THE QUADRATIC MINIMAL SPANNING TREE PROBLEM AND RELATED TOPICS

by
Weixuan Xu

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1984

7

APPROVAL SHEET

Title of Thesis: The Quadratic Minimal Spanning Tree Problem and Related Topics

Name of Candidate: Weixuan Xu

Doctor of Philosophy, 1984

Thesis and Abstract Approved: [illegible]

Arjang A. Assad

Associate Professor

College of Business & Management

Date Approved: 11/20/1984

ABSTRACT

Title of Dissertation : Quadratic Minimal Spanning Tree Problem and Related Topics

Weixuan Xu , Doctor of Philosophy , 1984

Dissertation directed by : Arjang A. Assad
Associate Professor
College of Business & Management

This dissertation investigates a new network optimization problem, obtained by defining a quadratic cost structure on trees in networks. For this reason, we refer to it as the *Quadratic Minimal Spanning Tree Problem.* Very little previous work on this problem exists in literature. In this dissertation, we prove that this problem is NP-hard hence we must use branch-and-bound algorithms to solve it to optimalty or resort to heuristic algorithms to solve it approximately. A number of bounding techniques based on linear programming, Lagrangean relaxation, decomposition arguments, and a parametrization bounding technique are developed and compared. The parametrization bounding technique leads to an interesting optimization problem: Find the m-dimensional vector of parameters that maximizes a piecewise linear concave function. The maximum provides a good lower bound for the Quadratic Minimal Spanning Tree Problem. A new method called the *Leveling Method* is proposed in lieu of the generally used (but cumbersome) subgradient search technique. Computational experiments show that the leveling method is very efficient and reliable for solving the above parameter search problem.

The leveling method is then generalized to a broader class of *Quadratic 0-1 programs.* Several theorems ensure its convergence to the best parameters. The method is applied to a famous combinatorial optimization problem -- the *Qua-*

dratic Assignment Problem. For a number of test problems of Koopmans-Beckmann type (a special case of the Quadratic Assignment Problem), the leveling method results in better lower bounds than the existing bounding techniques proposed by Gilmore[1962], Lawler[1962], Christofides et al[1980] and Frieze and Yadegar[1983], either in quality of the bounds or in computational efficiency. Our computational esperience shows that, on the average, the leveling method obtains lower bounds 40% larger than those obtained by Gilmore and Lawler's technique.

Based on different bounding techniques, three branch-and-bound algorithms for solving the Quadratic Minimal Spanning Tree Problems are proposed and compared. One of them focuses on a special but more applicable case of the Quadratic Spanning Tree Problem, called the *Adjacent-only Quadratic Minimal Spanning Tree Problem.* These exact algorithms are followed by three heuristic algorithms that search for near-optimal solutions. Computational experience shows that one of these heuristics, called the *Sequential Fixing Method,* obtains near-optimal solutions in an efficient way.

THE QUADRATIC MINIMAL SPANNING TREE PROBLEM AND RELATED TOPICS

by
Welxuan Xu

Maryland
LD
3231
M70d
Xu,
W.
Folio

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1984

copy 1

7

APPROVAL SHEET

Title of Thesis: The Quadratic Minimal Spanning Tree Problem and Related Topics

Name of Candidate: Weixuan Xu

Doctor of Philosophy, 1984

Thesis and Abstract Approved: [illegible]

Arjang A. Assad

Associate Professor

College of Business & Management

Date Approved: 11/20/1984

ABSTRACT

Title of Dissertation : Quadratic Minimal Spanning Tree Problem and Related Topics

Weixuan Xu , Doctor of Philosophy , 1984

Dissertation directed by : Arjang A. Assad
Associate Professor
College of Business & Management

This dissertation investigates a new network optimization problem, obtained by defining a quadratic cost structure on trees in networks. For this reason, we refer to it as the *Quadratic Minimal Spanning Tree Problem.* Very little previous work on this problem exists in literature. In this dissertation, we prove that this problem is NP-hard hence we must use branch-and-bound algorithms to solve it to optimalty or resort to heuristic algorithms to solve it approximately. A number of bounding techniques based on linear programming, Lagrangean relaxation, decomposition arguments, and a parametrization bounding technique are developed and compared. The parametrization bounding technique leads to an interesting optimization problem: Find the m-dimensional vector of parameters that maximizes a piecewise linear concave function. The maximum provides a good lower bound for the Quadratic Minimal Spanning Tree Problem. A new method called the *Leveling Method* is proposed in lieu of the generally used (but cumbersome) subgradient search technique. Computational experiments show that the leveling method is very efficient and reliable for solving the above parameter search problem.

The leveling method is then generalized to a broader class of *Quadratic 0-1 programs.* Several theorems ensure its convergence to the best parameters. The method is applied to a famous combinatorial optimization problem -- the *Qua-*

dratic Assignment Problem. For a number of test problems of Koopmans-Beckmann type (a special case of the Quadratic Assignment Problem), the leveling method results in better lower bounds than the existing bounding techniques proposed by Gilmore[1962], Lawler[1962], Christofides et al[1980] and Frieze and Yadegar[1983], either in quality of the bounds or in computational efficiency. Our computational esperience shows that, on the average, the leveling method obtains lower bounds 40% larger than those obtained by Gilmore and Lawler's technique.

Based on different bounding techniques, three branch-and-bound algorithms for solving the Quadratic Minimal Spanning Tree Problems are proposed and compared. One of them focuses on a special but more applicable case of the Quadratic Spanning Tree Problem, called the *Adjacent-only Quadratic Minimal Spanning Tree Problem.* These exact algorithms are followed by three heuristic algorithms that search for near-optimal solutions. Computational experience shows that one of these heuristics, called the *Sequential Fixing Method,* obtains near-optimal solutions in an efficient way.

To

my parents

and

my wife Jiehu

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my advisor, Professor Arjang A. Assad. His perceptive advice, his insistence on rigor and precision, his continuous prodding, and his patience in correcting my drafts were of great help to me in my research and the preparation of this dissertation.

I am grateful to Professor Bruce Golden for bringing the Stochastic Spanning Tree Problem to our attention. This problem serves as the starting point for this work.

I would like to take this opportunity to thank Professors Grace Yang, Hsin Chu, William Widhelm and Denny Gulick, as well as Mr. Thomas Tsao and Mr. Rick Vohra for their concern and help during my graduate study and the preparation of this thesis.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENT .... ii a

Chapter Page

1. INTRODUCTION 1

2. FORMULATION AND COMPLEXITY 8

2.1. Formulation of the Quadratic Minimal Spanning Tree Problem (QMST) 8

2.2. Motivation of our Study 11

2.3. NP-Completeness of the QMST 13

2.4. NP-Completeness of the AQMST 18

3. BOUNDING TECHNIQUES FOR THE QMST 21

3.1. Linear Programming Approach 21

3.2. Lagrangean Relaxation Approach 24

3.3. The Decomposition Bound (*DB*) 25

3.4. The Parametrized Bound (*PB*) 26

3.5. The Leveling Method for Searching the Best *PB* 31

3.6. Computational Experiments 33

4. A BOUNDING TECHNIQUE FOR A CLASS OF QUADRATIC 0-1 PROGRAMS 44

4.1. Brief Introduction of the Problem 44

4.2. The Parametrized Lower Bound 45

4.3. A Sufficient Condition for the Best Parameters 49

4.4. Leveling Algorithm and Its Convergence 51

5. ON THE LOWER BOUND OF THE QUADRATIC ASSIGNMENT PROBLEM (QAP) 56

5.1. Existing Bounding Techniques 56

5.2. Application of the Leveling Method to the QAP 64

5.3. Comparisons 68

6. SOLUTION TECHNIQUES FOR THE QMST 74

6.1. General Description 74

6.2. Branch-and-Bound Algorithm I (with decomposition bounds) 76

6.3. Branch-and-Bound Algorithm II (with parametrized bounds) 78

6.4. Branch-and-Bound Algorithm III (for the Adjacent-only QMST) 83

6.5. Heuristic Algorithms 86

6.6. Computational Experiments 91

7. CONCLUSIONS 95

APPENDIX A. Network Terminology 99

APPENDIX B. Efficient Algorithms for Solving the MST 101

APPENDIX C. Formulations of the QAP and the KBP 103

APPENDIX D. Test Problems for Section 5.3 105

BIBLIOGRAPHY 112

CHAPTER 1
INTRODUCTION

This dissertation involves a new network optimization problem, obtained by defining a quadratic cost structure on trees in networks. For this reason, we refer to it as the *Quadratic Minimal Spanning Tree Problem* (QMST). Very little previous work on this problem exists in literature. In this dissertation we develop bounding techniques, exact algorithms, and heuristic algorithms to solve the Quadratic Minimal Spanning Tree Problem optimally or near-optimally. Some methods designed for the QMST are generalized to a broader class of Quadratic 0-1 Programming problems. This allows us apply these methods to a famous combinatorial problem -- the Quadratic Assignment Problem (QAP), for which an extensive and growing literature exists.

A *spanning tree* in a connected undirected network is a connected subgraph without cycles which spans all vertices of the network. Examples of spanning trees are shown in Figures 1-2 and 1-3 for the network of Figure 1-1.

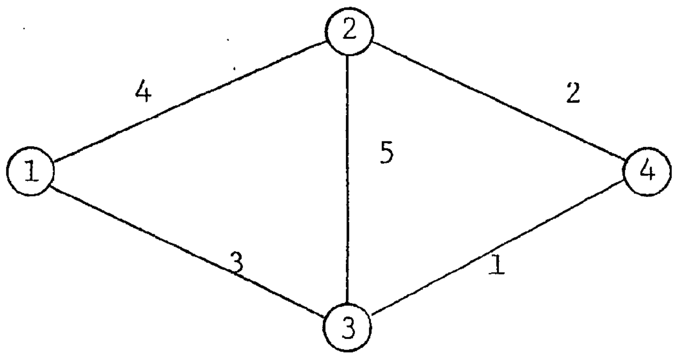

Figure 1-1. A network with costs assigned to edges.

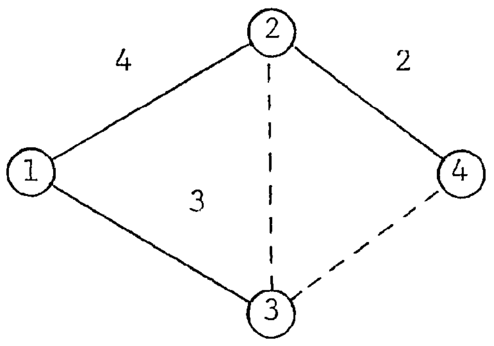

Figure 1-2. A spanning tree with total cost of 9.

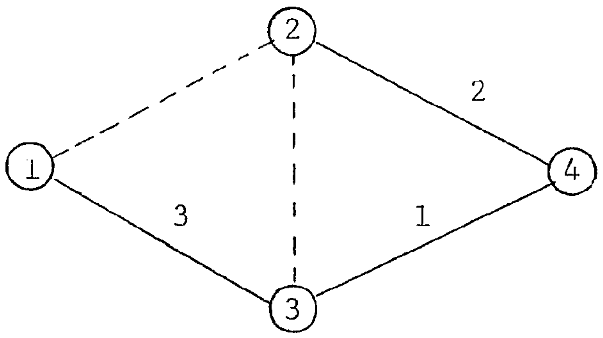

Figure 1-3. A minimal spanning tree with total cost of 6.

Given a connected undirected network with costs assigned to the edges, the *Minimal Spanning Tree Problem* (MST) is to find a spanning tree with minimal total cost among all spanning trees in the network. Minimal spanning trees are useful in a wide range of applications and in network theory. A simple example is to connect a number of locations using a pipeline of minimal total length. Other applications concern the design of television cable networks and telephone networks, clustering, pattern recognition, automatic control, network reliability and physical chemistry (see Bradley[1975] and Haymond et al[1980] for rich references). There are a number of variants of the preceding problem which involve different cost functions or other type of constraints (e.g. see Camerini et al[1980] and Chandy and Lo[1974]), all of which involve the construction of spanning trees. There are highly efficient implementations of algorithms for constructing minimal spanning trees. They draw upon the two algorithms by Kruskal[1956] and Prim[1957]. These algorithms are briefly described in Appendix B. The spanning tree is a network fundamental structure. Therefore, the Minimal Spanning Tree Problem and its variants are frequently used as important subproblems in solving other difficult network problems such as the network design problem (Gomory and Hu[1961]), the traveling salesman problem (Held and Karp[1970, 1971]), the vehicle routing problem (Christofides et al[1980]), and the multiple traveling salesman problem (Gavish and Srikanth[1983]).

Now suppose that the costs attached to a network depend not only on a single edge but also on each pair of edges. For instance, when transmitting from one pipe to another in a pipeline network, additional efforts may be required. When connecting a cable above the earth with an underearth cable, special apdapters may be needed. When traveling from one road to another in a transportation network, turn penalties might have to be considered in addition to the regular transportation expenses. In these cases, if one wants to find a spanning tree with minimal total cost, then an optimization problem with a quadratic objective function and the same constraints with MST will emerge. We call it the *Quadratic Minimal Spanning Tree Problem* (QMST). As an example, suppose that additional interactive costs are imposed on the network shown in Figure 1-1:

	(1-2)	(1-3)	(2-3)	(2-4)	(3-4)
(1-2)	4	0	1	3	2
(1-3)	0	3	4	8	7
(2-3)	1	4	5	6	9
(2-4)	3	8	6	2	3
(3-4)	2	7	9	3	1

e.g., the intercost between edge (2-3) and edge (3-4) is 9. The spanning tree in Figure 1-3 has a total cost of 42 while that in Figure 1-2 has a total cost of 31. This quadratic problem arises naturally when one considers a stochastic version of the minimal spanning tree problem, called by Ishii et al[1981a,1981b] the *Stochastic Spanning Tree Problem,* for which the edge costs are taken to be correlated random variables. Using a chance-constrained formulation of the stochastic problem, one may transform the problem into a Minimal Spanning Tree problem.

As mentioned earlier, certain results of this dissertation for the QMST may be generalized to a class of Quadratic 0-1 Programming problems. A general quadratic 0-1 programming problem has a quadratic objective function and a set of linear or quadratic constraints. Usually, it is an NP-hard problem. In order to solve it, one must resort to implicit enumeration methods. A highly important component of an enumeration method, from the point of view of its influence on overall efficiency, is the lower bounding procedure. For Quadratic 0-1 Programming problems, one of the lower bounds on the objective, given by Hansen([1975] and [1979]), is the sum of its negative coefficients. Based on this rough bound, some improvements can be performed by, for instance, considering the increments which may be added to a bound when a free variable is fixed at 0 or 1. In this dissertation, we try to study the bounding technique for a subclass of the Quadratic 0-1 Programming problems.

A famous combinatorial problem that we show to be a special case of the QMST problem is the Quadratic Assignment Problem (QAP). To motivate the QAP, consider the problem studied by Koopmans and Beckmann[1957]. Suppose that n plants are to be assigned to n locations so that each plant receives a unique location, and vice versa. Let f_{ij} be the amount of goods to be transported from plant i to plant j and d_{pq} the distance between locations p and q . Assume that the cost of transporting goods is linear in the amount of goods moved as well as in the distance over which these goods are transported. Then, if plants i and j are assigned to locations p and q respectively, the cost for transporting goods from plant i to plant j is $f_{ij}\, d_{pq}$. The problem is to find an assignment so that the total cost of transporting all goods between all plants is minimum. As Lawler[1962] and others have pointed out, this problem is a special case of the class of Quadratic Assignment Problems (QAP's) and is referred to as a Koopmans-Beckmann Problem (KBP). Quadratic Assignment Problems, especially Koopmans-Beckmann Problems, have many interesting applications in the area of layout and facility design. For example, the hospital layout problem (Elshafei[1977]), the backboard wiring problem (Steinboard[1961]), the campus planning problem (Dickey and Hopkins[1972]), among others, can all be formulated as Quadratic Assignment Problems. The QAP is an NP-complete problem and, in fact, has a reputation for being one of the more computationally intractable NP-hard problems. This is evident from the fact that even for a Koopmans-Beckmann Problem which is simpler than the general QAP, the size of problems that can be solved to optimality within a reasonable time is limited to 15 plants (n=15). Due to its wide variety of applications and its computational difficulty, the QAP has attracted the attention of many researchers. Surveys of approaches and applications to the QAP can be found in Gilmore[1962], Lawler[1962], Nugent et al[1968], Burkard and Stratmann[1978], Bazaraa and Sherali[1981] and Burkard[1984], among others. A contribution of the research in this dissertation is to specialize a new bounding technique for quadratic 0-1 programs to the Quadratic Assignment Problem. A comparison of this new technique with existing bounding procedure is also provided.

Plan of this thesis:

Chapter 2 starts with two formulations of the Quadratic Minimal Spanning Tree Problem. This problem is then proven to be NP-hard by showing that the Quadratic Assignment Problem and the famous Hamiltonian Path Problem are special cases of the Quadratic Minimal Spanning Tree Problem.

Chapter 3 concentrates on bounding techniques for the Quadratic Minimal Spanning Tree Problem, including several approaches for obtaining lower bounds, based on linear programming, Lagrangean relaxation, decomposition arguments, and a parametrization bounding technique are developed and compared in this chapter. The parametrization bounding technique leads to an interesting problem: Find the m-dimensional vector of parameters that maximizes a piecewise linear, concave function. This maximum provides a lower bound for the Quadratic Minimal Spanning Tree Problem. A new method called the *Leveling Method* is proposed in lieu of the generally-used (but cumbersome) subgradient search technique. Our computational experiments show that the leveling method is an efficient method for solving the preceding problem of finding a "best" set of parameters.

In Chapter 4 the leveling method is generalized to a broader class of Quadratic 0-1 Programming problems. It is proved theoretically that the leveling algorithm yields an improvement in the parametrized lower bound after each iteration and converges to a solution.

Chapter 5 opens with a brief review of some existing bounding techniques for the Quadratic Assignment Problem. We then apply the leveling method to a number of Koopmans-Beckmann Problems and general Quadratic Assignment Problems and, in a computational study, compare the resulting lower bounds with those obtained by Gilmore[1963], Lawler[1963], Edwards[1980], Christofides et al[1980], Frieze and Yadegar[1983].

Chapter 6 describes three branch-and-bound algorithms developed for solving the Quadratic Minimal Spanning Tree Problems to optimality. One of them focuses on a special but more applicable case of the Quadratic Minimal Spanning Tree Problem -- the *Adjacent-only Quadratic Minimal Spanning Tree Problem.* These exact algorithms are followed by three heuristic algorithms that search for near-optimal solutions. Computational results are reported and the above algorithms are compared.

Our conclusions appear in Chapter 7, together with some suggestions on directions for future research.

CHAPTER 2

FORMULATION AND COMPLEXITY

In this chapter, the Quadratic Spanning Tree Problem and one of its special cases, the Adjacent-only Quadratic Minimal Spanning Tree Problem (AQMST), are formulated mathematically. We then relate the QMST to the Stochastic Minimal Spanning Tree Problem. We further demonstrate that the QMST and the adjacent-only QMST are both NP-complete problems.

2.1. Formulation of the Quadratic Minimal Spanning Tree Problem (QMST)

Let $G = (V,E)$ be a connected undirected graph. The edges in E are labeled as $E = \{ e_1 , e_2 , . . . , e_m \}$ and the set of vertices is denoted by $V = \{ v_1 , v_2 , . . . , v_n \}$. A subgraph in G is represented by an m-vector $\mathbf{x} = (x_1, x_2, . . . , x_m)$ where $x_i = 1$ if edge e_i belongs to the subgraph and $x_i = 0$ otherwise. Let $T(G)$ denote the collection of all spanning trees in G. " $\mathbf{x} \in T(G)$ " means that the subgraph formed by edges for which the corresponding x_i equals one is a spanning tree in G. Since any spanning tree has $n-1$ edges, we must have

$$\sum_{i=1}^{m} x_i = n-1 \tag{2.1}$$

if $\mathbf{x} \in T(G)$.

Assume that there is a number b_i, called the edge cost, associated with each edge $e_i \in E$. The Minimal Spanning Tree Problem (MST) can be formulated as

(MST) Minimize $\sum_{i=1}^{m} b_i x_i$ (2.2)

subject to : $\mathbf{x} \in T(G)$ (2.3)

Assume further that there is an number a_{ij}, called the intercost between edge e_i and edge e_j, associated with each pair of edges (e_i, e_j). If one wants to find a spanning tree over G which has the minimal total cost among all spanning trees, the following problem arises:

(QMST) Minimize $z(\mathbf{x}) = \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m} b_i x_i$ (2.4)

subject to : $\mathbf{x} \in T(G)$ (2.3)

We refer to this problem as the *Quadratic Minimal Spanning Tree Problem* (QMST).

Though the intercosts a_{ij} assume any value, The following lemma allows us to assume without loss of generality that all a_{ij} in (2.4) are symmetric in i and j and nonnegative.

Lemma 2.1. Let P and Q be any constants. Under the transformation

$$\bar{b}_i = b_i + \mathrm{P}, \qquad 1 \leq i, j \leq m,$$
$$\bar{a}_{ij} = (a_{ij} + a_{ji})/2 + \mathrm{Q}, \quad 1 \leq i, j \leq m, \quad i \neq j,$$

the new QMST

Minimize $\bar{z}(\mathbf{x}) = \sum_{i=1}^{m} \bar{b}_i x_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} \bar{a}_{ij} x_i x_j$

subject to : $\mathbf{x} \in T(G)$

has the same solution tree as that of (2.4).

Proof. Let $\mathbf{x} \in T(G)$ be any spanning tree. The objective value of $\mathbf{x}$ in the new QMST is

$$\bar{z}(\mathbf{x}) = \sum_{i=1}^{m} b_i x_i + \frac{1}{2} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij} x_i x_j + \frac{1}{2} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ji} x_i x_j$$

$$+ \mathrm{P} \sum_{i=1}^{m} x_i + \mathrm{Q} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} x_i x_j .$$

The second and third summations are equal. Due to (2.1) and $x_i^2 = x_i$, the last summation is equal to

$$\mathrm{Q} \sum_{i=1}^{m} x_i (n-1-x_i) = (n-1)\mathrm{Q} \sum_{i=1}^{m} x_i - \mathrm{Q} \sum_{i=1}^{m} x_i^2$$

$$= (n-1)(n-2)\mathrm{Q} .$$

Thus

$$\bar{z}(\mathbf{x}) = z(\mathbf{x}) + (n-1)\mathrm{P} + (n-2)(n-1)\mathrm{Q} .$$

Since $\bar{z}(\mathbf{x})$ differs from $z(\mathbf{x})$ by only a constant, the proof is complete.

In the QMST, some a_{ij}'s may vanish. Examples are those we mentioned in Chapter 1 where the intercosts are turn penalties in a transportation network or cable networks. The interactive effect between two nonadjacent edges can be neglected while those of adjacent edges must be taken into consideration. To find the spanning tree with minimum total cost, we must solve a special case of the QMST in which $a_{ij} = 0$ if e_i and e_j are nonadjacent. We call this special case the *Adjacent-only Quadratic Minimal Spanning Tree Problem* (AQMST).

2.2. Motivation of our Study

Our examination of the QMST was initially motivated by investigating a probabilistic version of the Minimal Spanning Tree Problem.

The ordinary MST has studied in detail only for deterministic networks in which the edge costs or characteristics are all known. However, one could model edge costs as random variables rather than constants to allow for fluctuations in such costs. Suppose that the edge costs

$$\mathbf{c} = (c_1, c_2, \ldots, c_m)^t$$

have a multivariate normal distribution:

$$\mathbf{c} \sim N(\mu, \Sigma)$$

where

$$\mu = (\mu_1, \mu_2, \ldots, \mu_m)^t$$

is the mean vector and

$$\Sigma = E(\mathbf{c} - \mu)(\mathbf{c} - \mu)^t = \left[\sigma_{ij}\right]$$

is the variance-covariance matrix.

