China

DEGREES AWARDED:

Bachelor of Science in Mathematics, Lanzhou University, 2003
Master of Science in Mathematics, University of New Brunswick, 2005

PROFESSIONAL EXPERIENCE:

- Project Advisor/Analyst, Center for Operations Excellence, Sauder School of Business, University of British Columbia, 2011
- Sessional Instructor, Department of Mathematics, Simon Fraser University, 2009
- Research/Teaching Assistant, Department of Mathematics, Simon Fraser University, 2005-2011
- Research/Teaching Assistant, Department of Mathematics, University of New Brunswick, 2003-2005

PROFESSIONAL PUBLICATIONS

[1] A. Punnen and R. Zhang, Bottleneck flows in unit capacity networks, *Information Processing Letters*, 109(2009), 334-338.

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

[3] A. Punnen and R. Zhang, Quadratic bottleneck combinatorial optimization problems, *Naval Research Logistics*, 58(2011), 153-164.

[4] R. Zhang and A. Punnen, Quadratic bottleneck knapsack problems, *Journal of Heuristics* (2011), DOI: 10.1007/s10732-011-9175-1

[5] A. Punnen and R. Zhang, Analysis of an approximate greedy algorithm for the maximum edge clique partitioning problem, (2011), submitted to *Discrete Optimization*.

[6] T. Öncan, R. Zhang and A. Punnen, Assignment problem with conflict pair constraints, under preparation.

[7] R. Zhang, T. Öncan and A. Punnen, Quadratic bottleneck assignment problem and its applications, under preparation.

[8] R. Zhang and A. Punnen, Quadratic minimum spanning trees and its variations, under preparation.