Consider the following probabilistic optimization problem:

(SMST) Minimize z

$$\text{subject to: } Pr\left\{\sum_{i=1}^{m} c_i x_i \leq z\right\} \geq \alpha \tag{2.5}$$

$$\mathbf{x} \in T(G)$$

Call it the Stochastic Minimal Spanning Tree Problem (SMST). Ishii et al[1981a, 1981b] studied a special case when the edge costs are mutually independent. They gave a good algorithm to find the optimal solution. However, when the edge costs are correlated, the problem is far more complicated.

If all x_i's are continuous rather than 0-1 variables, this is a classical Chance-constrained Stochastic Programming which has a deterministic equivalent (see Charnes and Cooper[1963] and Koolbin[1971]). The deterministic equivalent of the SMST can be formed in the similar way as follows.

Let $Y = \sum_{i=1}^{m} c_i x_i$. Then Y is a normally distributed random variable with mean $\mu^t \mathbf{x}$ and variance $\mathbf{x}^t \Sigma \mathbf{x}$:

$$Y \sim N(\mu^t \mathbf{x}, \mathbf{x}^t \Sigma \mathbf{x}) .$$

From (2.5), we conclude that

$$Pr\left\{ Y \le z \right\} = Pr\left\{ \frac{Y - \mu^t \mathbf{x}}{(\mathbf{x}^t \Sigma \mathbf{x})^{1/2}} \le \frac{z - \mu^t \mathbf{x}}{(\mathbf{x}^t \Sigma \mathbf{x})^{1/2}} \right\}$$

$$= \Phi\left(\frac{z - \mu^t \mathbf{x}}{(\mathbf{x}^t \Sigma \mathbf{x})^{1/2}} \right),$$

where

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\xi^2/2} d\xi \tag{2.6}$$

Thus, the constraint (2.5) is equivalent to

$$\frac{z - \mu^t \mathbf{x}}{(\mathbf{x}^t \Sigma \mathbf{x})^{1/2}} \ge q_\alpha$$

or

$$z \ge \mu^t \mathbf{x} + q_\alpha (\mathbf{x}^t \Sigma \mathbf{x})^{1/2} , \tag{2.7}$$

where q_α is the α-quantile of the standard normal distribution, i.e., $\Phi(q_\alpha) = \alpha$. We assume that $1/2 < \alpha \le 1$ so that $q_\alpha > 0$.

Replacing (2.5) by (2.7), we can see that the SMST is equivalent to the following deterministic problem:

$$\text{Minimize} \quad z(\mathbf{x}) = \sum_{i=1}^{m} \mu_i x_i + q_\alpha \left(\sum_{i=1}^{m} \sum_{j=1}^{m} \sigma_{ij} x_i x_j \right)^{1/2} \tag{2.8}$$

subject to : $\mathbf{x} \in T(G)$

Since $q_\alpha > 0$, it can be absorbed into σ_{ij} by letting $a_{ij} = q_\alpha^2 \sigma_{ij}$. Then problem (2.8) takes the form

$$\text{Minimize} \quad z(\mathbf{x}) = \sum_{i=1}^{m} \mu_i x_i + \left(\sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j \right)^{1/2} \tag{2.9}$$

subject to $\mathbf{x} \in T(G)$.

This problem is complicated by the presence of the quadratic terms. When all μ_i's are equal, it reduces to the Quadratic Minimal Spanning Tree Problem defined in (2.4). Therefore, the QMST is a special case of the Stochastic Spanning Tree Problem.

On the other hand, a good understanding of the Quadratic Minimal Spanning Tree Problem will certainly benefit us in solving the Stochastic Spanning Tree Problem. For example, if there is a good bounding technique for the QMST , then it can be used to obtain a good lower bound for the Stochastic Spanning Tree Problem.

We therefore focus on the quadratic term in (2.9) and study the Quadratic Minimal Spanning Tree Problem in this dissertation.

2.3. NP-Completeness of the QMST

We will prove that the Quadratic Minimal Spanning Tree Problem is NP-hard by comparing it with the Quadratic Assignment Problem (QAP).

The formulations of the QAP and its special case, the Koopmans-Beckmann Problem (KBP), can be found in Appendix C. For convenience, we subject to : $\mathbf{x} \in T(G)$

Since $q_\alpha > 0$, it can be absorbed into σ_{ij} by letting $a_{ij} = q_\alpha^2 \sigma_{ij}$. Then problem (2.8) takes the form

$$\text{Minimize} \quad z(\mathbf{x}) = \sum_{i=1}^{m} \mu_i x_i + \left(\sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j \right)^{1/2} \tag{2.9}$$

subject to $\mathbf{x} \in T(G)$.

This problem is complicated by the presence of the quadratic terms. When all μ_i's are equal, it reduces to the Quadratic Minimal Spanning Tree Problem defined in (2.4). Therefore, the QMST is a special case of the Stochastic Spanning Tree Problem.

On the other hand, a good understanding of the Quadratic Minimal Spanning Tree Problem will certainly benefit us in solving the Stochastic Spanning Tree Problem. For example, if there is a good bounding technique for the QMST , then it can be used to obtain a good lower bound for the Stochastic Spanning Tree Problem.

We therefore focus on the quadratic term in (2.9) and study the Quadratic Minimal Spanning Tree Problem in this dissertation.

2.3. NP-Completeness of the QMST

We will prove that the Quadratic Minimal Spanning Tree Problem is NP-hard by comparing it with the Quadratic Assignment Problem (QAP).

The formulations of the QAP and its special case, the Koopmans-Beckmann Problem (KBP), can be found in Appendix C. For convenience, we restate briefly the QAP below with a slight modification in order to serve our purpose in this section.

Consider n machines labeled by $1, 2, \ldots, n$ that must be assigned to n locations labeled by $n+1, n+2, \ldots, 2n$. The Quadratic Assignment Problem is formulated as

$$\text{(QAP)} \quad \text{Minimize} \quad w(\mathrm{y}) = \sum_{i=1}^{m} \sum_{p=n+1}^{2n} \sum_{j=1}^{n} \sum_{q=n+1}^{2n} b_{ipjq}\, y_{ip}\, y_{jq} \tag{2.10}$$

$$\text{subject to :} \quad \sum_{p=n+1}^{2n} y_{ip} = 1, \qquad 1 \le i \le n, \tag{2.11}$$

$$\sum_{i=1}^{m} y_{ip} = 1, \qquad n+1 \le p \le 2n, \tag{2.12}$$

$$y_{ip} = 0 \text{ or } 1, \qquad 1 \le i \le n, \qquad n+1 \le p \le 2n, \tag{2.13}$$

where b_{ipjq} is the interactive cost when machine i is assigned to location p and machine j is assigned to location q simultaneously.

When all b_{ipjq} can be expressed as

$$b_{ipjq} = f_{ij}\, d_{pq} \ ,$$

where $[f_{ij}]$ and $[d_{pq}]$ are two $n \times n$ matrices, the problem is called the Koopmans-Beckmann Problem. Obviously, the Koopmans-Beckmann Problem is a special case of the Quadratic Assignment Problem.

Cook[1971] has proved that the KBP is NP-complete (also see Garey and Johnson[1979]). Moreover, Sahni and Gonzalez[1976] showed that the following ϵ-Approximation problem for the KBP is also NP-complete for any $\epsilon > 0$:

ϵ-Approximation Problem: Suppose F^* is the optimal value of the KBP. Given $\epsilon > 0$, find an approximate solution whose value $\hat{F}$ satisfies that

$$\frac{\left| F^* - \hat{F} \right|}{F^*} \leq \epsilon.$$

The fact that even the problem of finding an approximate solution to the KBP is NP-complete attests to the difficulty of constructing good algorithm for this problem.

Remark. It is well known that any feasible solution of an assignment problem can be described as a subgraph in a bipartite graph (Figure 2-1). In this graph, vertices 1 to n represent the machines to be assigned. Vertices $n+1$ to $2n$ the locations. A solution $\mathbf{y} = \left[y_{ip} \right]$ is feasible for (2.11) - (2.13) if and only if the subgraph represented by $\mathbf{y}$ consists of n edges with no two of them sharing a common vertex.

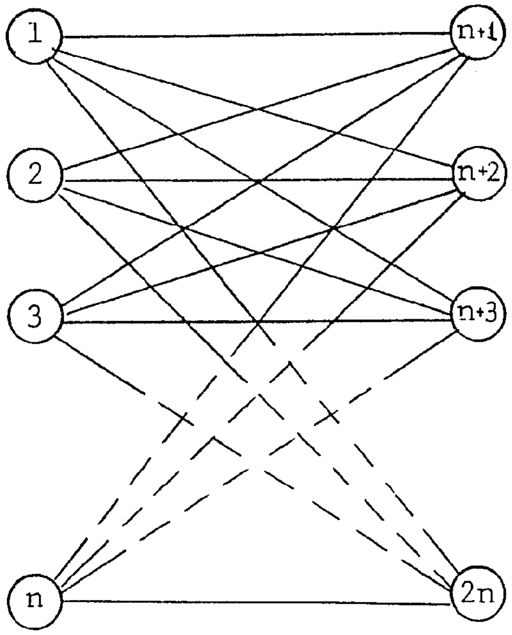

Figure 2-1.

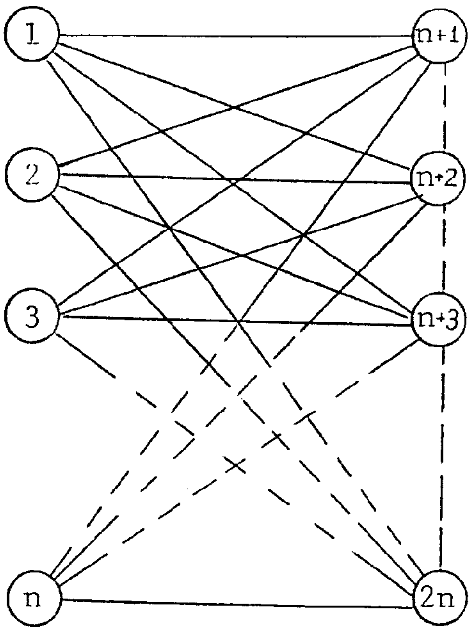

Figure 2-2.

We also rewrite the QMST in the following form which is slightly different from (2.4).

Let $V = \{ v_1, v_2, \dots, v_n \}$ be the vertex set and E the edge set. Denote by e_{ip}, $i < p$, the edge connecting vertex v_i and vertex v_p. Let c_{ipjq} be the interactive cost between e_{ip} and e_{jq}. Then the QMST has the following form:

(QMST) Minimize $z(\mathbf{x}) = \sum_{e_{ip} \in E} \sum_{e_{jq} \in E} c_{ipjq} x_{ip} x_{jq}$ (2.14)

subject to : $\mathbf{x} \in T(G)$ (2.15)

Since the QMST can clearly be solved in nondeterministic polynomial time, to show that it is NP-complete it suffices to show that any instance of the QAP can be polynomially transformed into an instance of the QMST. This is established in the next result.

Theorem 2.1. The Quadratic Assignment Problem (QAP) polynomially transforms to the Quadratic Minimal Spanning Tree Problem (QMST), that is, QAP $\propto$ QMST.

Proof. Let the bipartite graph related to the QAP be as shown in Figure 2-1. Add $n-1$ fictitious edges to join locations p and $p+1$ for $n+1 \le p \le 2n-1$. The resulting graph $G = (V, E)$ is shown in Figure 2-2 which has $2n$ vertices and $m = n^2 + (n-1)$ edges. Define the interactive costs between edge pairs for a QMST as follows.

(1) A fictitious edge has zero intercost with any other edges:

$$c_{ipq,q+1} = 0, \qquad 1 \le i \le n, \quad n+1 \le p \le 2n, \quad n+1 \le q \le 2n-1, \tag{2.16}$$

$$c_{p,p+1,q,q+1} = 0, \qquad n+1 \le p, q \le 2n-1. \tag{2.17}$$

(ii) The intercost between two adjacent edges is set to infinity:

$$c_{ipjq} = \infty\,, \qquad 1 \le i, j \le n\,, \quad n+1 \le p, q \le 2n\,,$$
$$i = j,\ p \ne q \quad \text{or} \quad i \ne j,\ p = q. \tag{2.18}$$

(iii) Other intercosts remain the same with those of the given QAP :

$$c_{ipjq} = b_{ipjq}\,, \qquad 1 \le i, j \le n\,, \quad i \ne j\,,$$
$$n+1 \le p, q \le 2n\,, \quad p \ne q. \tag{2.19}$$
$$c_{ipip} = b_{ipip}\,, \qquad 1 \le i \le n\,, \quad n+1 \le p \le 2n. \tag{2.20}$$

Let $\mathbf{y} = (y_{ip})$ be any feasible assignment for the QAP. In view of the remark, the subgraph representing this assignment consists of n edges (Figure 2-1) with no two edges sharing a common vertex. Add all fictitious edges to this subgraph and we will obtain a spanning tree in G (Figure 2-2). The objective value of this spanning tree for the QMST is obviously the same with that of $\mathbf{y}$ for the QAP due to our definition (i) and (iii). Thus we have shown that

(a) Any feasible assignment for the QAP corresponds to a spanning tree for the QMST with the same objective value.

On the other hand, let $\mathbf{x}$ be a spanning tree in G with finite objective value. By our definition (ii), $\mathbf{x}$ can not have nonfictitious edges incidental to a common vertex. Thus, $\mathbf{x}$ can have at most n nonfictitious edges. As a spanning tree, however, $\mathbf{x}$ must consist of $2n-1$ edges in G . This requires that $\mathbf{x}$ must contain all $n-1$ fictitious edges in addition to n nonfictitious ones. These n nonfictitious edges provide a feasible assignment for the QAP. Again, due to definitions (i) and (iii), this assignment has the same objective value with that of $\mathbf{x}$ for the QMST. Hence we have shown that

(b) Any feasible solution to the QMST with a finite objective value corresponds to an assignment for the QAP with the same objective value.

(a) and (b) together guarantee that solving the QMST also gives an optimal solution for the QAP. Therefore, QAP $\propto$ QMST.

2.4. NP-Completeness of the AQMST

We will show that even a special case of the QMST, the Adjacent-only Quadratic Minimal Spanning Tree Problem introduced in Section 2.1, is also NP-complete by showing that the following Hamiltonian Path Problem (HPP) transforms to it (The definition of the Hamiltonian Path Problem can be found in Appendix A).

Hamiltonian Path Problem: Given an undirected graph $G = (V, E)$. Does G contains a Hamiltonian path?

Karp[1972] proved that the Hamiltonian Path Problem is NP-complete. An improved proof can be found in Garey and Johnson[1979].

For convenience, we state two lemmas below which are easy to prove.

Lemma 2.2. A Hamiltonian path in an undirected graph is a spanning tree in which exactly two vertices have degree of 1 while all others have degree of 2.

Lemma 2.3. A tree with n vertices has a total degree of 2(n-1) summed over all vertices.

Theorem 2.2. The Hamiltonian Path Problem (HPP) polynomially transforms to the Adjacent-only Quadratic Minimal Spanning Tree Problem, i.e., HPP $\propto$ AQMST.

<u>Proof.</u> Let $G = (V, E)$ be the graph given for the Hamiltonian Path Problem. We define an AQMST on G by letting

$$\begin{cases} c_{ipjq} = 1, & \text{when } e_{ip} \text{ and } e_{jq} \text{ are adjacent,} \\ c_{ipjq} = 0, & \text{otherwise,} \end{cases} \tag{2.21}$$

in the formulation (2.14).

Solve this AQMST. Let $\mathbf{x}$ be the optimal solution and $z(\mathbf{x})$ the optimal value. We claim that if $z(\mathbf{x}) = 2(n-2)$ then $\mathbf{x}$ is a Hamiltonian path. Otherwise, if the previous equality does not hold, no Hamiltonian paths exist.

Assume that there is a Hamiltonian path and denote it by $\hat{\mathbf{x}}$. By lemma 2.2, $\hat{\mathbf{x}}$ is also a feasible solution for the AQMST. Since there are exactly $n-2$ vertices having degree of 2 in the subgraph represented by $\hat{\mathbf{x}}$, there are $n-2$ pairs of adjacent edges. Thus, its objective value for the AQMST is

$$z(\hat{\mathbf{x}}) = 2(n-2). \tag{2.22}$$

Here the factor of 2 appears because there are two terms, say $c_{ipjq}\, x_{ip}\, x_{jq}$ and $c_{jqip}\, x_{jq}\, x_{ip}$, corresponding to each pair of adjacent edges e_{ip} and e_{jq} and $c_{ipjq} = c_{jqip} = 1$ due to (2.21).

Now we prove that if $\mathbf{x}^*$, the optimal solution tree for the AQMST, is not a Hamiltonian path, then $z(\mathbf{x}^*) > 2(n-2)$. Let $\phi(i)$ be the degree of vertex v_i. Since $\mathbf{x}^*$ is a tree,

$$\phi(i) \geq 1, \qquad 1 \leq i \leq n. \tag{2.23}$$

Since $\mathbf{x}^*$ is a spanning tree, we have

$$\sum_{i=1}^{m} \phi(i) = 2(n-1) \tag{2.24}$$

by lemma 2.3. Assume that $\mathbf{x}^*$ is not a Hamiltonian path, then there must be at least one vertex which has a degree greater than 2. By reordering the vertices properly, we may assume that

$$\phi(i) \begin{cases} \geq 3 , & 1 \leq i \leq k , \\ = 2 , & k+1 \leq i \leq l , \\ = 1 , & l+1 \leq i \leq n , \end{cases} \tag{2.25}$$

where $k \geq 1$. From (2.24), we can derive that

$$\sum_{i=1}^{k} \phi(i) = 2(n-1) - 2(l-k) - (n-l)$$

$$= n - 2 - l + 2k. \tag{2.26}$$

At a vertex having degree $\phi(i) \geq 2$, there are

$$\binom{\phi(i)}{2} = \frac{1}{2} \phi(i)[\phi(i) - 1]$$

pairs of adjacent edges. Each pair contributes 2 to the objective thus $\mathbf{x}^*$ has an objective value

$$z(\mathbf{x}^*) = \sum_{i=1}^{k} \phi(i)[\phi(i) - 1] + 2(l-k)$$

$$\geq 2 \sum_{i=1}^{k} \phi(i) + 2(l-k).$$

The inequality comes from (2.25). In view of (2.26),

$$z(\mathbf{x}^*) \geq 2(n - 2 - l + 2k) + 2(l - k)$$

$$= 2(n-2) + 2k > 2(n-2) \tag{2.27}$$

(2.22) and (2.27) imply that if a Hamiltonian path exists, $z(\mathbf{x}^*) = 2(n-2)$ and if $z(\mathbf{x}^*) = 2(n-2)$, $\mathbf{x}^*$ must be a Hamiltonian path. This confirms our claim and completes the proof.

CHAPTER 3

BOUNDING TECHNIQUES FOR THE QMST

Given a graph $G = (V,E)$ with n vertices and m edges, we consider the Quadratic Minimal Spanning Tree problem:

$$\text{(QMST)} \qquad \text{Minimize} \quad z(\mathbf{x}) = \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij} x_i x_j = \mathbf{x}^t A \mathbf{x} \tag{3.1}$$

$$\text{subject to:} \quad \mathbf{x} \in T(G) \, , \tag{3.2}$$

where $T(G)$ is the collection of all spanning trees on graph G and $A = \left[a_{ij} \right]$ is the intercost matrix with $a_{ii} = b_i$, $1 \leq i \leq m$.

We have already shown that the QMST is NP-complete. Therefore, it is unlikely to be solvable by a computationally efficient algorithm. Thus, in order to obtain an optimal solution to the QMST, we must use implicit enumeration methods. Given the importance of having good bounding techniques for such methods, we consider several bounding techniques for the QMST in this chapter.

3.1. Linear Programming Approach

We show in this section that the QMST is equivalent to the following two problems named Q1 and Q2 .

$$\text{(Q1)} \qquad \text{Minimize} \quad v(\mathbf{x},\mathbf{w}) = \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij} w_{ij} \tag{3.3}$$

subject to : $\mathbf{x} \in T(G)$, (3.2)

$$\sum_{\substack{j=1 \\ j\neq i}}^{m} w_{ij} = (n-2)x_i, \quad 1\le i\le m, \tag{3.4}$$

$$w_{ij} = w_{ji}, \quad 1\le i,j\le m, \quad i\neq j, \tag{3.5}$$

$$0 \le w_{ij} \le 1, \quad 1\le i,j\le m, \quad i\neq j. \tag{3.6}$$

Problem (Q2) is obtained by replacing (3.5) with

$$\sum_{\substack{i=1 \\ i\neq j}}^{m} w_{ij} = (n-2)x_j, \quad 1\le j\le m . \tag{3.5'}$$

It is known that the spanning tree constraints (3.2) can be formulated as a set of linear constraints with 0-1 variables mixed with continuous variables(cf. Gavish[1982]). If the 0-1 variables are relaxed to be continuous variables between zero and one, we obtain the linear programming relaxation of problem Q1 . The optimal value of this linear programming problem serves as a lower bound to the original QMST.

Theorem 3.1. The three problems QMST, Q1 and Q2 are equivalent.

<u>Proof.</u> Denote by z^* and $\hat{v}$ the optimal values for the QMST and problem Q1/or Q2 respectively. We shall prove that $z^* = \hat{v}$.

Suppose that $\mathbf{x}^* = (x_i)$ is an optimal solution for the QMST. Let

$$w^*_{ij} = x^*_i x^*_j, \quad 1\le i,j\le m, \quad i\neq j. \tag{3.7}$$

Then

$$\sum_{\substack{j=1 \\ j\neq i}}^{m} w^*_{ij} = \sum_{\substack{j=1 \\ j\neq i}}^{m} x^*_i x^*_j = x^*_i \sum_{\substack{j=1 \\ j\neq i}}^{m} x^*_j$$

$$= x_i^* (n-1-x_i^*) = (n-2)x_i^*,$$

here we used $(x_i^*)^2 = x_i^*$ because $x_i^* =$ 0 or 1. Thus, $(\mathbf{x}^*,\mathbf{w}^*)$ satisfies (3.4). Easy to check that it also satisfies (3.5) and (3.6). Hence it is a feasible solution for problem Q1 (or Q2). $v(\mathbf{x}^*,\mathbf{w}^*)$, the objective value of $(\mathbf{x}^*,\mathbf{w}^*)$ is clearly equal to z^*. Therefore, $\hat{v} \leq z^*$.

On the other hand, let $(\hat{\mathbf{x}},\hat{\mathbf{w}})$ be an optimal solution for problem Q1 (or Q2). We will show that $\hat{w}_{ij} = \hat{x}_i\hat{x}_j$.

First note that $\hat{w}_{ij} \neq 0$ only if $\hat{x}_i = \hat{x}_j = 1$ since if $\hat{x}_i = 0$ then $\hat{w}_{ij} = 0$ by (3.4) and if $\hat{x}_j = 0$ then $\hat{w}_{ji} = 0$ which implies $\hat{w}_{ij} = 0$ by (3.5) (or: $\hat{w}_{ji} = 0$ directly from (3.5')). Next we show that $\hat{x}_i = \hat{x}_j = 1$ implies that $\hat{w}_{ij} = 1$. Fix i and suppose that $\hat{x}_i = 1$. Then (3.4) reads

$$\sum_{j \neq i} \hat{w}_{ij} = (n-2). \tag{3.8}$$

Now since $\hat{\mathbf{x}} \in T(G)$, exactly (n-1) $\hat{x}_i$'s are equal to 1. This says that all but (n-2) terms in the sum (3.8) are zero. Because of (3.6), all such terms must equal 1 for (3.8) to hold. So we have shown that $\hat{w}_{ij} = 1$ whenever $\hat{x}_i = \hat{x}_j = 1$. Thus $\hat{w}_{ij} = \hat{x}_i\hat{x}_j$, $1 \leq i,j \leq m$, and

$$\hat{v} = \sum_{i=1}^{m} b_i\hat{x}_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j\neq i}}^{m} a_{ij}\hat{x}_i\hat{x}_j$$

$$= z(\hat{\mathbf{x}}) \geq z^*.$$

In summary, the equality $\hat{v} = z^*$ must hold thereby completing the proof.

Because of the symmetry condition (3.5), there are actually m(m+1)/2 variables in problem Q1 . Considering that m could be large even for a network of moderate size and that there are a large number of extra constraints [(3.4) and (3.6)], evaluating a lower bound by solving the linear programming problem will be quite time consuming.

3.2. Lagrangean Relaxation Approach

The Lagrangean relaxation method is often used to derive lower bounds for various hard combinatorial problems. Let us consider this approach for the QMST . Introducing a set of multipliers $\lambda = (\lambda_i)$, $1 \leq i \leq m$, to relax (3.5') in problem Q2, the objective function becomes

$$\sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij} w_{ij} + \sum_{j=1}^{m} \lambda_j \left[\sum_{\substack{i=1 \\ i \neq j}}^{m} w_{ij} - (n-2) x_j \right]$$
$$= \sum_{i=1}^{m} \left[b_i - (n-2)\lambda_i \right] x_i + \sum_{i=1}^{m} \sum_{j \neq i} \left(a_{ij} + \lambda_j \right) w_{ij}.$$

The Lagrangean relaxation problem is then:

LR(λ): Minimize $\displaystyle\sum_{i=1}^{m} b_i(\lambda) x_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij}(\lambda) w_{ij}$ (3.9)

subject to : $\mathbf{x} \in T_G$, (3.2)

$$\sum_{\substack{j=1 \\ j \neq i}} w_{ij} = (n-2) x_i , \quad 1 \leq i \leq m. \tag{3.4}$$

$$0 \leq w_{ij} \leq 1 , \quad 1 \leq i,j \leq m, \ i \neq j. \tag{3.6}$$

where

$$\begin{aligned} b_i(\lambda) &= b_i - (n-2)\lambda_i , & 1 \leq i \leq m , \\ a_{ij}(\lambda) &= a_{ij} + \lambda_j , & 1 \leq i,j \leq m , \quad i \neq j . \end{aligned} \tag{3.10}$$

Notice that the symmetry condition $w_{ij} = w_{ji}$ is not required in problem LR(λ).

According to the general results of the Lagrangean relaxation (see Geofferion [1974]), the optimal value of problem LR(λ), denoted by L(λ), is a lower bound on problem Q2 hence on the original QMST. If there is an efficient method of solving LR(λ) for a fixed set of values of the multipliers $\lambda = (\lambda_i)$, then one can use the subgradient method to search for the multipliers $\lambda^* = (\lambda_i^*)$ that solve the Lagrangean dual

$$L(\lambda^*) = \max_{\lambda} L(\lambda). \tag{3.11}$$

However, we will not solve (3.11) directly. Instead, we will propose another bounding technique in section 4 that is superior to the present one both in quality and in computational efficiency.

3.3. The Decomposition Bound (*DB*)

From now on, we write for short the QMST as

$$\min\left\{ z(\mathbf{x}) = \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{j \neq i} a_{ij} x_i x_j \mid \mathbf{x} \in T(G) \right\}. \tag{3.12}$$

Let

$$f_i = \min \left\{ b_i + \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij} x_j \mid x_i = 1,\ \mathbf{x} \in T(G) \right\}, \quad 1 \leq i \leq m, \tag{3.13}$$

and

$$DB = \min \left\{ \sum_{i=1}^{m} f_i x_i \mid \mathbf{x} \in T(G) \right\}. \tag{3.14}$$

Both (3.13) and (3.14) are ordinary minimal spanning tree problems. They can be solved by efficient algorithms mentioned in Appendix B. The evaluation of *DB* decomposes into solving a number of ordinary minimal spanning tree problems by solving (3.13) for all i and then (3.14). For this reason it is called the *Decomposition Bound* . we now prove that the quantity DB is indeed a lower bound.

Theorem 3.2. DB provides a lower bound for the Quadratic Minimal Spanning Tree problem.

<u>proof.</u> Assume that $\mathbf{x} = (x_i)$, $1 \le i \le m$, is a spanning tree. Note that

$$z(\mathbf{x}) = \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j\neq i}}^{m} a_{ij} x_i x_j$$

$$= \sum_{i=1}^{m} \left[b_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} a_{ij} x_j \right] x_i .$$

Now, the relation

$$\left[b_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} a_{ij} x_j \right] x_i \ge f_i x_i$$

holds for $x_i = 1$ by the definition of f_i in (3.13) and is trivially true if $x_i = 0$. Thus

$$z(\mathbf{x}) \ge \sum_{i=1}^{m} f_i x_i \ge DB .$$

3.4. The Parametrized Bound (*PB*)

Let $\alpha = (\alpha_i)$ and $\beta = (\beta_i)$, $1 \le i \le m$, be two sets of real parameters. Define

$$\begin{cases} \bar{b}_i = b_i - (n-2)\alpha_i - (n-2)\beta_i , & 1 \leq m , \\ \bar{a}_{ij} = a_{ij} + \alpha_j + \beta_i , & 1 \leq i,j \leq m, \quad i \neq j. \end{cases} \tag{3.15}$$

Let $f_i(\alpha,\beta)$ and $PB(\alpha,\beta)$ denote the minimal values defined by (3.13) and (3.14) respectively, when b_i and a_{ij} are replaced by $\bar{b}_i$ and $\bar{a}_{ij}$.

Theorem 3.3. $PB(\alpha,\beta)$ provides a lower bound for problem (3.12) for any choice of α and β. Moreover, the value of $PB(\alpha,\beta)$ is independent of β, i.e. $PB(\alpha,\beta)$ only depends on α.

Proof. We can write $z(\mathbf{x})$ in (3.12) as

$$z(\mathbf{x}) = \mathbf{x}^t A \mathbf{x} ,$$

where A is a matrix with off-diagonal entries a_{ij} and diagonal entries $a_{ii} = b_i$. Let $\bar{A}$ be the corresponding matrix with entries defined in (3.15) and $\bar{A}_{ii} = \bar{b}_i$. Clearly we have

$$\bar{A} = A + Q$$

where $Q = \left[q_{ij}\right]$ and

$$q_{ij} = \alpha_j + \beta_i - (n-1)(\alpha_i + \beta_i)\delta_{ij} ,$$

where δ_{ij} is the Kronecker δ..

Moreover,

$$\bar{z}(\mathbf{x}) = \mathbf{x}^t \bar{A} \mathbf{x} = \mathbf{x}^t A \mathbf{x} .$$

Now since $\sum_{i=1}^{m} x_j = n-1$, direct computation shows that $\mathbf{x}^t Q \mathbf{x} = 0$. This shows that $\bar{z}(\mathbf{x}) = z(\mathbf{x})$. The original problem (3.12) is equivalent to

$$\min \left\{ \bar{z}(\mathbf{x}) = \sum_{i=1}^{m} \bar{b}_i + \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} \bar{a}_{ij} x_i x_j \mid \mathbf{x} \in T(G) \right\} . \tag{3.16}$$

Theorem 3.2 then implies that $PB(\alpha,\beta)$ is a lower bound for the problem in (3.16) and hence also for the original problem.

Note that by (3.15), the objective being minimized in

$$f_i(\alpha,\beta) = \min \left\{ \overline{b}_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} \overline{a}_{ij}x_j \mid x_i=1, \ \mathbf{x} \in T(G) \right\}$$

equals

$$\begin{aligned} & b_i - (n-2)\alpha_i - (n-2)\beta_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} (a_{ij} + \alpha_j + \beta_i) x_j \\ &= b_i - (n-2)\alpha_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} (a_{ij} + \alpha_j)x_j - (n-2)\beta_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} \beta_i x_j . \end{aligned}$$

Since $x_i = 1$, $\sum_{j\neq i} x_j = n-2$, the last two terms cancel. The remaining terms are equal to the objective function for $f_i(\alpha,0)$. Since $f_i(\alpha,\beta)$'s do not depend on β , the same is true for $PB(\alpha,\beta)$. Thus completes the proof.

In view of Theorem 3.3, the parameters β can be set to zero in the transformation (3.15):

$$\begin{cases} b_i(\alpha) = b_i - (n-2)\alpha_i , & 1\leq i\leq m , \\ a_{ij}(\alpha) = a_{ij} + \alpha_j , & 1\leq i,j\leq m , \ i\neq j . \end{cases} \tag{3.17}$$

The objective values of (3.13) and (3.15) with new intercost matrix $A(\alpha) = \left[a_{ij}(\alpha)\right]$, $a_{ii}(\alpha) = b_i(\alpha)$, will be denoted by $f_i(\alpha)$ and $PB(\alpha)$ correspondingly. $PB(\alpha)$ is called the *Parametrized Bound* with parameter vector α .

When all α_i's are zero, the parametrized bound $PB(0)$ coincides with the decomposition bound DB . One may ask if there is a relation between this bound and the Lagrangean relaxation bound mentioned in section 2. In fact, we can use the similarity between the two transformations of (3.10) and (3.17) to obtain a relation between these two lower bounds. For a fixed set of parameters $\alpha = (\alpha_i)$, we can prove the following

Theorem 3.4. Let $L(\alpha)$ be the Lagrangean relaxation bound and $PB(\alpha)$ be the parametrized bound corresponding to the same α. Then

$$L(\alpha) \leq PB(\alpha) . \tag{3.18}$$

Proof. Let $\mathbf{x}^{(i)} = (x_j^{(i)})$, $1 \leq i \leq m$, be optimal solutions for (3.13) and $\mathbf{x}^* = (x_j^*)$ be an optimal solution for (3.14) with b_i and a_{ij} replaced by $b_i(\alpha)$ and $a_{ij}(\alpha)$ under the transformation in (3.17). Define $w_{ij}^* = x_i^* x_j^{(i)}$, $1 \leq i,j \leq m$, $i \neq j$. Then since $0 \leq w_{ij}^* \leq 1$ and

$$\sum_{\substack{j=1 \\ j\neq i}}^{m} w_{ij}^* = x_i^* \sum_{\substack{j=1 \\ j\neq i}}^{m} x_j^{(i)} = (n-2)x_i^* , \qquad 1 \leq i \leq m ,$$

$(\mathbf{x}^*,\mathbf{w}^*)$ is feasible for the $\mathrm{LR}(\alpha)$ and has an objective value of $PB(\alpha)$. Thus (3.18) is true.

The converse of (3.18) is not true in general. Here, we give a counter example:

Example. Let G be a graph of 4 vertices and 4 edges given in figure 3-1. The intercost matrix is given by

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

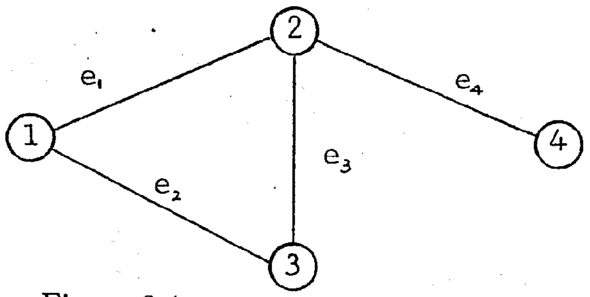

Figure 3-1.

with $b_1 = b_2 = b_3 = b_4 = 0$.

Take $\alpha = 0$.

Note that the edge e_4 must be included in any spanning tree over the given graph. Then it is easy to find that $f_1 = f_2 = f_3 = 1$, $f_4 = 2$. Take f_1 as example. By (3.13), we have

$$f_1(0) = \min\left\{ b_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 \mid x_1{=}1,\ \mathbf{x} \in T(G) \right\}$$

$$= \min\left\{ x_4 \mid x_1 = 1,\ \mathbf{x} \in T(G) \right\} = 1.$$

Then by (3.14),

$$PB(0) = \min\left\{ x_1 + x_2 + x_3 + 2x_4 \mid \mathbf{x} \in T(G) \right\} = 4.$$

Now let us chose

$$\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \qquad \mathbf{w} = \begin{pmatrix} - & 1 & 1 & 0 \\ 1 & - & 1 & 0 \\ 0 & 0 & - & 0 \\ 0 & 1 & 1 & - \end{pmatrix},$$

where the "-" in matrix A indicates that $i \neq j$ for w_{ij}. Then $\mathbf{x} \in T(G)$. $(\mathbf{x},\mathbf{w})$ also satisfies (3.4) and (3.6). Hence it is feasible for problem LR(0). Since its objective value (3.9) equals 2, we have

$$L(0) \leq 2 < PB(0) = 4\ .$$

The above discussion show that the parametrized lower bound is always at least as good as the Lagrangean one. Consequently, we will focus on the former in what follows.

It can be expected that different sets of parameters may result in different parametrized bounds. How then is one to choose the best parameters α^* so that

$$PB(\alpha^*) = \max_{\alpha}\ PB(\alpha)\ \ ? \tag{3.19}$$

It is desirable to have an efficient method capable of finding the best parameter

Note that the edge e_4 must be included in any spanning tree over the given graph. Then it is easy to find that $f_1 = f_2 = f_3 = 1$, $f_4 = 2$. Take f_1 as example. By (3.13), we have

$$f_1(0) = \min\left\{ b_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 \mid x_1{=}1,\ \mathbf{x} \in T(G) \right\}$$

$$= \min\left\{ x_4 \mid x_1 = 1,\ \mathbf{x} \in T(G) \right\} = 1.$$

Then by (3.14),

$$PB(0) = \min\left\{ x_1 + x_2 + x_3 + 2x_4 \mid \mathbf{x} \in T(G) \right\} = 4.$$

Now let us chose

$$\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \qquad \mathbf{w} = \begin{pmatrix} - & 1 & 1 & 0 \\ 1 & - & 1 & 0 \\ 0 & 0 & - & 0 \\ 0 & 1 & 1 & - \end{pmatrix},$$

where the "-" in matrix A indicates that $i \neq j$ for w_{ij}. Then $\mathbf{x} \in T(G)$. $(\mathbf{x},\mathbf{w})$ also satisfies (3.4) and (3.6). Hence it is feasible for problem LR(0). Since its objective value (3.9) equals 2, we have

$$L(0) \leq 2 < PB(0) = 4 .$$

The above discussion show that the parametrized lower bound is always at least as good as the Lagrangean one. Consequently, we will focus on the former in what follows.

It can be expected that different sets of parameters may result in different parametrized bounds. How then is one to choose the best parameters α^* so that

$$PB(\alpha^*) = \max_{\alpha} \ PB(\alpha) \ ? \qquad (3.19)$$

It is desirable to have an efficient method capable of finding the best parameter set α^*.

3.5. The Leveling Method for Searching the Best PB

The search for the "best" parameter set in (3.19) may be performed with the subgradient method. However, this method usually needs many iterations and takes a long time to arrive at a satisfactory (approximate) solution. In order to make the branch-and-bound procedure more practical, we propose a new method, called the *Leveling Method,* to solve (3.19). We first state briefly a theorem that serves as a basis for the leveling method and then provide an algorithm for the method.

After obtaining $f_i(\alpha)$, $1 \leq i \leq m$, by solving (3.13), we define

$$f_{\max}(\alpha) = \max_i \left\{ f_i(\alpha) \right\} ,$$

$$f_{\min}(\alpha) = \min_i \left\{ f_i(\alpha) \right\} .$$

and

$$\phi(\alpha) = f_{\max}(\alpha) - f_{\min}(\alpha) .$$

Call $\phi(\alpha)$ the deviation of α.

Theorem 3.5. Assume α^* is the best parameter set for problem (3.19). If $\hat{\alpha}$ is a parameter set such that $\phi(\hat{\alpha}) \leq \epsilon/(n-1)$, then

$$PB(\hat{\alpha}) \geq PB(\alpha^*) - \epsilon .$$

We leave the proof of this theorem to Chapter 4 where it follows from more general results for a broader class of 0-1 programs.

Theorem 3.5 shows that a set of parameters with small deviation is close to the best set of parameters. Any algorithm that is efficient in reducing the deviation to zero also provides arbitrary good approximation to the best choice of parameters. Here we propose such an algorithm. It attempts to reduce $\phi(\alpha)$ iteratively by using the information obtained from the previous iteration.

The Leveling Algorithm:

In this algorithm, $\delta > 0$ is a preset number used for the stopping rule. we will discuss the choice of δ in section 3.6.

STEP 1. For $i = 1, 2, \ldots, m$, solve the ordinary MST's:

$$f_i = \min\left\{ b_i + \sum_{\substack{j=1 \\ j \neq i}}^{m} a_{ij} x_j \mid x_i = 1, \ \mathbf{x} \in T(G) \right\} .$$

STEP 2. Solve an MST:

$$PB = \min\left\{ \sum_{i=1}^{m} f_i x_i \mid \mathbf{x} \in T(G) \right\} .$$

STEP 3. Let $f_{\max} = \max_i \left\{ f_i \right\}$, $f_{\min} = \min_i \left\{ f_i \right\}$ and $\phi = f_{\max} - f_{\min}$.

If $\phi < \delta$, stop.

STEP 4. Choose a set of parameters α_i, $1 \le i \le m$. There are two options:

<u>Option 1:</u> Set $\alpha_i = f_i / (n-1)$.

<u>Option 2:</u> Set $\bar{f} = \sum_{i=1}^{m} f_i / m$ and

$$\Delta = \frac{1}{2(n-2)} \min\left\{ f_{\max} - \bar{f}, \ \bar{f} - f_{\min} \right\} .$$

Let

$$\alpha_i = \begin{cases} -\Delta\,, & \text{if } f_i \le f\,, \\ +\Delta\,, & \text{otherwise}\,. \end{cases}$$

STEP 5. Let

$$b_i \leftarrow b_i - (n-2)\alpha_i, \qquad 1 \le i \le m\,,$$
$$a_{ij} \leftarrow a_{ij} + \alpha_j\,, \qquad 1 \le i,j \le m, \quad i \ne j\,.$$

Return to step 1.

This algorithm performs very well for reducing the deviation. In addition, it improves the parametrized lower bound after each iteration. Again, a proof is given in a more general setting in Chapter 4.

Using Option 1 in step 4 (rather than Option 2) results in faster convergence. A computational comparison of these two options will be given in the next section.

3.6. Computational Experiments

In this section, we report on the computational experience with our algorithm on a set of randomly-generated QMST test problems. In particular, we address the following issues: (i) the comparison of the two options for step 4 of the algorithm, (ii) the effect of selecting different values of the parameter δ in the stopping rule, and (iii) empirical results on how well the algorithm performs.

Construction of the test problems

the graphs used in our study are all undirected, complete graphs with n vertices and $m = n(n-1)/2$ edges. The edges are ordered according to their terminal vertices so that

$$e_1 = (1,2)\ ,\ \ e_2 = (1,3)\ , \ldots,\ \ e_{n-1} = (1,n)\ ,$$
$$e_n = (2,3)\ , \ldots ,\ \ e_m = (n-1,n)\ .$$

The intercost matrices are all symmetric whose entries, including those on the diagonal, are random integers distributed uniformly between 0 and 20 inclusively. We employ the subroutine GGUBT form package IMSL to generate the uniform variate U(0,1). The package IMSL is available on the UNIVAC 1100/80 in the Computer Science Center, University of Maryland at College Park. Starting with a given seed, we generate a sequence of U(0,1)-random numbers $\{ r_i \}$. Let

$$t_i = [21 r_i\,]\ ,$$

where $[r]$ is the largest integer not exceeding r. In such a way, we obtain a sequence of random integers varied from 0 to 20. Then we use these random integers to construct the intercost matrix in the following order:

$a_{11}\ ,\ a_{12}\ ,\ a_{13}\ , \ldots ,\ a_{1m}\ ,$

$a_{22}\ ,\ a_{23}\ , \ldots ,\ a_{2m}\ ,$

$, \ldots ,$

$a_{m-1,m-1}\ ,\ a_{m-1,m}\ ,$

$a_{mm}\ .$

Finally, the matrix is completed by using the symmetry.

Comparison of two options

To evaluate the efficiency of the algorithm for each problem tested, we compute the following quantities after iteration t:

PB ---- the parametrized lower bound.

ϕ_t ---- the deviation of the current parameters.

CPU ---- CPU time in seconds used up to the current iteration.

PB/DB -- a quantity showing how much the current *PB* improves the decomposition bound *DB*.

In addition, we compute a quantity RR_t at iteration t, called the *Reduction Ratio*:

$$RR_t = \frac{\phi_t}{\phi_{t-1}} ,$$

where ϕ_t is the deviation at iteration t. Obviously, the smaller the RR_t is, the faster the deviation is forced to zero.

As an example, we show the results of applying options 1 and 2 to the same problem in Tables 3-1 and 3-2 respectively. The problem has parameters $n = 15$, $m = 105$ and starting seed = 24690.

t	PB	ϕ_t	RR_t	PB/DB	CPU(s)
1	183.000	43.000	--	1.000	.18
2	299.857	9.857	.22	1.638	1.41
3	325.219	3.230	.32	1.777	2.64
4	332.339	1.199	.37	1.816	3.87
5	334.811	.455	.37	1.829	5.11
6	335.816	.181	.39	1.835	6.34
7	336.205	.070	.38	1.837	7.57
8	336.375	.031	.43	1.838	8.80
9	336.449	.013	.42	1.838	10.04
10	336.481	.005	.42	1.838	11.27
11	336.495	.002	.43	1.838	12.50
12	336.501	.001	.42	1.838	13.73
13	336.504	.000	.42	1.838	14.69
14	336.505	.000	.45	1.838	16.20
15	336.506	.000	.39	1.838	17.43
16	336.506	.000	.54	1.838	18.66

Table 3-1. Printout for a QMST with n=15, m=105 and seed=24690. Use Option 1 in step 4.

We see from this example that the leveling algorithm reduces the deviation to zero very quickly, especially when using Option 1. Also, the parametrized lower bound improves after every iteration. The parametrized bound (PB) approaches the best value for this bound very fast (only after a few iterations). In this example, the best PB is approximately equal to 336.506. When the deviation is less than 1.0, $PB > 333.236$ which differs from the best PB no more than 1%. When the deviation is less than 0.4, $PB > 334.95$ which differs the best PB by no more than 0.5%. The results for all other test problems are almost the same.

Option 1 results in faster convergence than Option 2. This is evident from the fact that at the end of a fixed number of iterations, Option 1 results in a larger value for the parametrized bound than obtained by Option 2. Also, the

t	PB	ϕ_t	RR_t	PB/DB	CPU(s)
1	183.000	43.000	--	1.000	.18
2	196.912	28.312	.65	1.076	1.41
3	262.542	17.805	.62	1.434	2.64
4	280.718	12.046	.67	1.534	3.87
5	299.156	7.648	.63	1.724	5.10
6	315.584	5.004	.65	1.748	6.33
7	319.924	3.295	.65	1.780	7.56
8	325.776	1.996	.60	1.804	8.79
9	330.250	1.117	.55	1.821	10.03
10	333.236	.669	.59	1.827	11.26
11	334.230	.493	.73	1.830	12.49
12	334.950	.320	.64	1.833	13.72
13	335.446	.214	.67	1.834	14.96
14	335.778	.154	.71	1.836	16.19
15	336.069	.097	.63	1.837	17.42
16	336.193	.063	.64	1.837	18.66
17	336.320	.039	.62	1.838	19.89
18	336.395	.026	.65	1.838	21.12
19	336.432	.015	.59	1.838	22.35
20	336.468	.008	.53	1.838	23.59

Table 3-2. Printout of a QMST with n=15, m=105 and seed=24690 (Option 2).

reduction ratios (RR_t) in Option 1 are much smaller than in Option 2. To confirm this, we report the results of a statistical comparison between two options in Table 3-3. For each n, 20 different problems are generated by setting the starting seeds at 12345, 12345x2, 12345x3, . . . , 12345x20. We terminate the algorithm after the deviation falls below 1.0 . Then the means and standard deviations of the number of iterations and the reduction ratios are computed across the 20 problems.

	# of iteration				Reducing ratio			
	Option 1		Option 2		Option 1		Option 2	
n	μ	σ	μ	σ	μ	σ	μ	σ
6	5.26	.69	11.80	4.58	.399	.051	.684	.094
7	5.33	.60	11.45	3.17	.407	.067	.692	.067
8	5.36	.49	11.60	3.78	.416	.043	.689	.080
9	5.30	.46	12.05	4.19	.398	.038	.694	.084
10	5.13	.34	11.40	3.84	.366	.041	.682	.088
11	5.06	.25	11.35	3.82	.356	.028	.675	.086
12	5.03	.18	11.15	3.52	.355	.025	.672	.071
13	5.00	.00	11.00	3.43	.332	.012	.667	.079
14	5.00	.00	11.05	3.90	.315	.027	.670	.090
15	5.00	.00	11.70	4.33	.307	.027	.678	.094

Table 3-3. A statistical comparison between two Options.
The left and right columns contain
means and standard deviations.

Due to its superiority over Option 2, the leveling algorithm was implemented with Option 1 only in the computational work discussed below.

Inspection of the stopping rule

In step 3 of the algorithm, the stopping accuracy δ of the deviation is a preset positive number. A smaller δ gives a better PB but requires more iterations and therefore longer computation time. The issue of an appropriate choice of a value for δ arises. We now investigate the sensitivity of the algorithm to different choices of δ.

Since the deviation always reduces to 0.0001 or less after 15 iterations in every problems (regardless of problem dimension), we may take the value of the parametrized bound at the 20-th iteration as a good approximation to the best parametrized bound and denote it by PB^*. Consider the values 2.0, 1.0, 0.5, 0.2 and 0.1 for δ. For each problem, the number of iterations needed to reduce the deviation to below δ and the parametrized bounds are recorded. Then we compute two ratios: PB/PB^* and PB/DB. The first quantity indicates how close this PB is to the best one while the second quantity shows how much this parametrized bound improves the decomposition bound. We also keep a record of the parametrized bounds at the second iteration, denoted by PB_1, and the corresponding ratios PB_1/PB^* and PB_1/DB.

A total of 50 problems are tested for each n. These different problems are generated by setting the starting seeds at 12345, 12345x2, . . . , 12345x50. The means and standard deviations of the preceding quantities are computed across the 50 problems and shown in Table 3-4 to Table 3-9.

Table 3-4 shows that there is little disparity in the numbers of iterations needed to reduce the deviation to below δ. For instance, the deviation is less than 2.0 after 5 iterations, less than 1.0 after 6 iterations, etc., even though the problem dimension varies.

	Average # of Iterations					Maximum # of Iter.				
n	2.	1.	.5	.2	.1	2.	1.	.5	.2	.1
6	4.26	5.36	6.46	7.86	9.04	5	7	8	10	11
7	4.32	5.32	6.34	7.72	8.54	6	7	8	9	11
8	4.22	5.22	6.22	7.36	8.36	5	6	8	9	10
9	4.08	5.08	6.10	7.20	8.20	5	6	7	9	10
10	4.14	5.14	6.14	7.34	8.34	5	6	7	8	9
11	4.04	5.04	6.04	7.08	8.08	5	6	7	8	9
12	4.00	5.00	6.00	7.02	8.02	5	6	7	8	9
13	4.00	5.00	6.00	7.00	8.00	4	5	6	7	8
14	3.98	4.98	5.98	6.98	7.98	4	5	6	7	8
15	4.00	5.00	6.00	7.00	8.00	4	5	6	7	8

Table 3-4. Numbers of iterations needed to reach specified δ.

	δ				
n	2.	1.	.5	.2	.1
6	.600	.693	.762	.904	1.087
7	.593	.635	.642	.822	.842
8	.471	.471	.519	.614	.646
9	.418	.418	.418	.525	.525
10	.274	.274	.303	.404	.404
11	.198	.198	.198	.198	.198
12	.198	.198	.198	.274	.274
13	.000	.000	.000	.141	.141
14	.000	.000	.000	.000	.000
15	.141	.141	.141	.141	.141

Table 3-5. Deviation of the average number of iterations.

		δ				
n	PB_1/PB^*	2.0	1.0	0.5	0.2	0.1
6	.921	.986	.993	.996	.998	.999
7	.914	.987	.994	.997	.999	.999
8	.909	.987	.994	.997	.999	.999
9	.900	.986	.994	.997	.999	.999
10	.900	.986	.994	.997	.999	.999
11	.893	.986	.994	.997	.999	.999
12	.896	.986	.994	.997	.999	.999
13	.889	.986	.994	.998	.999	1.000
14	.891	.987	.995	.998	.999	1.000
15	.890	.988	.995	.998	.999	1.000

Table 3-6. The quantity PB/PB^* for different δ.

		δ				
n	PB_1/PB^*	2.0	1.0	0.5	0.2	0.1
6	.022	.006	.004	.002	.001	.001
7	.021	.005	.003	.001	.001	.000
8	.020	.004	.002	.001	.001	.000
9	.016	.003	.002	.001	.001	.000
10	.019	.003	.001	.001	.000	.000
11	.016	.003	.002	.001	.001	.000
12	.015	.003	.002	.001	.001	.000
13	.013	.003	.002	.001	.001	.000
14	.012	.003	.001	.001	.000	.000
15	.011	.003	.001	.001	.000	.000

Table 3-7. Deviation of the ratio PB/PB^*.

	δ					
n	PB_1/DB	2.0	1.0	0.5	0.2	0.1
6	1.232	1.319	1.329	1.333	1.336	1.337
7	1.278	1.380	1.390	1.394	1.397	1.397
8	1.323	1.438	1.447	1.452	1.454	1.455
9	1.341	1.471	1.482	1.487	1.489	1.490
10	1.380	1.513	1.525	1.530	1.532	1.533
11	1.442	1.593	1.606	1.611	1.613	1.615
12	1.452	1.599	1.613	1.618	1.621	1.622
13	1.499	1.664	1.678	1.683	1.686	1.687
14	1.520	1.685	1.698	1.703	1.705	1.706
15	1.552	1.724	1.737	1.741	1.743	1.744

Table 3-8. The quantity PB/DB for different δ.

	δ					
n	PB_1/DB	2.0	1.0	0.5	0.2	0.1
6	.089	.102	.104	.105	.106	.106
7	.076	.091	.092	.093	.093	.093
8	.074	.092	.093	.093	.093	.093
9	.105	.129	.131	.131	.131	.132
10	.071	.092	.094	.094	.095	.095
11	.091	.108	.110	.110	.110	.110
12	.089	.113	.114	.114	.114	.114
13	.070	.082	.084	.084	.084	.084
14	.080	.097	.098	.098	.098	.098
15	.104	.122	.124	.124	.124	.124

Table 3-9. Deviation of the ratio PB/DB.

Table 3-6 gives another indication of the algorithm's stable behavior. When the deviation falls below a certain δ, say 0.5, the parametrized lower bound is greater than 0.997 of the best parametrized lower bound, again regardless of the problem dimension.

Naturally, the choice of δ depends on the input data, i.e. the matrix $A = [\ a_{ij}\ ]$ and $\{\ b_i\ \}$. In each of our test problems, the largest entry in the input data, denoted by $a_{\max}$, is 20. $\delta = 2.0$ corresponds to $\delta = 10\%$ of $a_{\max}$.

$\delta = 0.5$ corresponds to $\delta = 2.5\%$ of $a_{\max}$. Now let $E_r = (PB^* - PB)/PB^*$ denote the relative error of PB against PB^*. Then the stopping accuracy δ, measured by a certain percentage of $a_{\max}$, and the relative error in PB are related as follows:

δ (% of $a_{\max}$)	E_r (less than)
10.0%	2.0%
5.0%	1.0%
2.5%	0.5%
1.0%	0.2%
0.5%	0.1%

One may choose a $\delta > 0$ a priori once he decides what the relative error Er should be.

The small standard deviations of any quantities in Tables 3-5, 3-7 and 3-9 may be used to give further confirmation of the algorithm's reliability for different problems.

Further inspection

From the above discussion, we may conclude that the leveling algorithm serves as a reliable and efficient method for solving the parameter search problem (3.19).

One measure of the algorithm's efficiency is available in the reduction ratio $RR_t = \phi_t / \phi_{t-1}$ that indicates the extent of reduction at each iteration. In Table 3-3, the average reduction ratio is less than 1/2 for all problem dimensions. Moreover, the reduction ratio tends to be smaller for problems of larger dimensions. If we denote ϕ_1, ϕ_2, . . . , ϕ_k, . . . , the deviations at each iteration, then we would have

$$\phi_2 < \phi_1/2 , \quad \phi_2 < \phi_1/2^2 \quad , \ldots , \quad \phi_k < \phi_1/2^{k-1} \quad , \ldots ,$$

This implies that the algorithm causes the deviation to approach zero at a geometric rate. In such a case, we may say that the algorithm converges "geometrically".

However, each iteration requires solving $(m+1)$ minimal spanning tree problems hence needs $O(n^4)$ computations. When n is large, it would take too much time to evaluate a lower bound if we desire more accurate results. However, note that the lower bound has already improved significantly by the second iteration as can be seen from the ratios PB_1 / PB^* and PB_1 / DB in Tables 3-8 and 3-9. One can therefore use PB_1 as a lower bound embedded within a branch-and-bound procedure. We will adopt this approach in Chapter 6 where solution techniques are discussed.

Table 3-10 compares the parametrized lower bounds and the optimal values obtained in Chapter 6. Since all entries in the intercost matrix are integers, the lower bounds are also integers. So, if $PB = 442.342$, then it can be rounded up to 443.

Finally, we see that the gap between PB^* and the optimal value is still large. This is not surprising since the Quadratic Minimal Spanning Tree problem is at least as "hard" as the Quadratic Assignment Problem whose intractability is well-known.

n	DB	PB_1	PB^*	z^*	DB/z^*	PB_1 /z^*	PB^*/z^*
6	129	169	178	207	.623	.816	.860
7	197	272	286	326	.604	.834	.874
8	211	304	328	412	.512	.738	.796
9	260	371	416	533	.488	.696	.780
10	270	408	443	638	.423	.639	.694
11	243	410	468	740	.328	.554	.632
12	302	502	547	917	.329	.547	.597

Table 3-10. Lower bounds via the optimal values.

CHAPTER 4
A BOUNDING TECHNIQUE FOR A CLASS OF QUADRATIC 0-1 PROGRAMS

In this chapter, the leveling method designed for the Quadratic Minimal Spanning Tree problem is generalized to a broader class of quadratic 0-1 programs. We also present theorems that ensure the convergence of this method.

4.1. Brief Introduction of the Problem

A quadratic 0-1 program is a problem of the form

$$\text{(QP)} \qquad \text{Minimize} \quad z(\mathbf{x}) = \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j \tag{4.1}$$

$$\text{subject to:} \quad \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ijt} x_i x_j \leq b_t \,, \quad 1 \leq t \leq n, \tag{4.2}$$

$$x_i = 0 \text{ or } 1 \qquad \text{for all} \quad i \,. \tag{4.3}$$

Let us rewrite this problem in a more convenient form:

$$\min \left\{ z(\mathbf{x}) = \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j \mid \mathbf{x} \in S \right\} \tag{4.4}$$

where S is a subset of B_2^m , $B_2 = \{0, 1\}$, either given by a set of constraints such as (4.2) and (4.3) or implicitly in some other way. We specify the class of quadratic 0-1 program to be worked on by adding following two assumptions to the feasible set S :

Assumption (1). There is a computationally efficient algorithm to solve the problem

$$\min\left\{ \sum_{i=1}^{m} c_i x_i \mid \mathbf{x} \in S \right\} \tag{4.5}$$

which has the same feasible set as (4.4) but has a linear objective function instead of the quadratic one.

This assumption is reasonable since we would not consider a quadratic problem if we are not able to solve its linear counterpart.

Assumption (ii). The sum of all components of any feasible $\mathbf{x}$ is a fixed number:

$$\sum_{i=1}^{m} x_i = K \qquad \text{for any} \quad \mathbf{x} \in S .$$

The quadratic minimal spanning tree problem discussed in Chapters 2 and 3, the quadratic assignment problem, and possibly some other problems belong to this class. For the quadratic minimal spanning tree problem with n vertices, $K = n-1$. For the quadratic assignment problem with n plants, $K = n$ since

$$\sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} = \sum_{i=1}^{n} 1 = n .$$

4.2. The Parametrized Lower Bound

Since $x_i = 0$ or 1, $x_i^2 = x_i$. The diagonal terms in the objective of (4.4) can be separated:

$$z(\mathbf{x}) = \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{j \neq i} a_{ij} x_i x_j ,$$

where $b_i = a_{ii}$. Introduce m parameters

$$\pi = (\pi_1 , \pi_2 , \ldots , \pi_m)$$

and let

$$b_i(\pi) = b_i - (K-1)\pi_i\,, \qquad 1 \le i \le m\,, \tag{4.6}$$

$$a_{ij}(\pi) = a_{ij} + \pi_j\,, \quad 1 \le i,j \le m\,, \quad i \ne j\,, \tag{4.7}$$

where K is a fixed number satisfying assumption (ii). An argument similar to that one given for theorem 3.3 shows that the objective remains unchanged under the transformations (4.6) and (4.7):

$$z(\mathbf{x}) = \sum_{i=1}^{m} b_i(\pi)x_i + \sum_{i=1}^{m} \sum_{j \ne i} a_{ij}(\pi)x_ix_j$$

for any $\mathbf{x} \in S$.

Define

$$f_i(\pi) = \min\left\{ b_i(\pi) + \sum_{j \ne i} a_{ij}(\pi)x_j \mid x_i{=}1,\ \mathbf{x} \in S \right\} \tag{4.8}$$

for $i = 1, \ldots, m$ and

$$PB(\pi) = \min\left\{ \sum_{i=1}^{m} f_i(\pi)x_i \mid \mathbf{x} \in S \right\}\,. \tag{4.9}$$

Both (4.8) and (4.9) have linear objective functions hence are efficiently solvable by assumption (i).

$PB(\pi)$ is a lower bound on the original problem. Our attention then turns to finding the best parameters π^* such that

$$PB(\pi^*) = \max\left\{ PB(\pi) \mid \pi \in \mathbf{R}^m \right\}\,. \tag{4.10}$$

call (4.10) the *Best-Parameter Search Problem.*

Lemma 4.1. $PB(\pi)$ is a piecewise linear and concave function on the parameters $\pi \in \mathbf{R}^m$.

Proof. Since S is a subset of B_2^m, the number of feasible solutions in S does not exceed 2^m hence S is finite. $b_i(\pi)$ and $a_{ij}(\pi)$ are all linear functions on $\pi \in \mathbf{R}^m$. (4.8) says that $f_i(\pi)$ is the minimum of a finite number of linear functions on π hence it is a piecewise linear and concave function. $PB(\pi)$ in

(4.9) is in turn the minimum of finite number of piecewise linear, concave functions. Therefore, $PB(\pi)$ is also piecewise linear and concave.

A popular method for searching the maximum of a piecewise linear, concave function is the subgradient method (cf. Held and Karp[1970, 1971], Held et al[1974] and Goffin[1977]). Here we state briefly the principle of this method:

A piecewise linear, concave function takes the form

$$g(\pi) = \min\left\{ c_i + \pi\, \mathbf{v}^i \mid i \in \mathbf{I} \right\}, \tag{4.11}$$

where $\mathbf{I}$ is a finite index set, c_i are constants, $\mathbf{v}^i$ are vectors in $\mathbf{R}^m$ with constant components and $\pi\, \mathbf{v}^i$ is the inner product of two vectors. Each $c_i + \pi\, \mathbf{v}^i$ can be viewed as a hyperplane. The problem is to find the best π^* such that

$$g(\pi^*) = \max_{\pi} g(\pi) . \tag{4.12}$$

For any fixed $\pi \in \mathbf{R}^m$, $g(\pi)$ in (4.11) is taken on some hyperplanes:

$$g(\pi) = c_i + \pi\, \mathbf{v}^i , \qquad i \in \mathbf{I}(\pi) , \tag{4.13}$$

where $\mathbf{I}(\pi)$ is a subset of $\mathbf{I}$ depending on π. If $\mathbf{I}(\pi)$ consists of exactly one member, then $g(\pi)$ is differentiable at π and we can search the better parameters along the gradient direction. Otherwise, $g(\pi)$ locates at the intersection of several hyperplanes hence we cannot use any differentiability. The maximizing vector of parameters π^* usually locates at such a nondifferentiable point. In any case, we take one of the hyperplanes from (4.13):

$$g(\pi) = c_k + \pi\, \mathbf{v}^k \tag{4.14}$$

and $\mathbf{v}^k$ can be regarded as a subgradient direction. Take another parameters π^1 along this direction:

$$\pi^1 = \pi + t\, \mathbf{v}^k$$

with a step size t carefully selected. Then $g(\pi)$ might improve at π^1. If the value of the function does not improve after a number of iterations, then we choose a smaller t and search until t is less than a preset number.

For our problem (4.10), we may also employ this subgradient method. The subgradient at the known parameters π can be determined as follows.

Given π, let us solve (4.8) and then (4.9). Let $\mathbf{x}^i = (x_j^i) \in S$ be the solution for $i = 1, 2, \ldots, m$, and $\mathbf{x}^* = (x_j^*) \in S$ is a solution for (4.9). Then

$$\begin{aligned} PB(\pi) &= \sum_{i=1}^{m}\left[b_i(\pi) + \sum_{j\neq i} a_{ij}(\pi)x_j^i\right] x_j^* \\ &= \sum_{i=1}^{m}\left[b_i - (K-1)\pi_i + \sum_{j\neq i} a_{ij}x_j^i + \sum_{j\neq i}\pi_j x_j^i\right] x_i^* \\ &= \left[\sum_{i=1}^{m} b_i x_i^* + \sum_{i=1}^{m}\sum_{j\neq i} a_{ij}x_i^*\right] \\ &\quad + \sum_{j=1}^{m}\left[\sum_{i\neq j} x_j^i x_i^* - (K-1)x_j^*\right]\pi_j . \end{aligned}$$

Compare the above expression with (4.14). One of the subgradient directions at this π is given by a vector $\mathbf{v}^k$ with components

$$v_j^k = \sum_{i=1}^{m} x_j^i x_i^* - (K-1)x_j^* , \quad 1 \leq j \leq m . \tag{4.15}$$

As being shown by Held at al[1974] and Goffin[1977] and others, the subgradient method can finally locate the best parameters approximately by properly choosing a sequence of step size $\{ t_j \}$. However, this method has some drawbacks. Firstly, this method usually takes many iterations to obtain a satisfactory result. Secondly, there is no guarantee that each iteration would improve the value of the function. Thirdly, the number of iterations needed depends to a great degree on the choice of the scales $\{ t_j \}$ which are determined by the nature of the constraints and the size of the problem. As solving problem (4.10) is only an intermediate step in a branch-and-bound procedure, we must look for more efficient ways to search for the best parameters. We will prove from the theoretical point of view that the leveling method is a good candidate.

4.3. A Sufficient Condition for the Best Parameters

As in Chapter 3, we define

$$\begin{cases} f_{\max}(\pi) = \max_i \left\{ f_i(\pi) \right\} , \\ f_{\min}(\pi) = \min_i \left\{ f_i(\pi) \right\} , \end{cases} \tag{4.16}$$

where $f_i(\pi)$, $1 \le i \le m$, are from (4.8). Define

$$\phi(\pi) = f_{\max}(\pi) - f_{\min}(\pi) . \tag{4.17}$$

Call $\phi(\pi)$ the deviation of π . The following theorem shows that $\phi(\pi)$ can be used to characterize the optimality of (4.10).

Theorem 4.1. Given a fixed choice of parameters $\hat{\pi}$, we have

$$PB(\pi) \le PB(\hat{\pi}) + K\phi(\hat{\pi})$$

for any $\pi \in \mathbf{R}^m$, where K is the integer satisfying assumption (ii).

Proof. By definition of $PB(\hat{\pi})$, we have

$$f_{\min}(\hat{\pi}) \sum_{i=1}^{m} x_i \le PB(\hat{\pi}) .$$

Using $\sum_{i=1}^{m} x_i = K$ and $f_{\min}(\hat{\pi}) = f_{\max}(\hat{\pi}) - \phi(\hat{\pi})$ we can write this as

$$Kf_{\max}(\hat{\pi}) \le PB(\hat{\pi}) + K\phi(\hat{\pi}) .$$

So to complete the proof, we need to show that

$$PB(\pi) \leq Kf_{\max}(\hat{\pi})$$

for any $\pi \in \mathbf{R}^m$. To see this, fix π and express it as $\pi = \hat{\pi} + \mathbf{d}$, $\mathbf{d} \in \mathbf{R}^m$. Since (4.6) and (4.7) are linear on π, it can be rewritten as

$$b_i(\pi) = b_i(\hat{\pi}) - (K-1)d_i, \quad 1 \leq i \leq m,$$

$$a_{ij}(\pi) = a_{ij}(\hat{\pi}) + d_j, \quad 1 \leq i,j \leq m, \ i \neq j.$$

Thus we get

$$f_i(\pi) = \min\left\{ b_i(\hat{\pi}) + \sum_{j\neq i} a_{ij}(\hat{\pi})x_j - (K-1)d_i + \sum_{j\neq i} d_j x_j \mid x_i = 1, \ \mathbf{x} \in S \right\}$$

$$= \min\left\{ b_i(\hat{\pi}) + \sum_{j\neq i} a_{ij}(\hat{\pi})x_j - Kd_i + \sum_{j=1}^{m} d_j x_j \mid x_i = 1, \ \mathbf{x} \in S \right\} \tag{4.18}$$

for $1 \leq i \leq m$. Since S is a finite set, there is an $\bar{\mathbf{x}} \in S$ satisfying

$$D = \max\left\{ \sum_{j=1}^{m} d_j x_j \mid \mathbf{x} \in S \right\} = \sum_{j=1}^{m} d_j \bar{x}_j . \tag{4.19}$$

Using D to bound the last summation in (4.18), we obtain

$$f_i(\pi) \leq \min\left\{ b_i(\hat{\pi}) + \sum_{j\neq i} a_{ij}(\hat{\pi})x_j \mid x_i = 1, \ \mathbf{x} \in S \right\} - Kd_i + D$$

$$= f_i(\hat{\pi}) - Kd_i + D$$

$$\leq f_{\max}(\hat{\pi}) - Kd_i + D, \quad 1 \leq i \leq m.$$

By (4.19) we have

$$\sum_{i=1}^{m} f_i(\pi)\bar{x}_i \leq \left[f_{\max}(\hat{\pi}) + D \right] \sum_{i=1}^{m} \bar{x}_i - \sum_{i=1}^{m} d_i \bar{x}_i$$

$$= K\left[f_{\max}(\hat{\pi}) + D \right] - KD = Kf_{\max}(\hat{\pi}) .$$

Observing that

$$PB(\pi) \leq \sum_{i=1}^{m} f_i(\pi) x_i \leq K f_{\max}(\hat{\pi}) ,$$

completes the proof.

Corollary 1. $\phi(\pi^*) = 0$ is a sufficient condition for the best parameters π^*. In another words, if $\phi(\pi^*) = 0$ for some $\pi^* \in \mathbf{R}^m$, then

$$PB(\pi) \leq PB(\pi^*)$$

for any $\pi \in \mathbf{R}^m$.

Corollary 2. Assume that π^* are the best parameters. Given $\epsilon > 0$, if another set of parameters $\hat{\pi}$ has a deviation $\phi(\hat{\pi}) \leq \epsilon / K$, then

$$PB(\hat{\pi}) \geq PB(\pi^*) - \epsilon .$$

In view of the above corollaries, if an algorithm is efficient to level $f_1(\pi)$, $f_2(\pi)$, . . . , $f_m(\pi)$, which come from (4.8), it is also efficient for locating the best parametrized lower bound. This is why we name our method the *leveling method.* We will prove in next section that the algorithm we used for the QMST is such a good algorithm suitable for our class of Quadratic 0-1 Programs.

4.4. Leveling Algorithm and Its Convergence

In this section, we are to prove theoretically the convergence of the leveling algorithm by showing that it does reduce the deviation and also improves the lower bound after each iteration.

When applied to the class of quadratic 0-1 programs considered in Section 3.5, the algorithm needs the minor modification of changing (n-1) to K in steps 4 and 5 there. For ease of reference, we rewrite the algorithm (with option 1) below.

Leveling Algorithm. In this algorithm, $\delta > 0$ is an input parameter governing the termination criterion. K is the integer specified in assumption (ii) on the class of problems under study.

STEP 1. Solve m problems with linear objectives:

$$f_i = \min\left\{ b_i + \sum_{j \neq i} a_{ij} x_j \mid x_i = 1,\ \mathbf{x} \in S \right\} \tag{4.21}$$

for $1 \le i \le m$ and set

$$PB = \min\left\{ \sum_{i=1}^{m} f_i x_i \mid \mathbf{x} \in S \right\} . \tag{4.22}$$

STEP 2. Set $f_{\max} = \max\{f_i\}$, $f_{\min} = \min\{f_i\}$ and $\phi = f_{\max} - f_{\min}$. If $\phi \le \delta$, stop.

STEP 3. Let

$$b_i \leftarrow b_i - \left(1 - \frac{1}{K}\right) f_i, \quad 1 \le i \le m, \tag{4.23}$$

$$a_{ij} \leftarrow a_{ij} + \frac{f_i}{K}, \quad 1 \le i,j \le m, \quad i \neq j . \tag{4.24}$$

return to step 1.

To prove the convergence of this algorithm, we examine two consecutive iterations and trace through changes in the values of the quantities computed in the algorithm. Since the transformation in step 5 is linear in π, without loss of generality, we can start with an iteration for which $\pi = 0$. Let a_{ij}, b_i, f_i, $f_{\max}$, $f_{\min}$, ϕ and PB denote the quantities obtained by the algorithm when $\pi = 0$. Let $\bar{a}_{ij}$, $\bar{b}_i$, $\bar{f}_i$, $\bar{f}_{\max}$, $\bar{f}_{\min}$, $\bar{\phi}$ and $\overline{PB}$ denote the corresponding quantities after one iteration of the algorithm.

Theorem 4.2. Assume $\phi > 0$. After one iteration, there must be

$$f_{\min} \leq \bar{f}_i \leq f_{\max} \tag{4.25}$$

for any i.

Proof. By the updating step in (4.23) and (4.24), we have

$$\bar{b}_i = b_i - (1 - \frac{1}{K}) f_i, \qquad 1 \leq i \leq m,$$

$$\bar{a}_{ij} + \frac{f_j}{K}, \quad 1 \leq i,j \leq m, \quad i \neq j.$$

For a fixed i, denote by $\hat{\mathbf{x}}^i = (\hat{x}_j^i)$ and $\hat{\mathbf{x}} = (\hat{x}_j)$ the solutions to (4.21) and (4.22) when $\pi = 0$. Let $\bar{\mathbf{x}}^i = (\bar{x}_j^i)$ and $\bar{\mathbf{x}} = (\bar{x}_j)$ be the corresponding solutions after one iteration. Then

$$\begin{aligned}\bar{f}_i &= \bar{b}_i + \sum_{j \neq i} \bar{a}_{ij} \bar{x}_j^i \\ &= b_i + \sum_{j \neq i} a_{ij} \bar{x}_j^i - (1 - 1/K) f_i + \frac{1}{K} \sum_{j \neq i} f_j \bar{x}_j^i \\ &= b_i + \sum_{j \neq i} a_{ij} \bar{x}_j^i - f_i + \frac{1}{K} \sum_{j=1}^{m} f_j \bar{x}_j^i .\end{aligned}$$

By the optimality of f_i,

$$b_i + \sum_{j \neq i} a_{ij} \bar{x}_j^i \geq b_i + \sum_{j \neq i} a_{ij} \hat{x}_j^i = f_i . \tag{4.26}$$

Thus

$$\bar{f}_i \geq f_i - f_i + \frac{1}{K} \sum_{j=1}^{m} f_j \bar{x}_j^i = \frac{1}{K} \sum_{j=1}^{m} f_j \bar{x}_j^i \tag{4.27}$$

$$\geq \frac{1}{K} f_{\min} \sum_{i=1}^{m} \bar{x}_j^i = f_{\min} . \tag{4.28}$$

On the other hand, the optimality of $\bar{f}_i$ tells us:

$$\bar{f}_i \leq \bar{b}_i + \sum_{j \neq i} \bar{a}_{ij} \hat{x}_j^i \tag{4.29}$$

$$= b_i + \sum_{j \neq i} a_{ij} \hat{x}_j^i - (1-1/K) f_i + \frac{1}{K} \sum_{j \neq i} f_j \hat{x}_j^i$$

$$= f_i - f_i + \frac{1}{K}\sum_{j=1}^{m} f_j \hat{x}_j^i \tag{4.30}$$

$$\leq \frac{1}{K} f_{\max} \sum_{j=1}^{m} \hat{x}_j^i \tag{4.31}$$

$$= f_{\max} \,. \tag{4.32}$$

Combining (4.28) and (4.32) results in (4.25) and completes the proof.

Comments. Let us review our argument in the proof and find out what are the conditions for (4.25) to hold as an equality.

Let π and $\overline{\pi}$ denote the parameters in two consecutive iterations refered to below as the old and new parameters respectively.

For a fixed i, if $\overline{f}_i = f_{\min}$, then the equalities in (4.26) and (4.28) must all hold. These can be translated into:

(i) $\overline{\mathbf{x}}^i$ is also optimal for (4.21) with the old parameters, that is, we may take $\hat{\mathbf{x}}^i = \overline{\mathbf{x}}^i$.

(ii) $f_j \equiv f_{\min}$ for all j such that $\overline{x}_j^i = 1$, or, in view of (i), for all j such that $\hat{x}_j^i = 1$.

If $\overline{f}_i = f_{\max}$, then the equalities in (4.29) and (4.31) must all hold. These translate into:

(iii) $\hat{\mathbf{x}}^i$ is also optimal for (4.21) with the new parameters, that is, we may take $\overline{\mathbf{x}}^i = \hat{\mathbf{x}}^i$.

(iv) $f_j \equiv f_{\max}$ for all j such that $\hat{x}_j^i = 1$.

Conditions (i) - (iv) are rather restrictive. Since we assume that $\phi > 0$, that is, f_j, $1 \leq j \leq m$, are different, (ii) and (iv) would rarely be satisfied .

Corollary. The leveling algorithm forces the deviation ϕ to decrease whenever $\phi \neq 0$.

Theorem 4.3. Assume $\phi \geq 0$. After one iteration, there must be

$$\overline{PB} \geq PB .$$

Proof. Employing the inequality (4.27) in the proof of theorem 4.2, we obtain for each i

$$\bar{f}_i \geq \frac{1}{K}\sum_{j=1}^{m} f_j \, \bar{x}_j^i \geq \frac{1}{K}\sum_{j=1}^{m} f_j \, \hat{x}_j^i = PB/K . \tag{4.33}$$

Thus,

$$\overline{PB} = \sum_{i=1}^{m} \bar{f}_i \, \bar{x}_i \geq \frac{1}{K} PB \sum_{i=1}^{m} \bar{x}_i = PB . \tag{4.34}$$

The equality in (4.34) holds only when condition (1) hold for all i such that $\bar{x}_i = 1$.

CHAPTER 5

ON THE LOWER BOUND OF THE QAP

The Quadratic Assignment Problem (QAP) belongs to the class of quadratic 0-1 programming problems specified in Chapter 4. In this chapter, we utilize the leveling method to provide lower bounds on this problem and compare our method with other bounding techniques in the literature.

5.1. Existing Bounding Techniques

Since the availability of good lower bounds forms an important component of implicit enumeration methods for the QAP, many papers have been devoted to obtaining lower bounds for the problem, especially for the Koopmans-Beckmann form.

Background on the Quadratic Assignment Problem and the Koopmans-Beckmann Problem can be found in Appendix C. After separating the linear terms, the Quadratic Assignment Problem has the following form:

$$\text{(QAP)} \qquad \text{Minimize} \quad \left\{ \sum_{i=1}^{n}\sum_{p=1}^{n} b_{ip}x_{ip} + \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} a_{ipjq}x_{ip}x_{jq} \right\} \tag{5.1}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} x_{ij} = 1 , \qquad 1 \le j \le n , \tag{5.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 , \qquad 1 \le i \le n , \tag{5.3}$$

$$x_{ij} = 0 \text{ or } 1 , \qquad 1 \le i,j \le n . \tag{5.4}$$

For notational convenience, we assume that $a_{ipjq} = 0$ whenever $i = j$ or $p = q$. The Koopmans-Beckmann Problem is

(KBP) Minimize $\left\{ \sum_{i=1}^{n}\sum_{p=1}^{n} b_{ip}x_{ip} + \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} c_{ij}d_{pq}x_{ip}x_{jq} \right\}$ (5.5)

subject to: (5.2) – (5.4) .

The Gilmore-Lawler bound (*GLB*):

For a fixed pair (i,p), $i,p = 1, \ldots, n$, let

$$f_{ip} = \min \sum_{j=1}^{n}\sum_{q=1}^{n} a_{ipjq}y_{jq} \tag{5.6}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} y_{jq} = 1 , \qquad 1 \le q \le n , \tag{5.7}$$

$$\sum_{q=1}^{n} y_{jq} = 1 , \qquad 1 \le j \le n , \tag{5.8}$$

$$y_{jq} = 0 \text{ or } 1, \quad 1 \le j,q \le n , \tag{5.9}$$

$$y_{ip} = 1 . \tag{5.10}$$

Then the expression (5.1) is bounded by

$$GLB = \min \sum_{i=1}^{n}\sum_{p=1}^{n} f_{ip}x_{ip} \tag{5.11}$$

s.t. (5.2) – (4.4) .

For the Koopmans-Beckmann Problem, (5.6) assumes the following form:

$$f_{ip} = \min \left\{ b_{ip} + \sum_{j=1}^{n}\sum_{q=1}^{n} c_{ij}d_{pq}y_{jq} \right\} . \tag{5.12}$$

with $c_{ii} = d_{pp} = 0$ $(1 \le i,p \le n)$. Gilmore[1962] and Lawler[1962] proposed to solve (5.12) by using the minimal scalar product: View c_{ij} , $1 \le j \le n$, $j \ne i$, and d_{pq} , $1 \le q \le n$, $p \ne q$, as two vectors with (n-1) components: $\mathbf{u} = (u_1, \ldots, u_{n-1})$ and $\mathbf{v} = (v_1, \ldots, v_{n-1})$. (5.12) is to find a permutation $\phi(i)$ of the index set $\{ 1, 2, \ldots, n-1 \}$ so as to minimize the scalar product:

$$\sum_{i=1}^{n-1} u_i v_{\phi(i)} .$$

Such a minimal scalar product is determined by arranging the elements of vector $\mathbf{u}$ in nondecreasing order and the elements of $\mathbf{v}$ in nonincreasing order. After evaluating (5.12), a lower bound is obtained by solving (5.11).

Recently, several researchers have proposed some modifications of the Gilmore-Lawler bound. For example, Burkard and Stratmann[1978], Roucairo[1979] and Edwards[1980] propose row and/or column reductions on matrices $\mathbf{C} = \left[c_{ij}\right]$ and $\mathbf{D} = \left[d_{pq}\right]$. However, these reductions may or may not result in improved lower bound depending on the problem data, that is, the matrices $\mathbf{C}$ and $\mathbf{D}$. Thus, none of these methods can be proved to dominate any of others.

Finding the Gilmore-Lawler bound for a Koopmans-Beckmann Problem needs $O(n^3 \log n)$ computations. For a general QAP, the complexity increases to $O(n^5)$.

The Christofides bound (*CHB*):

Christofides, Mingozzi and Toth[1980] proposed a bounding technique for the Koopmans-Beckmann Problem. Their algorithm consists of two phases.

The first phase involves a series of reductions as follows:

(1) Row (and column) reduction on matrices $\mathbf{C}$ and $\mathbf{D}$. Let

$$\begin{aligned} c_{ij} &= \bar{c}_{ij} + \alpha_i , \quad 1 \leq i,j \leq n , \quad i \neq j , \\ d_{pq} &= \bar{d}_{pq} + \beta_p , \quad 1 \leq p,q \leq n , \quad p \neq q , \\ \bar{c}_{ii} &= \bar{d}_{pp} = 0 , \quad 1 \leq i,p \leq n . \end{aligned} \tag{5.13}$$

where

$$\alpha_i = \min_{j \neq i}\left\{c_{ij}\right\}, \quad \beta_p = \min_{q \neq p}\{d_{pq}\}, \quad 1 \leq i,p \leq n .$$

Then let

$$\bar{b}_{ip} = b_{ip} + \alpha_i \sum_{q=1}^{n} \bar{d}_{pq} + \beta_p \sum_{j=1}^{n} \bar{c}_{ij} + (n-1)\alpha_i\beta_p , \tag{5.14}$$

for $1 \leq i,p \leq n$. Let $\mathbf{X} = \left[x_{ip}\right]$ be any feasible solution satisfying (4.2) - (4.4). Then we have

$$\sum_{i=1}^{n}\sum_{p=1}^{n} \bar{b}_{ip}x_{ip} + \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} \bar{c}_{ij}\bar{d}_{pq}x_{ip}x_{jq}$$
$$= \sum_{i=1}^{n}\sum_{p=1}^{n} b_{ip}x_{ip} + \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} c_{ij}d_{pq}x_{ip}x_{jq} = z(\mathbf{X}) . \tag{5.15}$$

Therefore, the value of the objective function (5.5) remains unchanged under row reductions (5.13) and (5.14).

Column reduction can be performed on the row-reduced quadratic terms in the similar way.

(ii) Apply the Hungarian algorithm (Kuhn[1955]) to the linear terms in (5.15), i.e. minimize $\sum_{i=1}^{n}\sum_{j=1}^{n} \bar{b}_{ij}x_{ij}$ subject to constraints (4.2) to (4.4). Let V be the minimum. Then

$$\sum_{i=1}^{n}\sum_{p=1}^{n} \bar{b}_{ip}x_{ip} = V + \sum_{i=1}^{n}\sum_{p=1}^{n} \hat{b}_{ip}x_{ip} \tag{5.16}$$

for any feasible $\mathbf{X} = \left[x_{ip}\right]$. $\hat{\mathbf{B}} = \left[\hat{b}_{ip}\right]$ is the reduced matrix.

(iii) For $1 \leq i,p \leq n$, find the (row) minimal scalar product r_{ip} such that

$$r_{ip} = \min\left\{\sum_{j=1}^{n}\sum_{q=1}^{n} \bar{c}_{ij}\bar{d}_{pq}y_{jq}\right\}$$

subject to (5.7) - (5.10). Then

$$\sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} \overline{c}_{ij}\overline{d}_{pq}x_{ip}x_{jq} \geq \sum_{i=1}^{n}\sum_{p=1}^{n} r_{ip}x_{ip}$$

for any feasible $\mathbf{X}$.

Similarly, find the (column) minimal scalar product $\overline{r}_{ip}$ for $1 \leq j,q \leq n$ such that

$$\overline{r}_{ip} = \min\left\{ \sum_{j=1}^{n}\sum_{q=1}^{n} \overline{c}_{ji}\overline{d}_{qp}y_{jq} \right\}$$

subject to constraint (5.7) - (5.10). Now let

$$t_{ip} = (r_{ip} + \overline{r}_{ip})/2 \; ,$$
$$l_{ip} = \hat{b}_{ip} + t_{ip} \, .$$

Then the objective value in (5.15) will satisfy

$$\begin{aligned} z(\mathbf{X}) &\geq V + \sum_{i=1}^{n}\sum_{p=1}^{n} (\hat{b}_{ip} + t_{ip})x_{ip} \\ &= V + \sum_{i=1}^{n}\sum_{p=1}^{n} l_{ip}x_{ip} \, . \end{aligned} \tag{5.17}$$

(iv) The above linear sum can be reduced further by applying the Hungarian algorithm:

$$\sum_{i=1}^{n}\sum_{p=1}^{n} l_{ip}x_{ip} = AP(L) + \sum_{i=1}^{n}\sum_{p=1}^{n} \overline{l}_{ip}x_{ip} \, .$$

Let $CHB = V + AP(L)$, (7.20) reduces to

$$z(\mathbf{X}) \geq CHB + \sum_{i=1}^{n}\sum_{p=1}^{n} \overline{l}_{ip}x_{ip} \, . \tag{5.18}$$

The authors claim that the bound denoted by CHB in (5.18) "is quite clearly superior in quality to other existing bounds".

The second phase is to examine the reduced matrix $L = (l_{ip})$ and $\overline{L} = (\overline{l}_{ip})$ in (5.17) and (5.18) respectively and try to fix a partial assignment. Set up an (n block) by (n block) matrix $S = \left[s_{ip} \right]$ where block (i,p) corresponds to assigning i to p. Let $s_{ip}(j,q)$ denote the element at the j-th row and q-th column in block (i,p), which corresponds to assigning j to q (in addition to assigning i to p). Initially, all element are set to one. Assume that an upper bound denoted by $\hat{z}$ is available. Such a bound may be obtained by a heuristic algorithm providing a feasible solution. The difference ($\hat{z} - CHB$) indicates the gap between this upper bound and the current lower bound. Examine (5.17) and (5.18) carefully according the following criteria:

a) If

$$\overline{l}_{ip} \geq \hat{z} - CHB \tag{5.19}$$

then set all element in block s_{ip} to zero.

b) If

$$V + l_{ip} + \sum_{k \neq i,j} \min_{r \neq p,q} \left[l_{kr} \right] \geq \hat{z} \tag{5.20}$$

or

$$CHB + \max \left[\overline{l}_{ip}, \overline{l}_{jq} \right] \geq \hat{z}\ , \tag{5.21}$$

then set element $s_{ip}(j,q)$ to zero.

Observe the block matrix S. If in any block-row or block-column all blocks are zero, infeasibility is implied. If there is only one non-zero block (say s_{ip}), then the assignment of i to p is implied. If for any row or column of S all entries except one (say $s_{ip}(j,q)$) are zero, then the twin assignments of i to p and j to q are implied. In this way, a partial assignment may be fixed and the dimension of problem may be reduced. For the reduced problem, repeat the procedure which lead to further fixing some other assignments. Thus, the lower bound will be improved further, as the authors claimed.

In their paper, the authors applied their method to an example from Gavett and Plyter[1966] which has dimension 4. They chose $\hat{z} = 816$ as the known upper bound. After two iterations of the preceding procedure, all four pairs of assignment are fixed and the optimal value (=806) comes out. However, the authors do not discuss further application to other examples.

Remarks.

(1) The criteria listed in (5.19) - (5.21) can be strengthened to

$$\overline{l}_{ip} + \sum_{j \neq i} \min_{q \neq p} \left[\overline{l}_{jq} \right] \geq \hat{z} - CHB , \tag{5.19'}$$

$$V + l_{ip} + l_{jq} + \sum_{k \neq i,j} \min_{r \neq p,q} \left[l_{kr} \right] \geq \hat{z} , \tag{5.20'}$$

$$CHB + \overline{l}_{ip} + \overline{l}_{jq} + \sum_{k \neq i,j} \min_{r \neq p,q} \left[l_{kr} \right] \geq \hat{z} , \tag{5.21'}$$

Since all the above violate (5.17) and (5.18).

(2) As we pointed out before, no single reduction method is provably superior to others.can be proved Therefore, the authors' claim that CHB in (5.18) is superior in quality to other existing bounds can only be validated empirically. In our computational experience, described in Section 5.4, the Christofides bound after the second phase performs no better than the Gilmore-Lawler bound without any reductions. We will give some examples for which the Christofides bounds in (5.18) are even worse than the Gilmore-Lawler bounds.

Frieze-Yadegar bound (***FYB*** **):**

Frieze and Yadegar[1983] proposed a Lagrangean relaxation approach to evaluate the lower bound of the general Quadratic Assignment Problem. They introduce $2n^3$ multipliers α_{pjq} and β_{pjq}, $1 \leq p,j,q \leq n$, and prove that problem (5.1) can be relaxed to the following Lagrangean problem:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{p=1}^{n} \bar{b}_{ip}x_{ip} + \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} \bar{a}_{ipjq}y_{ipjq} \tag{5.22}$$

subject to: (5.2)–(5.4)

$$\sum_{j=1}^{n} y_{ipjq} = x_{ip}, \qquad 1 \le i,p,q \le n, \tag{5.23}$$

$$\sum_{q=1}^{n} y_{ipjq} = x_{ip}, \qquad 1 \le i,j,p \le n, \tag{5.24}$$

$$y_{ipip} = x_{ip}, \qquad 1 \le i,p \le n, \tag{5.25}$$

$$0 \le y_{ipjq} \le 1, \qquad 1 \le i,p,j,q \le n, \tag{5.26}$$

where

$$\bar{a}_{ipjq} = 0, \qquad \text{whenever } i=j \text{ or } p=q,$$

$$\bar{a}_{ipjq} = a_{ipjq} - \alpha_{pjq} - \beta_{ijq}, \qquad 1 \le i,p,j,q \le n, \tag{5.27}$$

$$\bar{b}_{ip} = b_{ip} + \sum_{q=1}^{n} \alpha_{qip} + \sum_{j=1}^{n} \beta_{jip}, \qquad 1 \le i,p \le n.$$

The solution of the Lagrangean relaxation problem for any fixed multipliers α and β, denoted by $L(\alpha,\beta)$, is a lower bound on the original Quadratic Assignment Problem. Furthermore, they proved that $L(\alpha,\beta)$ is equal to the Gilmore-Lawler bound of a new Quadratic Assignment Problem:

$$\min\left\{ z(X) = \sum_{i=1}^{n}\sum_{p=1}^{n} \bar{b}_{ip}x_{ip} + \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} \bar{a}_{ipjq}x_{ip}x_{jq} \right\} \tag{5.28}$$

subject to (5.2) -- (5.4), where $\bar{b}_{ip}$ and $\bar{a}_{ipjq}$ are obtained by (5.27).

The search for the best multipliers is performed using the subgradient method. The iterative algorithm converges to the best multipliers α^* and β^* such that

$$L(\alpha^*,\beta^*) = \max_{\alpha,\beta} L(\alpha,\beta) . \tag{5.29}$$

This is the Frieze-Yadegar bound (FYB).

The authors applied this technique to evaluate the lower bounds for some examples and obtained very sharp lower bounds. Unfortunately, the subgradient method takes too many iterations to come close to the best multipliers. They were not able to calculate the lower bounds for problems with dimension greater than 9 and then placed their hopes in using a powerful parallel processing computer like the I. C. L. Distributed Array Processor. The authors also modified the above algorithm so that the minimum scalar products can be used if applied to problems of Koopmans-Beckmann type. As seen from their computational result, this variant is still very time-consuming. In our discussion later, we denote their lower bounds by FYB1 and FYB2 respectively.

5.2. Application of the Leveling Method to the QAP"

Denote by A the collection of all $\mathbf{X} \in B_2^{n^2}$ that are feasible to (5.2) -- (5.4). Then the problem can be rewritten as

$$\min \left\{ \sum_{i=1}^{n} \sum_{p=1}^{n} b_{ip} x_{ip} + \sum_{i=1}^{n} \sum_{p=1}^{n} \sum_{j=1}^{n} \sum_{q=1}^{n} a_{ipjq} x_{ip} x_{jq} \mid \mathbf{X} \in A \right\} . \tag{5.30}$$

Again, we assume that $a_{ipjq} = 0$ whenever $i=j$ *or* $p=q$.

Notice that (5.2) and (5.3) imply the relation

$$\sum_{i=1}^{n} \sum_{p=1}^{n} x_{ip} = n . \tag{5.31}$$

And, the linear assignment problem can be solved efficiently (see Appendix C). Therefore, the Quadratic Assignment Problem satisfies assumptions (i) and (ii) in Section 4.1 with $K = n$ hence belongs to the class of problems which the leveling method applies.

Starting with any parameter set $\pi = (\pi_{ip})$, $1 \leq i,p \leq n$, let

$$\begin{aligned} b_{ip}(\pi) &= b_{ip} - (n-1)\pi_{ip} \,, && 1 \leq i,p \leq n \,, \\ a_{ipjq}(\pi) &= a_{ipjq} + \pi_{jq} \,, && 1 \leq i,j,p,q \leq n \,, \\ a_{ipjq}(\pi) &= 0 \,, && i=j \text{ or } p=q \,. \end{aligned} \tag{5.32}$$

Solve n^2 linear assignment problems

$$f_{ip}(\pi) = \min \left\{ b_{ip}(\pi) + \sum_{j=1}^{n} \sum_{q=1}^{n} a_{ipjq}(\pi) x_{jq} \mid x_{ip}=1, \ \mathbf{X} \in A \right\} \tag{5.33}$$

for $1 \leq i,p \leq n$ and let

$$PB(\pi) = \min \left\{ \sum_{i=1}^{n} \sum_{p=1}^{n} f_{ip}(\pi) x_{ip} \mid \mathbf{X} \in A \right\} \,. \tag{5.34}$$

The next parameter set will be chosen by

$$\overline{\pi}_{ip} = \pi_{ip} + f_{ip}(\pi)/n \,, \qquad 1 \leq i,p \leq n \,. \tag{5.35}$$

The above algorithm iteratively arrives at the best parameters π^* such that

$$PB(\pi^*) = \max_{\pi} \left\{ PB(\pi) \right\} \,. \tag{5.36}$$

Denote this bound by the *Leveling Bound* (*LVB*).

Since the value of $PB(\pi)$ in (5.34) equals the Gilmore-Lawler bound when $\pi = 0$, we can immediately conclude that *LVB* dominates *GLB* in quality.

Let us compare *LVB* with *FYB* . When the multipliers in (5.27) are restricted in a $2n^2$ -dimensional subset of $\mathbf{R}^{2n^3}$ such that

$$\alpha_{pjq} = \overline{\alpha}_{jq} \,, \qquad \beta_{pjq} = \overline{\beta}_{jq} \,, \qquad 1 \leq p,j,q \leq n \,,$$

(5.27) takes the same form as (5.32) if we denote $\pi_{jq} = -(\overline{\alpha}_{jq} + \overline{\beta}_{jq})$. So *FYB* dominates *LVB* and we may view the leveling method as an efficient method to search for the "locally-optimal" multipliers for problem (5.29) in the sense of restricting the multipliers to a subset in $\mathbf{R}^{2n^3}$.

Our computational experiments, which will be described in the next section, show that the leveling algorithm reduces the deviation very efficiently. To get some feeling of how fast the leveling algorithm converges, we show below an output for a Koopmans-Beckmann Problem with n=7 which is taken from Lawler[1962] and an output for an general Quadratic Assignment Problem whose construction will be introduced in Section 5.3.

# ITER	PB	ϕ	CPU(s)
1	499.00	160.00	.144
2	527.29	21.29	.276
3	533.61	6.63	.409
4	537.92	2.17	.541
5	539.06	.95	.672
6	539.85	.42	.804
7	540.22	.20	.936
8	540.36	.10	1.068
9	540.44	.05	1.200
10	540.48	.02	1.333
11	540.48	.01	1.466
12	540.48	.00	1.600

Table 5-1. Result for problem Lawler7.

# ITER	PB	ϕ	CPU(s)
1	1322.00	175.00	1.553
2	1784.00	39.07	3.105
3	1900.07	11.49	4.645
4	1930.54	4.22	6.182
5	1941.00	1.22	7.710
6	1944.06	.37	9.241
7	1945.12	.13	10.775
8	1945.39	.04	12.309
9	1945.48	.01	13.843
10	1945.51	.00	15.380
11	1945.52	.00	16.926

Table 5-2. Result for a QAP with n=14.

We can see that the deviation decreases to zero very quickly and the lower bound PB gets improved with every iteration. Considering that all data in the given matrices are all integers, the lower bound must be integers also. Therefore, the leveling method actually arrives the best parametrized lower bounds (541 and 1946 respectively) at the seventh iteration in both examples. All outputs for problems of different size look quite the same.

5.3. Comparisons

In order to compare the leveling method with those existing techniques, we have coded both the leveling algorithm and the algorithm due to Christofides et al. We have run these codes on several Koopmans-Beckmann Problems and general Quadratic Assignment Problems.

In the first part of our experiments, the data for the test problems are extracted from works of Nugent, Vollmann and Ruhl[1968], Gavett and Plyter[1966] and Lawler[1962]. All are in the Koopmans-Beckmann form. We refer to each problem by its source and its dimension. Thus, problem "Nugent20" is from Nugent et al[1968]and has dimension n=20. For convenience to refer, we show these problem data in Appendix D.

The leveling algorithm starts with $\pi = 0$ that corresponds to the Gilmore-Lawler bound. To obtain the Christofides bound, we chose the best-known objective values (all of them except for Nugent 20 are known to be optimal) as $\hat{z}$ so that we can observe the effects of the second phase in their algorithm.

Table 5-3 compares the three bounds (*GLB*, *CHB* and *LVB*). In the column "OTHER", "a" indicates the lower bound obtained by Roucairo[1979] and "b" by Edwards[1980]. " *BV* " is the best-known objective value for the corresponding problem. The values with (*) are known to be optimal.

Problem	n	GLB	CHB	OTHER	LVB	BV
Gavett	4	792	806	806(a)	804	806*
Lawler	7	499	472	505(b)	541	559*
Nugent	5	50	50		50	50*
Nugent	6	82	82		82	86*
Nugent	7	137	137		139	148*
Nugent	8	186	186		188	214*
Nugent	10	493	493		495	578*
Nugent	15	963	963		968	1150*
Nugent	20	2057	2057		2071	2570

Table 5-3. Comparison of several bounds.
* indicates optimal value.

We see from Table 5-3 that the LVB is better than other bounds in quality for all but one problem. The Christofides' algorithm works well for Gavett 4 but fails to fix a partial assignment for the other test problems. This is not surprising because fixing an assignment of i to p requires to exclude all assignments of j to p , $j \neq i$, or to exclude all assignments of i to q , $q \neq p$, according to criteria (5.19) -- (5.21). This strict condition happens only to a problem with small dimension and unusual data. For the above test problems, when $n=5$ or 6, not a single assignment can be determined even though several assignments have been excluded. When $n>$ 6, the elements in the reduced matrix $L = \left[\bar{l}_{ip}\right]$ are so small compared to the gap between the current lower bound and the best-known objective value $\hat{z}$ that not a single assignment can be excluded. Thus, the Christofides bound is essentially no better than the Gilmore-Lawler bound without any reductions.

Now let us compare the leveling bound with the Frieze-Yadegar bound. Table 5-4 shows these two lower bounds for the same problems as those in Table 5-3. The data for the latter are from Frieze and Yadegar[1983]. In this Table, $FYB1$ and $FYB2$ are two versions of Frieze-Yadegar lower bounds. NIT1 and NIT2 are the numbers of iterations needed to reach $FYB1$ and $FYB2$ respectively. Each iteration requires calculating $2n^3$ components of a subgradient direction plus solving $n^2 + 1$ linear assignment problems. NIT is the number of iterations needed to reach the LVB. Each iteration needs solving $n^2 + 1$ linear assignment problems but no extra effort in evaluating any other parameters. Hence the computational time needed in each iteration for the LVB is far less than that for the Frieze-Yaegar bound.

From Table 5-4, we can see that LVB is worse than $FYB1$ but better than $FYB2$ in quality. As for the computational effort, to obtain the Frieze-Yadegar bound needs hundreds of iterations while computing LVB needs only a few.

PROBLEM	$FYB1$	$FYB2$	LVB	NIT1	NIT2	NIT
Gavett4	806*	806*	804	2	1	11
Lawler7	559*	511	541	23	50	7
Nugent5	50*	50*	50*	1	1	1
Nugent6	86*	82	82	166	1	1
Nugent7	148*	138	139	376	30	3
Nugent8	194	187	188	411	20	3
Nugent12	---	494	495	--	350	3
Nugent15	---	963@	968	--	1	4
Nugent20	---	2057@	2071	--	1	4

Table 5-4. Comparison between LVB and FYB.
*) indicates the optimal value.
@) is the same with GLB

Both versions of the Frieze-Yadegar bounding algorithm can only evaluate lower bounds for small problems. Though the $FYB2$ utilizes the scalar product to save computational effort, it still fails to evaluate lower bounds for problems with $n > 12$. Those bounds for $n = 15$ and $n = 20$ are actually the Gilmore-Lawler bounds.

In the second part of our experiments, we try to examine the effect of row and column reductions. We have constructed several problems of Koopmans-

Beckmann type and run our codes on them. The problem data are shown in Appendix D. Since we have not yet coded a heuristic algorithm for the Koopmans-Beckmann problem, we do not have a reasonable upper bound $\hat{z}$ to test the second phase in the Christofides algorithm. Thus we only compute the *CHB* in (5.18). We test all possible combinations of reductions and scalar products, e.g. row reduction on matrices C and D followed by (column) scalar product, or row reduction followed by all (row and column) scalar products, or all (row and column) reductions followed by (row) scalar product, etc. Since it is proved by Frieze and Yadegar[1983] that row reduction followed by (row) scalar product or column reduction followed by (column) scalar product would provide the same bound as the Gilmore-Lawler bound, we do not list this two combinations. The first column in Table 5-5 corresponds to the Gilmore-Lawler bound. The best lower bound among the last four columns is chosen as the *CHB* in (5.18). It is marked by "#" in Table 5-5 and shown in the column " *CHB* " in Table 5-6.

Reduction(s)		No	Col	Row	All	All	All
Scalar Prod.		Row	Row	Col	Row	Col	All
Problem	n						
1	5	2151	2121	2121	2138#	2138#	2138#
2	6	1114	1114	1114	1110	1110	1115#
3	7	598	590#	590#	589	589	590#
4	8	1132	1045	1045	1141#	1141#	1097
5	10	1265	1205	1205	1250#	1250#	1230
6	12	2009	1974	1974	1984#	1984#	1980

Table 5-5. Effect of reductions (1).
# is taken to be the *CHB* in (5.18).

We also try to study the effect of reduction on the leveling method. For this purpose, we perform all (row and column) reductions on matrices C and D before running the leveling algorithm. The resulting bounds are shown in Table

5-6 comparing with the Gilmore-Lawler bound, the *CHB* obtained by preceding experiment and the *LVB* without reduction.

Problem	n	*GLB*	*CHB*	*LVB*	*LVB* with reductions
1	5	2151	2138	2157*	2155
2	6	1114	1115	1130	1139*
3	7	598	590	609	612*
4	8	1132	1141	1172	1184*
5	10	1265	1250	1281*	1272
6	12	2009	1984	2020*	2011

Table 5-6. Effect of reductions(2).

The best lower bound for each problem is marked by a $*$. We can see that the Gilmore-Lawler bounds are better than the *CHB* in (5.18) for problems 1, 3, 5 and 6. Also, whether reduction benefits the *LVB* depends on the problem data.

Finally, we apply the leveling algorithm to some general Quadratic Assignment Problems (not the Koopmans-Beckmann type). Since no test problems for general QAP's are available in literature, we construct some ourselves. In our test problems, the elements in the cost matrix $A = \left[a_{ipjq} \right]$ are all random integer numbers distributed uniformlly over $\{ 1, 2, \ldots, 100 \}$. For each dimension, 50 different cost matrices are generated, starting with different seeds. Table 5-7 gives the results of this computational experiment. Since, of the bounds we have discussed, only the Gilmore-Lawler bound is applicable to the general Quadratic Assignment Problem, we compare the *LVB* with the *GLB* for each problem by showing the ratio LVB/GLB , where the *LVB* is the parametrized lower bound when the deviation ϕ becomes less than 1. We also show the ratio $PB1/GLB$, where $PB1$ is obtained by applying the leveling method only once, i.e. the parametrized lower bound at the second iteration in the leveling algorithm. The

CPU(s) is the CPU time needed to reach the LVB. Both mean value and standard deviation of 50 samples are shown in Table 5-7 for each item.

	LVB/GLB		$PB1/GLB$		NIT		CPU(s)	
n	μ	σ	μ	σ	μ	σ	μ	σ
4	1.340	.142	1.215	.117	8.900	1.488	.087	.014
6	1.450	.127	1.295	.099	7.300	.763	.341	.037
8	1.418	.071	1.290	.064	6.520	.505	.979	.076
10	1.433	.068	1.307	.059	5.980	.141	2.254	.060
12	1.410	.041	1.298	.037	5.780	.418	4.672	.345

Table 5-7. A statistical comparison of the LVB and the GLB.
# of samples for each n : 50.

We see from Table 5-7 that the LVB is much better (40% larger) than the GLB. The $PB1$ is about 30% larger than the GLB. It is interesting to notice that the number of iterations required by the leveling algorithm actually decreases as the problem size grows. Moreover, the computational time needed to evaluate the $PB1$ is just twice the time needed to evaluate the GLB. These favorable characteristics point to the efficiency of the leveling bounding technique.

CHAPTER 6
SOLUTION TECHNIQUES FOR THE QMST

In this chapter, branch-and-bound algorithms for solving the QMST optimally and heuristic algorithms for finding approximate solutions are developed and compared. Computational experience with the algorithms is also reported.

6.1. General Description

The branch-and-bound algorithms developed here are all depth-first implicit enumeration algorithm. In the search tree, each branching node stands for a feasible subset in which a partial forest has been fixed. That is, a number of edges have been selected as part of the spanning tree while some other edges have been excluded. When searching on a node, say subset S, a lower bound (LB) is evaluated. The best value known for the upper bound is denoted by CUB and corresponds to a solution (a spanning tree). If $LB \geq CUB$, we say that the node is fathomed and we backtrack to another branch of the tree. If $LB < CUB$, an edge is chosen according to a certain branching rule and is used to divide the current subset S into two separated feasible subsets S_1 and S_2 (see Figure 6-1).

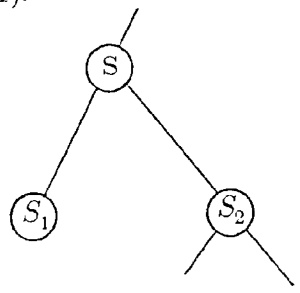

Figure 6-1. Branching nodes.

The subset S_1 includes the edge used for branching while S_2 excludes it. The search will proceed with the node of the tree corresponding to S_1, leaving S_2 aside temporarily. When all nodes in the search tree are fathomed, the branch-and-bound procedure terminates.

Let $\mathbf{M}$ denote the index set $\{1, 2, \ldots, m\}$. Suppose a node is being searched. At this node, a number of variables have been fixed:

$$\begin{cases} x_i = 1 & \text{when } i \in \mathbf{U}, \\ x_j = 0 & \text{when } j \in \mathbf{V}, \end{cases} \tag{6.1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are index subset of $\mathbf{M}$. Let $\mathbf{F} = \mathbf{M} - \mathbf{U} \bigcup \mathbf{V}$ denote the index set of the "free" edges. Then the objective value can be expressed as

$$\begin{aligned} z(\mathrm{x}) &= \left[\sum_{i \in \mathrm{U}} b_i + \sum_{i \in \mathrm{U}} \sum_{\substack{j \in \mathrm{U} \\ j \neq i}} a_{ij} \right. \\ &\quad + \sum_{i \in \mathrm{F}} \left[b_i + \sum_{j \in \mathrm{V}} (a_{ij} + a_{ji}) \right] x_i \\ &\quad + \sum_{i \in \mathrm{F}} \sum_{\substack{j \in \mathrm{F} \\ j \neq i}} a_{ij} x_i x_j \\ &= PV + \sum_{i \in \mathrm{F}} \bar{b}_i x_i + \sum_{i \in \mathrm{F}} \sum_{\substack{j \in \mathrm{F} \\ j \neq i}} a_{ij} x_i x_j \,. \end{aligned} \tag{6.2}$$

We define the partial value PV and new coefficients $\bar{b}_i$ in an obvious fashion as dictated by the equality in (6.2). Let $n_1 = (n-1) - |\mathrm{U}|$. Then we have

$$\sum_{i \in \mathrm{F}} x_i = n_1 \,. \tag{6.3}$$

So, both decomposition bounding technique and parametrized bounding technique may be applied to the last two terms in (6.2) to evaluate a lower bound for the current node.

6.2. Branch-and-Bound Algorithm I (with decomposition bounds)

Bounding:

As in Section 3.3, we can evaluate for the current nodes

$$f_i = \min \left\{ \bar{b}_i + \sum_{\substack{j \in \mathbf{F} \\ j \neq i}} a_{ij} x_j \;\middle|\; \begin{array}{l} x_i = 1,\ \mathbf{x} \in T(G) \\ \text{and satisfies (6.1)} \end{array} \right\}, \tag{6.4}$$

for $i \in \mathbf{F}$, and

$$DB = PV + \min \left\{ \sum_{i \in \mathbf{F}} f_i x_i \;\middle|\; \mathbf{x} \in T(G),\ \mathbf{x} \text{ satisfies } (6.1) \right\}. \tag{6.5}$$

DB is the decomposition bound for the current node.

We employ Kruskal's algorithm to solve (6.4) and (6.5). The solution of (6.5) gives us a spanning tree whose objective value UB is an upper bound. The CUB is updated if $UB < CUB$.

Branching:

Let

$$f_k = \min_{i \in \mathbf{F}} \left\{ f_i \right\}$$

after evaluating (6.4). Take edge e_k as the branching edge.

Computational considerations:

When using Kruskal's algorithm, sorting the edge costs takes $O(m \log m)$ computations while the remaining steps require $O(n^2)$. We presort the rows of the intercost matrix $A = \left[a_{ij} \right]$ at the beginning of the branch-and-bound procedure to save considerable computational effort.

Example 6.1. Consider a simple network shown in Figure 6-2 with inter-cost matrix:

	1	2	3	4	5	6
1	13	12	15	10	4	8
2	12	1	14	16	8	11
3	15	14	6	6	9	7
4	10	16	6	10	10	11
5	4	8	9	10	4	3
6	8	11	7	11	3	9

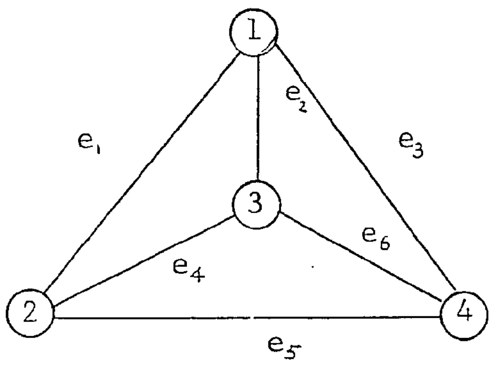

Figure 6-2.

the search tree of this quadratic minimal spanning tree problem is shown in Figure 6-3, where a branching node is represented by, say, $\textcircled{6}$ or $\textcircled{\bar{6}}$. The former includes edge e_6 while the latter excludes it. The number above each node gives the order of its occurrence in the search. The lower bound and the

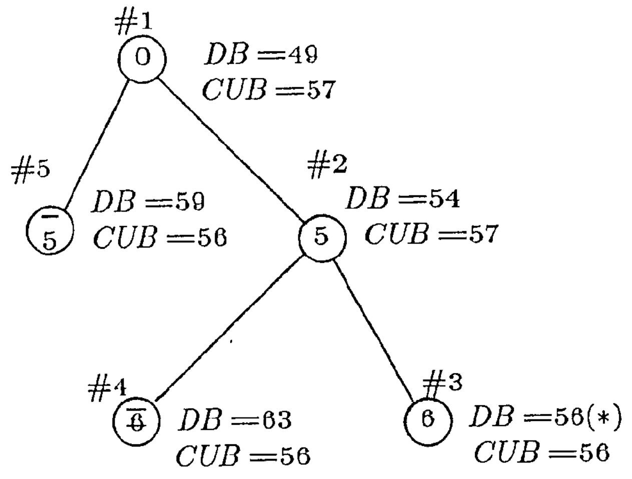

Figure 6-3. A search tree. (*) is the optimum.

best-known upper bound are also indicated for each node.

Let us carry through the computation for node #2 . In the subset represented by #2 , edge e_5 must be included into the spanning tree. Easy to see that $PV = a_{55} = 4$. The new coefficients $\{\bar{b}_i\}$ in (6.2) are :

$$\{\bar{b}_i\} = \{\, 21 \,,\, 17 \,,\, 24 \,,\, 30 \,,\, - \,,\, 15 \,\} \,,$$

where the symbol "—" indicates that e_5 is not a free edge. Then solve (6.4) and obtain

$$\{f_i\} = \{\, 29 \,,\, 28^* \,,\, 30 \,,\, 36 \,,\, - \,,\, 22^* \,\} \,.$$

Take $i = 1$ as example:

$$\bar{b}_1 = b_1 + (a_{15} + a_{51}) = 13 + 4 + 4 = 21 \,.$$

The solution tree for (6.4) must include edge e_1 and e_5 . So, the spanning tree is given by adding edge e_6 to the previous two edges and $f_1 = \bar{b}_1 + a_{16} = 21 + 8 = 29$.

Now solve (6.5) with e_5 already fixed and with f_i as the edge costs. The solution tree is given by $\{\, e_2 \,,\, e_5 \,,\, e_6 \,\}$ which has a value of

$$DB = PV + (f_2 + f_6) = 4 + 28 + 27 = 54 \,.$$

Since f_6 is the smallest one among f_i , $i \neq 5$, node #2 is to be branched into two nodes #4 and #3 as shown in Figure 6-3.

6.3. Branch-and-Bound Algorithm II (with parametrized bounds)

Bounding:

The leveling method for improving the parametrized bound is applied. Here, we do not utilize the best parametrized bound. Instead, we take the parametrized bound at the second iteration of the leveling algorithm as a lower bound on the current node. After evaluating the decomposition bound by solving (6.4) and (6.5), we set

$$\begin{aligned} a_{ij}' &= a_{ij} + f_j/n_1 , & i,j \in \mathbf{F} , \\ b_i' &= \bar{b}_i - (1-\frac{1}{n_1})\, f_i , & i \in \mathbf{F} , \end{aligned} \tag{6.6}$$

where $n_1 = n-1-|\mathbf{U}|$, $\bar{b}_i$ and $\mathbf{F}$ are as indicated in Section 6.1. A parametrized lower bound denoted by PB_1 is obtained by solving (6.4) and (6.5) with a_{ij}' and b_i' as the coefficients.

We also employ Kruskal's algorithm to solve the ordinary minimal spanning tree problems (6.4) and (6.5). Our preliminary testing experience revealed that the feasible solution of (6.5) in the second iteration does not give a better upper bound than in the first iteration. Therefore, we evaluate UB and CUB right after the first iteration. That is, UB and CUB are the same with those in the previous algorithm.

Branching:

The branching rule for choosing an edge is as described for the preceding algorithm.

Computational considerations:

As before, we presort the rows of the matrix A at the beginning of the branch-and-bound procedure so as to save the computational time in evaluating the decomposition bounds. After parametrization, however, the coefficients in (6.4) are changed. Thus, we must sort each row in matrix A again. The second iteration therefore takes a longer time than the first iteration.

Example 6.2.

Let G be an undirected graph shown in Figure 6-4. The intercost matrix is given by Table 6-1.

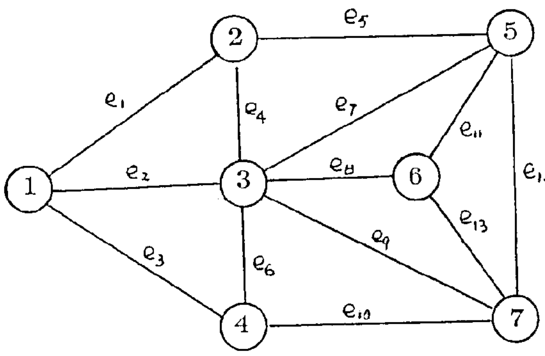

Figure 6-4. Example 6.2.

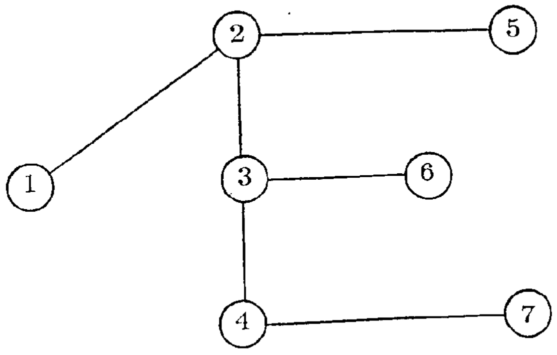

Figure 6-5. Solution tree for Example 6.2.

The branch-and-bound tree is shown in Figure 6-6. The branch-and-bound procedure terminates after searching 21 nodes and arrives at the optimal solution

$$T = \left\{ e_1 , e_4 , e_5 , e_6 , e_8 , e_{10} \right\}$$

which is shown in Figure 6-5. The optimal value is 308 .

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	30	11	3	9	2	20	8	8	14	8	5	15	0
2	11	70	5	18	4	9	20	13	19	18	16	13	17
3	3	5	40	7	5	12	15	18	18	1	9	14	12
4	9	18	7	25	0	9	12	3	9	10	19	19	8
5	2	4	5	0	25	4	16	4	10	6	18	19	11
6	20	9	12	9	4	0	5	5	7	1	18	13	5
7	8	20	15	12	16	5	65	9	6	0	0	8	13
8	8	13	18	3	4	5	9	35	6	5	5	2	11
9	14	19	18	9	10	7	6	6	70	12	13	12	4
10	8	18	1	10	6	1	0	5	12	5	7	19	20
11	5	16	9	19	18	18	0	5	13	7	5	20	3
12	15	13	14	19	19	13	8	2	12	19	20	25	7
13	0	17	12	8	11	5	13	11	4	20	3	7	90

Table 6-1. Intercost matrix $\left[a_{ij} \right]$ for Example 6.2.

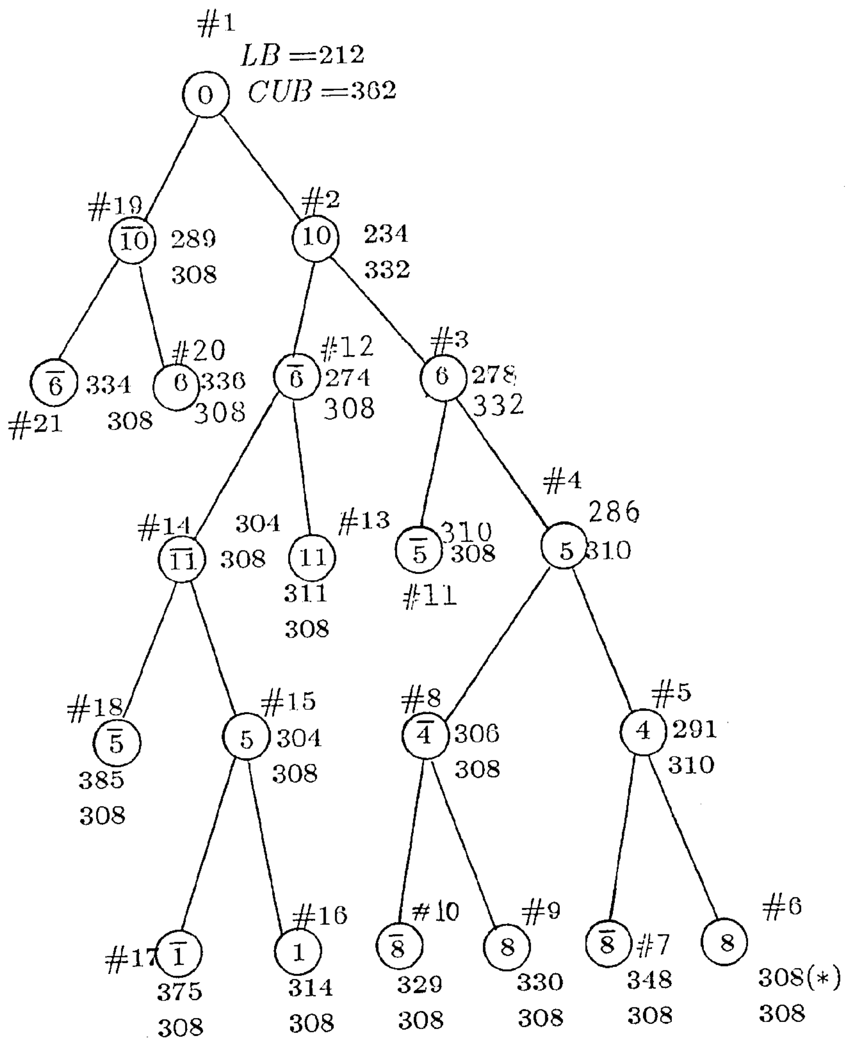

Figure 6-6. The search tree for Example 6.1 with Algorithm I.
(*) Indicates an optimal value.

The application of Algorithm II results in the search tree shown in Figure 6-7. One can see that a smaller tree has resulted. The number of nodes has gone down to 13 from 21 for Algorithm I.

Algorithm I and Algorithm II will be compared further in Section 6.6.

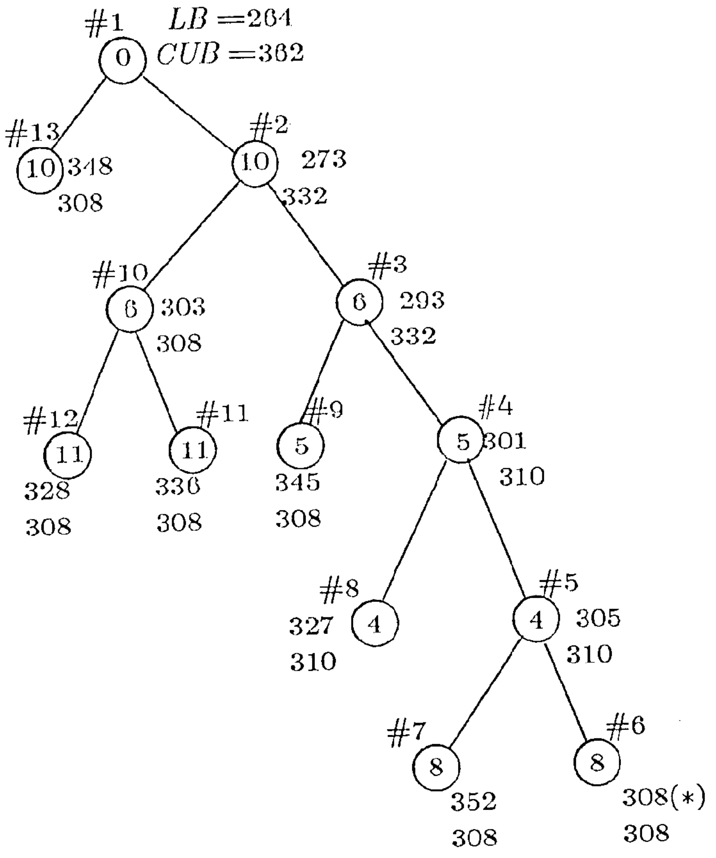

Figure 6-7. The search tree for Example 6.1. with Algorithm II.
(*) indicates an optimal value.

6.4. Branch-and-Bound Algorithm III (for the AQMST)

We have mentioned the Adjacent-only Quadratic Minimal Spanning Tree problem (AQMST) in Chapter 1. This special case of QMST is also an NP-complete problem which arises when the intercosts between nonadjacent edges are all zero. The special data structure of this problem allows us to obtain a more efficient lower bounding technique for it.

Denote $\mathbf{A}_i$ the index set of all edges that are adjacent to edge e_i. Then the objective of the AQMST can be expressed as

$$z(\mathbf{x}) = \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{j \in \mathbf{A}_i} a_{ij}\, x_i x_j \,. \tag{6.7}$$

For a branching node in the search tree, the objective in (6.2) is

$$\begin{aligned} z(\mathbf{x}) &= PV + \sum_{i \in \mathbf{F}} \bar{b}_i x_i + \sum_{i \in \mathbf{F} \cap \mathbf{A}_i} a_{ij}\, x_i x_j \\ &= PV + \sum_{i \in \mathbf{F}} \left[\bar{b}_i + \sum_{j \in \mathbf{F} \cap \mathbf{A}_i} a_{ij}\, x_j \right] x_i \,. \end{aligned} \tag{6.8}$$

To evaluate the lower bound of (6.8), we notice that a spanning tree which contains an edge e_i has the following properties:

(i) It must include at least one edge adjacent to e_i.

(ii) Any vertex other than the endpoints of e_i can be an endpoint of at most two edges adjacent to e_i. At most one of these can go into the spanning tree.

Assume all a_{ij} are nonnegative. Based on property (i), the summation inside the bracket in (6.8), i.e.

$$\bar{b}_i + \sum_{j \in \mathbf{F} \cap \mathbf{A}_i} a_{ij}\, x_j \,, \tag{6.9}$$

can be bounded from below by

$$g_i = \begin{cases} \bar{b}_i + \min\left\{ a_{ij} \mid j \in \mathbf{F} \cap \mathbf{A}_i \right\}, & \text{if } \mathbf{U} \cap \mathbf{A}_i = \phi \,, \\ \bar{b}_i \,, & \text{otherwise} \,, \end{cases} \tag{6.10}$$

where $\mathbf{U}$ is as defined in Section 6.1, i.e. $x_i = 1$ if $i \in \mathbf{U}$ for the current branching node.

The corresponding lower bound for (6.8) is

$$LB = PV + \min\left\{ \sum_{i \in \mathbf{F}} g_i x_i \,\middle|\, \begin{array}{l} \mathbf{x} \in T(G)\,, \\ \mathbf{x} \text{ satisfies } (6.1) \end{array} \right\} . \tag{6.11}$$

Compare this bound with DB in Section 6.2. Because of property (i), we always have $f_i \geq g_i$ hence $LB \leq DB$. However, computing g_i in (6.10) is much easier than computing f_i in (6.4). In fact, if we presort each row of the intercost matrix A at the beginning of the branch-and-bound procedure, one only needs to solve an MST to evaluate LB. The computational requirements for this are significantly less than what evaluating the decomposition bound would require.

We have run this algorithm on some adjacent-only quadratic minimal spanning tree problems and compared it with Algorithm I and Algorithm II working on the same problems. Computational results will be shown in Section 6.6.

When some of a_{ij}'s are negative, (6.9) is no longer bounded from below by g_i. We may modify our bounding procedure by means of property (ii). Suppose edge e_i is fixed in a spanning tree. Let k be any vertex. If k is other than the end points of e_i, then there possibly two edges in $\mathbf{F}$, say e_{j_1} and e_{j_2}, that are adjacent to e_i and end at vertex k. Let

$$d_i(k) = \min\left\{ a_{ij_1}\,,\; a_{ij_2}\,,\; 0 \right\} . \tag{6.12}$$

If k is one of the end points of e_i, then set $d_i(k) = 0$. Let

$$D_i = \sum_{k=1}^{n} d_i(k)\,, \tag{6.13}$$

then $D_i < 0$ implies that some a_{ij}'s in (6.9) are negative. Define

$$h_i = \begin{cases} \bar{b}_i + D_i\,, & \text{if } D_i < 0\,, \\ g_i\,, & \text{if } D_i = 0\,. \end{cases} \tag{6.14}$$

Then a lower bound on (6.8) can be obtained by evaluating (6.11) with g_i

replaced by h_i .

6.5. Heuristic Algorithms

Already, even for networks of moderate sizes, we have found that solving the QMST to optimality is very time-consuming. This argues for using heuristic techniques for larger networks. three heuristics for the QMST are discussed below.

Algorithm H1 (Average Contribution Method).

Let us observe an edge e_k and rewrite the objective value of the QMST by separating all terms containing index k :

$$z(\mathrm{x}) = \left[b_k + \sum_{j \neq k} (a_{jk} + a_{kj})\, x_j \right] x_k + z_2(\mathrm{x})$$

$$= z_1(\mathrm{x}) + z_2(\mathrm{x}) \,, \tag{6.15}$$

where $z_2(\mathrm{x})$ does not contain index k . Suppose that e_k goes into the solution tree. Let us estimate how much e_k would contribute to the objective value. There is a total of $(m-1)$ terms in the summation of $z_1(\mathrm{x})$. All but $(n-2)$ terms are bound to be zero. So, the average contribution of e_k to the objective is

$$p_k = b_k + \frac{n-2}{m-1} \sum_{j \neq k} (a_{kj} + a_{jk}) \,, \quad 1 \leq k \leq m \,. \tag{6.16}$$

After evaluating the average contribution of each e_k , we may solve an ordinary MST :

$$P = \min \left\{ \sum_{k=1}^{m} p_k x_k \,\middle|\, \mathbf{x} \in T(G) \right\} . \qquad (6.17)$$

Then the solution of (6.17) is taken as an approximate solution.

Algorithm H1 needs evaluating (6.16) m times and solving an MST. Total computational time is $O(m^2)$.

Example 6.3. Let us find out a heuristic solution for the problem in Example 6-1. For this problem, we have $n = 4$, $m = 6$. By (6.16) we obtain

$$\{p_k\} = \{ 52.2 , 49.8 , 46.8^* , 63 , 38^* , 41^* \} .$$

The heuristic solution tree is given by $\{ e_3 , e_5 , e_6 \}$ which has an objective value 57 (optimal value = 56).

Heuristic Algorithm H2 (Sequential Fixing Method).

In heuristic H1, the average contributions of all edges are estimated simultaneously. We notice that when some edges in the spanning tree have been fixed, the expected contributions of the remaining edges would be different. Heuristic H2 takes this into consideration. After evaluating the contributions of the edges, it fixes *only one* edge to the growing spanning tree, then re-evaluates the "average" contributions of the remaining edges, and then fixes another edge into the spanning tree. In this way, the edges in the spanning tree are determined sequentially.

Suppose that a number of edges have been fixed into the spanning tree. Denote the index set for these included edges by $\mathbf{U}$, i.e.

$$x_i = 1 , \qquad i \in \mathbf{U} .$$

Some other edges then need not be considered since adding any one of them in to

U would close a cycle. Take the following simple network (Figure 6-8) as an example:

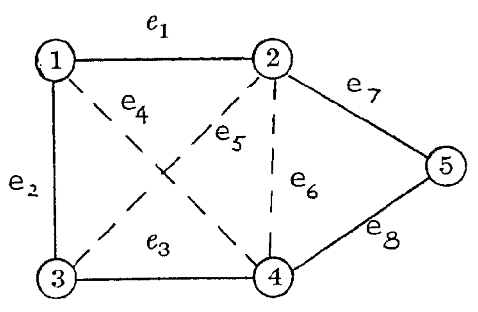

Figure 6-8.

If e_1 , e_2 and e_3 are already in **I** , then e_4 , e_5 and e_6 must be excluded from our further consideration. Denote the index set for these excluded edges by **V** :

$$x_j = 0 , \qquad j \in \mathbf{V} .$$

Let **F** be the index set for the free edges. Then the objective value of the QMST is

$$z(\mathbf{x}) = PV + \sum_{i \in \mathbf{F}} b_i x_i + \sum_{i \in \mathbf{F}} \sum_{j \in \mathbf{U}} (a_{ij} + a_{ji}) \, x_i$$
$$+ \sum_{i \in \mathbf{F}} \sum_{\substack{j \in \mathbf{F} \\ j \neq i}} a_{ij} \, x_i x_j \, .$$

Separating all terms containing index $k \in \mathbf{F}$, we obtain

$$z(\mathbf{x}) = z_1(\mathbf{x}) + z_2(\mathbf{x}) \, ,$$

where

$$z_1(\mathbf{x}) = \left[b_k + \sum_{j \in \mathbf{U}} (a_{kj} + a_{jk}) + \sum_{\substack{j \in \mathbf{F} \\ j \neq k}} (a_{kj} + a_{jk}) \right] x_k \, . \qquad (6.18)$$

The second summation in (6.18) has $m_1 = |\mathbf{F}|$ terms of which all but

$n_1 = n - 2 - |U|$ terms vanish. Thus, the "average" contribution of e_k is

$$q_k = b_k + \sum_{j \in U} (a_{kj} + a_{jk}) + \frac{n_1}{m_1} \sum_{j \in F} (a_{kj} + a_{jk}) . \tag{6.19}$$

We propose the following algorithm :

<u>Heuristic H2.</u>

STEP 0. Start with $\mathbf{U} = \phi$, $\mathbf{F} = \{1, 2, \ldots, m\}$, $n_1 = n-1$, $m_1 = m$.

STEP 1. Compute q_k by (6.19) for all $k \in \mathbf{F}$.

STEP 2. Choose i such that

$$q_i = \min_{k \in \mathbf{F}} \{ q_k \} .$$

Let $x_i = 1$ and set $n_1 \leftarrow n_1 - 1$. If $n_1 = 0$, stop.

STEP 3. Remove from $\mathbf{F}$ those indices k and set $x_k = 0$ if adding edge e_k closes a cycle.

The main computational effort of Algorithm H2 is at step 1 which requires $O(m^2)$ additions at the beginning but much less later. Step 2 needs $O(m)$ computations and step 3 $O(n^2)$. This algorithm takes $(n-1)$ iterations from step 1 through step 3 hence requires $O(m^2 n)$ computational time.

Example 6.4. Continue working on the problem in Example 6.3 but now we use heuristic H2. At the beginning, $m_1 = m-1 = 5$, $n_1 = n-2 = 2$, $\mathbf{U} = \phi$. Hence (6.19) coincides with (6.16). The smallest q_k is $q_5 = 38$.

Fix e_5 into the spanning tree. Then $\mathbf{U} = \{5\}$, $\mathbf{F} = \{1, 2, 3, 4, 6\}$, $n_1 = 1$, $m_1 = 4$. Again we compute q_k from (6.19):

$$(q_k) = (\ 43.5\ ,\ 43.5\ ,\ 42\ ,\ 51.5\ ,\ —\ ,\ 33.5\ )$$

where "—" in the fifth place means that e_5 is not a free edge.

The smallest q_6 brings e_6 into the spanning tree. So, $\mathbf{U} = \{\ 5, 6\ \}$. But then e_4 must be excluded and $\mathbf{V} = \{\ 4\ \}$, $\mathbf{F} = \{\ 1, 2, 3\ \}$. With e_5 and e_6 fixed, $n_1 = 0$ and $m_1 = 2$, the last summation in (6.19) vanishes. We get from (6.19) that

$$\{q_k\} = \{\ 37^*\ ,\ 39\ ,\ 38\ ,\ —\ ,\ —\ ,\ —\ \}\ .$$

Thus, e_1 will go into the spanning tree. The spanning tree is completed now giving $\{\ e_1\ ,\ e_5\ ,\ e_6\ \}$ as a heuristic solution with objective value of 56. This happens to be the optimal value as shown in Example 6.1.

Heuristic Algorithm H3 (Early termination of the branch-and-bound search).

As is well known, a good heuristic solution may be obtained by early terminating the branch-and-bound algorithm. We terminate Algorithm I after 2n iterations and then take the currently best-known solution as the approximate solution.

The computational comparison between these three heuristic algorithms will be shown in Section 6.6.

6.6. Computational Experiments

Our computational experiments consist of three parts. In the first part, we construct a set of quadratic minimal spanning tree test problems and solve them to optimality with Algorithms I and II. In the second part, we run Algorithm III on adjacent-only quadratic minimal spanning tree problems and compare it with Algorithms I and II. In the third part, we execute three heuristics H1, H2 and H3.

Algorithms I and II on the QMST :

As in Section 3.6, we consider complete networks for n from 6 to 12. For each n, we randomly generate a matrix employing the subroutine from the IMSL package on UNIVAC 1100/80 computer of the University of Maryland at College Park. All matrices are symmetric and have integer entries. Starting with seed $=$ 4547057.0D0 (double precision), a sequence of U(0,1)-random numbers $\{ r_i \}$ are generated. Let $a_{ii} = [101 r_i]$. Then another sequence of U(0,1)-random numbers $\{ t_i \}$ are generated, starting with seed $=$ 7797057.0D0. Let $u_i = [21 t_i]$ and fill $\{ u_i \}$ into the intercost matrix in the following order:

$$a_{12} , a_{13} , a_{14} , \ldots , a_{1n} ,$$

$$a_{23} , a_{24} , \ldots , a_{2n} ,$$

$$, \ldots ,$$

$$a_{m-1,m} .$$

Finally, the matrix is completed by symmetry.

Table 6-2 reports (1) the number of nodes in the branch-and-bound tree, (2) the depth of the tree , (3) CPU time used in seconds, and (4) the optimal value.

	# of nodes		Depth of tree		CPU (s)		Optimal value
n	I	II	I	II	I	II	z^*
6	19	11	5	4	.11	.21	207
7	63	23	7	4	.51	.92	326
8	131	39	9	5	1.66	2.71	412
9	577	113	14	9	9.93	11.63	533
10	2397	359	19	15	57.28	58.17	638
11	6289	1069	22	16	208.86	252.30	740
12	54413	7391	32	26	2214.06	2284.37	917

Table 6-2. Comparison of Algorithms I & II.

As expected, the number of nodes and the depth of the search tree for Algorithm II are smaller than those for Algorithm I. However, Algorithm II seems not saving computational time. This is due to a longer time required by the second iteration in evaluating the parametrized lower bounds for which we can not take advantage of presorting. Of course, our codings can certainly benefit from further improvements. For example, it may be better to employ Prim's Algorithm in solving the MST, not Kruskal's Algorithm as we do here.

Algorithm III on the AQMST :

The dimension of our test problems, n , runs from 6 to 20. The diagonal and off-diagonal elements share the same distributions as described above. The only difference is that the non-adjacent entries are preset at zero and thus do not require random generation. Details of the construction of these test problems are also shown in Appendix F. In order to compare Algorithm 3 with Algorithms I and II (which are designed for general QMST's), we have run all three algorithms on the same set of test problems. The computer results are reported in Table 6-3.

	# of nodes			Depth of tree			CPU (s)			Opt. value
n	I	II	III	I	II	III	I	II	III	z^*
6	21	11	21	4	3	4	.12	.23	.04	142
7	35	13	69	7	5	8	.35	.57	.18	203
8	69	35	69	7	5	7	.98	2.48	.25	212
9	159	81	165	9	8	9	3.13	8.52	.73	228
10	145	73	149	8	8	8	4.69	13.94	.87	227
11	373	185	377	11	10	11	15.42	49.68	2.55	237
12	705	359	735	11	10	11	43.2	146.70	5.90	261
13	365	179	365	11	11	11	29.07	105.33	3.75	239
14	2415	1263	2441	15	15	15	233.52	971.74	28.02	286
15	--	--	9337	--	--	17	--	--	125.26	--

Table 6-3. Comparison of Algorithms I, II & III working on Adjacent-only QMST.

We observe that Algorithm III results in search trees of roughly the same size as for Algorithm I and is considerably more efficient than the other two algorithms when appled to AQMST problems.

Heuristic Algorithms :

We now use the set of test problems described in part one of our experiments to test out heuristic algorithms. Approximate values from three heuristic algorithms are compared with the optimal values. The best heuristic value obtained for each problem is marked by (*).

	Heuristic Value			CPU time(s)			Optimal
n	H1	H2	H3	H1	H2	H3	Value
6	211	207*	207*	.0026	.0044	.0788	207
7	326*	335	326*	.0044	.0080	.1468	326
8	468	412*	412*	.0070	.0126	.2468	412
9	727	586	564*	.0108	.0198	.3426	533
10	709	638*	666	.0156	.0294	.5156	638
11	900	740*	761	.0222	.0416	.8076	740
12	1017	937*	937*	.0308	.0580	1.1856	917
13	1397	1087	1085*	.0418	.0784	1.7306	---
14	1818	1348	1262*	.0552	.1048	2.2522	---
15	1781	1621*	1652	.0716	.1346	2.8422	---
16	1912	1578*	1634	.0920	.1720	4.1054	---
17	2409	1910	1891*	.1164	.2168	5.0016	---
18	2920	2227*	2303	.1452	.2690	6.5746	---
19	2887	2453*	2535	.1794	.3296	8.8288	---
20	3439	2688*	2711	.2192	.4006	10.1036	---

Table 6-4. Comparison of three heuristic algorithms.
A * indicates the best heuristic value.

From Table 6-4, we see that both algorithms H2 and H3 give heuristic values close to the optimal ones. H2 works much faster than H3. Moreover, it obtains better solutions when the problem dimension becomes larger.

CHAPTER 7
CONCLUSION

In this dissertation, we introduce and investigate the *Quadratic Minimal Spanning Tree Problem.* In Chapter 2, we discussed how this problem is motivated by the *Stochastic Spanning Tree Problem* with random edge costs in a network as well as other potential applications of this problem. We prove that the Quadratic Minimal Spanning Tree Problem and its special (but more applicable) case, the *Adjacent-only Quadratic Minimal Spanning Tree Problem,* both are NP-hard. Thus, we must resort to implicit enumeration methods to solve it optimally or resort to heuristic algorithms to solve it approximately. Since a good bounding technique is important to the efficiency of a branch-and-bound procedure, this thesis considers several bounding techniques based on linear programming, Lagrangean relaxation, decomposition arguments and parametrization.

We have found the parametrization bound to be superior to others in quality. In order to find out the best parametrized bound, we must solve a parameter search problem which is to maximize a piecewise linear, concave function. We propose the *leveling method,* in lieu of the generally-used (but cumbersome) subgradient method, to solve this problem. Our computational experience with the Quadratic Minimal Spanning Tree Problem shows that the leveling method is both efficient and reliable for solving the parameter search problem. The leveling method is capable of locating the best parameters in just a few iterations. Empirically, the rate of convergence of the leveling method is "geometrical", independent of the problem dimension.

A major contribution of this thesis is to perform a detailed analysis of the leveling algorithm as a general method for obtaining lower bounds for a class of quadratic 0-1 programs. We specify the class of problems to which this method applies, establish its convergence, and provide computational experience with this method on a variety of test problems. These investigations are described in greater detail below.

The class of quadratic 0-1 programs to which the leveling method applies is characterized by two reasonable conditions: (1) an efficient algorithm for solving the linear counterpart of the quadratic problem must be available, and (2) the sum of all variables must equal a fixed number. The parametrization bounding technique for this class of problems results in a parameter search problem similar to the one encountered in the Lagarangean relaxation approach to the Quadratic Minimal Spanning Tree Problem. We prove that the leveling algorithm iteratively solves this search problem to optimality. Moreover, the leveling algorithm improves the objective function with every iteration. These are desirable characteristics for an optimization algorithm which the subgradient method does not possess.

The leveling method can also be used to evaluate lower bounds for the *Quadratic Assignment Problem.* Our computational results show that the lower bounds obtained by the leveling method are better, either in quality or in computational efficiency, than existing techniques proposed by Gilmore[1962], Lawler[1962], Christofides et al[1980] and Frieze and Yadegar[1983], among others.

Besides testing standard problems available in the literature which are all of Koopmans-Beckmann type, we also generate randomly a larger number of general quadratic assignment problems and apply the leveling algorithm to them. Our tests indicate that the lower bounds obtained by the leveling method are 40% larger than those obtained by Gilmore and Lawler's method which is the only existing bounding technique we know that is applicable to the general Quadratic Assignment Problem.

Based on different bounding techniques, we develop three branch-and-bound algorithms to solve the Quadratic Minimal Spanning Tree Problem to optimality. One of them focuses on the Adjacent-only Quadratic Minimal Spanning Tree Problem. These algorithms are coded and run on a set of randomly-generated test problems with dimensions up to 12 nodes and 66 edges for a general Quadratic Minimal Spanning Tree Problem; and up to 15 nodes and 105 edges for an Adjacent-only Quadratic Minimal Spanning Tree Problem.

Having considered exact solution techniques, we also devise three heuristic algorithms for solving the Quadratic Minimal Spanning Tree Problem approximately. Computational results on their performance indicate that one of these algorithms, called the *Sequential Fixing Method,* is a good heuristic which arrives at a near-optimal solution with a low computational burden.

There are a number of other directions along which the Quadratic Minimal Spanning Tree Problem can be investigated in further work. In the following, we briefly list some possible future directions.

(1) Though the general Quadratic Minimal Spanning Tree Problem is NP-hard, there may be some special cases that are solvable by computationally efficient algorithms. For example, we may consider the case in which all but two edges ending at each vertex have no interaction.

(2) As mentioned before, the study of the Quadratic Minimal Spanning Tree Problem was inspired by the Stochastic Spanning Tree Problem. Naturally, we may utilize our results on the Quadratic Minimal Spanning Tree Problem as tools to study the Stochastic Spanning Tree Problem.

(3) The leveling method has proven to be efficient in evaluating lower bounds for the Quadratic Minimal Spanning Tree Problem and the Quadratic Assignment Problem. There may be more problems that belong to our class of quadratic 0-1 programming problems in Chapter 4 hence more applications of the

leveling method.

(4) A new exact algorithm for solving the Quadratic Assignment Problem may be invented by embedding the leveling bound LVB into the branch-and-bound procedure.

(5) We have proved theorems that assure the convergence of the leveling algorithm. However, we are not able to explain, from theoretical point of view, why this algorithm actually converges at a "geometrical" rate as described in the computational experiments on the Quadratic Minimal Spanning Tree and the Quadratic Assignment Problems. There may be a stronger relation between the leveling method and the Lagrangean relaxation than what we have shown in Theorem 3.2. There may also be some connection between this method and the general minimax optimization problems.

(6) The leveling method is based on Theorem 4.1 which gives a characterizing condition for the best parameters. Any algorithm capable of reducing the deviation to zero certainly converges to the best parameters. The leveling method tries to attain this goal iteratively by solving a sequence of ordinary Minimal Spanning Tree Problems in each iteration. The deviation at each iteration needs not be evaluated actually. The computational time for the algorithm mostly spends on solving a large number of Minimal Spanning Tree Problems. However, solving them is only an intermediate step in the course of reducing the deviation. If we can find a way to save these intermediate calculations but preserve the ability of reducing the deviation, then we might be able to significantly improve the performance of this algorithm.

Appendix A. Network Terminology

A *graph* $G = (V, E)$ is a collection of vertices, denoted by $V = \{v_1, v_2, \ldots, v_n\}$, and a collection of edges joining all or part of these vertices, denoted by $E = \{e_1, e_2, \ldots, e_m\}$. A graph $G' = (V', E')$ is called a *subgraph* of G if V' V and E' E.

Let (v_i, v_j) denote an edge linking v_i and v_j. The graph is termed *directed* if we distinguish between the two edges (v_i, v_j) and (v_j, v_i), that is, if (v_i, v_j) is viewed as an ordered pair.

A *path* from a vertex v_{i_0} to vertex v_{i_k} is a finite sequence of edges:

$$\left\{(v_{i_0}, v_{i_1}), (v_{i_1}, v_{i_2}), \ldots, (v_{i_{k-1}}, v_{i_k})\right\}.$$

A path with $i_0 = i_k$ is called a *cycle.*

Two vertices v_i and v_j are *connected* if there exists at least one path between v_i and v_j. A *connected graph* is a graph in which there is a path connecting any pair of vertices. A *component* of a graph G is a maximum connected subgraph of G.

A *tree* is a connected graph containing no cycle. This definition is equivalent to: (i) A tree is a connected graph of n vertices and $(n-1)$ edges. or: (ii) A tree is a graph in which a unique path exists from v_i to v_j for all $v_i, v_j \in V$. A graph is called a *forest* if each of its k components is a tree.

A *spanning tree* (forest) of a graph G is a subgraph of G that is a tree (forest) and contains all vertices of G.

A *Hamiltonian path* from v_i to v_j is a path that starts from v_i and then visits every vertex once and only once and finally ends at v_j. When $v_i = v_j$, it is a *Hamiltonian cycle.*

The *degree* of a vertex in a graph G is the number of distinguished edges starting with the vertex. In a connected undirected graph G, a Hamiltonian path between v_i and v_j is a connected subgraph passing all vertices in G in which all vertices other than v_i and v_j have degree of 2 while these two vertices have degree of 1.

Appendix B. Efficient Algorithms for Solving the MST

There are several "greedy" algorithms that solve the Minimal Spanning Tree Problem very efficiently. Here we state two of them.

Let $G = (V, E)$ be a given graph. $V = (1,2,\ldots,n)$ and $E = (e_1, e_2, \ldots, e_m)$. The cost associated with edge e is $c(e)$.

Kruskal's Algorithm (Kruskal[1957]) This algorithm starts with a spanning forest $F = \{V\}$ containing no edges.

STEP 1. List the edges in E in ascending order according to their costs.

STEP 2. Add the first edge in this list to the forest F provided that this addition does not form a cycle in F.

STEP 3. Remove the first edge from the list. If (n-1) edges have been in F, stop. Otherwise, go to step 2.

Prim-Dijkstra's Algorithm (Prim[1957] and Dijkstra[1959]). This algorithm starts with a subgraph $T = \phi$, vertex subsets $W = \{1\}$ and $W^c = V - W$.

STEP 1. Choose an edge $e = (i, j)$ with $i \in W$ and $j \in W^c$ such that

$$c(e) = \min\left\{ c(e') \mid e' = (i', j'),\, i' \in W,\ j' \in W^c \right\}.$$

STEP 2. Set $W = W \bigcup \{j\}$, $W^c = W^c - \{j\}$.

STEP 3. If $W^c = \phi$, stop. Otherwise, return to step 1.

Both algorithms have been implemented in a veriety of ways (see Charlton and Tarjan[1976], Heymond et al[1980], Kershenbaum and Van Slyke[1972] and Yao[1975]). The worst-case computational complexity of the Kruskal's algorithm ranges from $O(m \log m)$ to $O(m \log \log n)$. The Prim-Dijkstra's algorithm has a worst-case computational complexity $O(n^2)$ for the general graphs and $O(m)$

for the dense graphs.

Published codes for these algorithms can be found in Ross[1969], Witney[1972], Seppanen[1970]. Nijenhuis and Wilf[1978], and Phillips and Garcia-Diaz[1981] among other sources.

Appendix C. Formulations of the QAP and the KBP

Suppose that n machines are to be assigned to n locations, with each machine assigned to a unique location, and vice versa. Suppose that the cost c_{ij} for assigning machine i to location j is known for each (i, j) pair. A linear Assignment Problem is to find an assignment with the minimum total cost. It can be formulated as follows: Given an $n \times n$ matrix $C = \left[c_{ij}\right]$, $1 \leq i,j \leq n$, determine an $n \times n$ solution matrix $\mathbf{X} = \left[x_{ij}\right]$ so as to

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad \text{(C-1)}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} x_{ij} = 1 , \qquad 1 \leq j \leq n , \qquad \text{(C-2)}$$

$$\sum_{j=1}^{n} x_{ij} = 1 , \qquad 1 \leq i \leq n , \qquad \text{(C-3)}$$

$$x_{ij} = 0 \ \ or \ \ 1 , \qquad 1 \leq i,j \leq n . \qquad \text{(C-4)}$$

Since the introduction of the Hungarian method for assignment problem by Kuhn[1955], many efficient algorithms designed for solving it have been publiched (see Hatch[1975], Barr et al[1977], Hung and Rom[1980], Berteskas[1981]). Codings in FORTRAN or ALGOL60 can be found in Burkard and Derigs[1980] and Silver[1960]. Solving an n-dimensional assignment problem requires $O(n^3)$ computations.

The Quadratic Assignment Problem (QAP) and its special case, the Koopmans-Beckmann Problem (KBP), are variants of the assignment problem with linear objective (C-1) replaced by quadratic one.

Koopmans and Beckmann studied the following problem: n plants are to be assigned to n locations. Let f_{ij} be the amount of goods to be transported from plant i to plant j and d_{pq} the distance between location p and q. If plants

i and j are assigned to locations p and q respectively, the cost for transporting goods from plant i to plant j is $f_{ij}d_{pq}$. The problem is to find an assignment so that the total cost of transporting all goods between all plants is the minimum. This problem can be formulated as follows.

$$\text{(KBP)} \quad \text{Minimize} \quad \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} f_{ij}d_{pq}x_{ip}x_{jq} \qquad \text{(C-5)}$$

subject to: $(C\text{–}2) - (C\text{–}4)$.

Lawler[1962] generalized the above problem to the following form: Given n^4 cost coefficients a_{ipjq} , $1 \leq i,p,j,q \leq n$, determine an $n \times n$ solution matrix $\mathbf{X} = \left[x_{ij}\right]$ so as to

$$\text{(QAP)} \quad \text{Minimize} \quad \sum_{i=1}^{n}\sum_{p=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{n} a_{ipjq}x_{ip}x_{jq} \qquad \text{(C-6)}$$

subject to: $(C\text{–}2) - (C\text{–}4)$.

The above problem is named as the *Quadratic Assignment Problem.* Problem (C-5) is a special case of (C-6) and is called the *Koopmans-Beckmann Problem.*

Appendix D. Test Problems for Section 5.3

All Koopmans-Beckmann problem tested in Section 5.3 have symmetric flow matrices $F = [\, f_{ij} \,]$ and distance matrices $D = [\, d_{pq} \,]$. Thus, we show each problem by a matrix A :

$$A = \begin{bmatrix} F & D \end{bmatrix} .$$

Nugent5:

$$\begin{bmatrix} 0 & 1 & 1 & 2 & 3 \\ 5 & 0 & 2 & 1 & 2 \\ 2 & 3 & 0 & 1 & 2 \\ 4 & 0 & 0 & 0 & 1 \\ 1 & 2 & 0 & 5 & 0 \end{bmatrix}$$

Nugent6:

$$\begin{bmatrix} 0 & 1 & 2 & 1 & 2 & 3 \\ 5 & 0 & 1 & 2 & 1 & 2 \\ 2 & 3 & 0 & 3 & 2 & 1 \\ 4 & 0 & 0 & 0 & 1 & 2 \\ 1 & 2 & 0 & 5 & 0 & 1 \\ 0 & 2 & 0 & 2 & 10 & 0 \end{bmatrix}$$

Nugent7:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 3 & 4 \\ 5 & 0 & 1 & 2 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 & 2 & 1 & 2 \\ 4 & 0 & 1 & 0 & 3 & 2 & 1 \\ 1 & 2 & 0 & 5 & 0 & 1 & 2 \\ 0 & 2 & 2 & 2 & 10 & 0 & 1 \\ 0 & 2 & 5 & 2 & 0 & 5 & 0 \end{bmatrix}$$

Nugent8:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 1 & 2 & 3 & 4 \\ 5 & 0 & 1 & 2 & 2 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 & 3 & 2 & 1 & 2 \\ 4 & 0 & 0 & 0 & 4 & 3 & 2 & 1 \\ 1 & 2 & 0 & 5 & 0 & 1 & 2 & 3 \\ 0 & 2 & 0 & 2 & 10 & 0 & 1 & 2 \\ 0 & 2 & 0 & 2 & 0 & 5 & 0 & 1 \\ 6 & 0 & 5 & 10 & 0 & 1 & 10 & 0 \end{bmatrix}$$

Nugent12:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 5 \\ 5 & 0 & 1 & 2 & 2 & 1 & 2 & 3 & 3 & 2 & 3 & 4 \\ 2 & 3 & 0 & 1 & 3 & 2 & 1 & 2 & 4 & 3 & 2 & 3 \\ 4 & 0 & 0 & 0 & 4 & 3 & 2 & 1 & 5 & 4 & 3 & 2 \\ 1 & 2 & 0 & 5 & 0 & 1 & 2 & 3 & 1 & 2 & 3 & 4 \\ 0 & 2 & 0 & 2 & 10 & 0 & 1 & 2 & 2 & 1 & 2 & 3 \\ 0 & 2 & 0 & 2 & 0 & 5 & 0 & 1 & 3 & 2 & 1 & 2 \\ 6 & 0 & 5 & 10 & 0 & 1 & 10 & 0 & 4 & 3 & 2 & 1 \\ 2 & 4 & 5 & 0 & 0 & 1 & 5 & 0 & 0 & 1 & 2 & 3 \\ 1 & 5 & 2 & 0 & 5 & 5 & 2 & 0 & 0 & 0 & 1 & 2 \\ 1 & 0 & 2 & 5 & 1 & 4 & 3 & 5 & 10 & 5 & 0 & 1 \\ 1 & 0 & 2 & 5 & 1 & 0 & 3 & 0 & 10 & 0 & 2 & 0 \end{bmatrix}$$

Nugent15:

$$\begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 & 6 \\
10 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 & 3 & 2 & 3 & 4 & 5 \\
0 & 1 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 & 3 & 2 & 3 & 4 \\
5 & 3 & 10 & 0 & 1 & 4 & 3 & 2 & 1 & 2 & 5 & 4 & 3 & 2 & 3 \\
1 & 2 & 2 & 1 & 0 & 5 & 4 & 3 & 2 & 1 & 6 & 5 & 4 & 3 & 2 \\
0 & 2 & 0 & 1 & 3 & 0 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 \\
1 & 2 & 2 & 5 & 5 & 2 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 \\
2 & 3 & 5 & 0 & 5 & 2 & 6 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 \\
2 & 2 & 4 & 0 & 5 & 1 & 0 & 5 & 0 & 1 & 4 & 3 & 2 & 1 & 2 \\
2 & 0 & 5 & 2 & 1 & 5 & 1 & 2 & 0 & 0 & 5 & 4 & 3 & 2 & 1 \\
2 & 2 & 2 & 1 & 0 & 0 & 5 & 10 & 10 & 0 & 0 & 1 & 2 & 3 & 4 \\
0 & 0 & 2 & 0 & 3 & 0 & 5 & 0 & 5 & 4 & 5 & 0 & 1 & 2 & 3 \\
4 & 10 & 5 & 2 & 0 & 2 & 5 & 5 & 10 & 0 & 0 & 3 & 0 & 1 & 2 \\
0 & 5 & 5 & 5 & 5 & 5 & 1 & 0 & 0 & 0 & 5 & 3 & 10 & 0 & 1 \\
0 & 0 & 5 & 0 & 5 & 10 & 0 & 0 & 2 & 5 & 0 & 0 & 2 & 4 & 0
\end{bmatrix}$$

Nugent20:

$$\begin{bmatrix}
0 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 & 6 & 3 & 4 & 5 & 6 & 7 \\
0 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 & 3 & 2 & 3 & 4 & 5 & 4 & 3 & 4 & 5 & 6 \\
5 & 3 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 & 3 & 2 & 3 & 4 & 5 & 4 & 3 & 4 & 5 \\
0 & 10 & 2 & 0 & 1 & 4 & 3 & 2 & 1 & 2 & 5 & 4 & 3 & 2 & 3 & 6 & 5 & 4 & 3 & 4 \\
5 & 5 & 0 & 1 & 0 & 5 & 4 & 3 & 2 & 1 & 6 & 5 & 4 & 3 & 2 & 7 & 6 & 5 & 4 & 3 \\
2 & 1 & 5 & 0 & 5 & 0 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 4 & 5 & 6 \\
10 & 5 & 2 & 5 & 6 & 5 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 & 3 & 2 & 3 & 4 & 5 \\
3 & 1 & 4 & 2 & 5 & 2 & 0 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 & 3 & 2 & 3 & 4 \\
1 & 2 & 4 & 1 & 2 & 1 & 0 & 1 & 0 & 1 & 4 & 3 & 2 & 1 & 2 & 5 & 4 & 3 & 2 & 3 \\
5 & 4 & 5 & 0 & 5 & 6 & 0 & 1 & 2 & 0 & 5 & 4 & 3 & 2 & 1 & 6 & 5 & 4 & 3 & 2 \\
5 & 2 & 0 & 10 & 2 & 0 & 5 & 10 & 0 & 5 & 0 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 \\
5 & 5 & 0 & 2 & 0 & 0 & 10 & 10 & 3 & 5 & 5 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 & 4 \\
0 & 0 & 0 & 2 & 5 & 10 & 2 & 2 & 5 & 0 & 2 & 2 & 0 & 1 & 2 & 3 & 2 & 1 & 2 & 3 \\
0 & 10 & 5 & 0 & 1 & 0 & 2 & 0 & 5 & 5 & 5 & 10 & 2 & 0 & 1 & 4 & 3 & 2 & 1 & 2 \\
5 & 10 & 1 & 2 & 1 & 2 & 5 & 10 & 0 & 1 & 1 & 5 & 2 & 5 & 0 & 5 & 4 & 3 & 2 & 1 \\
4 & 3 & 0 & 1 & 1 & 0 & 1 & 2 & 5 & 0 & 10 & 0 & 1 & 5 & 3 & 0 & 1 & 2 & 3 & 4 \\
4 & 0 & 0 & 5 & 5 & 1 & 2 & 5 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 2 & 3 \\
0 & 5 & 5 & 2 & 2 & 0 & 1 & 2 & 0 & 5 & 2 & 1 & 0 & 5 & 5 & 0 & 5 & 0 & 1 & 2 \\
0 & 10 & 0 & 5 & 5 & 1 & 0 & 2 & 0 & 5 & 2 & 2 & 0 & 5 & 10 & 2 & 2 & 1 & 0 & 1 \\
1 & 5 & 0 & 5 & 1 & 5 & 10 & 10 & 2 & 2 & 5 & 5 & 5 & 0 & 10 & 0 & 0 & 1 & 6 & 0
\end{bmatrix}$$

Gavett4:

$$\begin{bmatrix} 0 & 28 & 25 & 13 \\ 6 & 0 & 15 & 4 \\ 7 & 5 & 0 & 23 \\ 2 & 6 & 1 & 0 \end{bmatrix}$$

Lawler7: This problem has the flow and distance matrices denoted by $A = \begin{bmatrix} F & D \end{bmatrix}$ and linear cost matrix denoted by $B = \begin{bmatrix} b_{ip} \end{bmatrix}$.

$$A = \begin{bmatrix} 0 & 0 & 6 & 1 & 1 & 8 & 4 \\ 5 & 0 & 1 & 0 & 3 & 1 & 3 \\ 0 & 9 & 0 & 8 & 8 & 4 & 2 \\ 5 & 7 & 9 & 0 & 7 & 6 & 4 \\ 0 & 3 & 4 & 1 & 0 & 0 & 6 \\ 5 & 8 & 4 & 1 & 0 & 0 & 9 \\ 4 & 6 & 4 & 9 & 6 & 9 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 51 & 27 & 14 & 9 & 0 & 18 & 0 \\ 0 & 1 & 22 & 17 & 0 & 41 & 13 \\ 2 & 0 & 13 & 22 & 2 & 12 & 27 \\ 38 & 11 & 0 & 0 & 22 & 13 & 14 \\ 62 & 56 & 0 & 67 & 1 & 0 & 5 \\ 61 & 0 & 3 & 14 & 9 & 1 & 67 \\ 41 & 12 & 23 & 0 & 18 & 41 & 0 \end{bmatrix}$$

Problem 1 ($n = 5$):

$$\begin{bmatrix} 0 & 15 & 12 & 8 & 9 \\ 9 & 0 & 7 & 10 & 12 \\ 12 & 13 & 0 & 8 & 8 \\ 14 & 9 & 6 & 0 & 11 \\ 12 & 12 & 11 & 15 & 0 \end{bmatrix}$$

Problem 2 ($n = 6$):

$$\begin{bmatrix} 0 & 10 & 7 & 6 & 12 & 5 \\ 3 & 0 & 11 & 20 & 5 & 9 \\ 6 & 5 & 0 & 8 & 14 & 13 \\ 4 & 2 & 7 & 0 & 10 & 8 \\ 7 & 6 & 5 & 4 & 0 & 15 \\ 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

Problem 3 ($n = 7$):

$$\begin{bmatrix} 0 & 5 & 9 & 3 & 7 & 8 & 4 \\ 3 & 0 & 8 & 5 & 6 & 3 & 8 \\ 0 & 4 & 0 & 8 & 5 & 4 & 7 \\ 2 & 5 & 3 & 0 & 2 & 0 & 3 \\ 1 & 0 & 4 & 2 & 0 & 5 & 7 \\ 6 & 3 & 4 & 4 & 5 & 0 & 7 \\ 5 & 2 & 2 & 3 & 5 & 6 & 0 \end{bmatrix}$$

Problem 4 ($n = 8$):

$$\begin{bmatrix} 0 & 11 & 12 & 13 & 11 & 12 & 13 & 14 \\ 5 & 0 & 9 & 8 & 8 & 9 & 8 & 7 \\ 2 & 3 & 0 & 3 & 12 & 8 & 15 & 11 \\ 4 & 0 & 0 & 0 & 14 & 9 & 4 & 7 \\ 1 & 2 & 0 & 5 & 0 & 11 & 10 & 14 \\ 0 & 2 & 0 & 2 & 10 & 0 & 14 & 12 \\ 0 & 2 & .0 & 2 & 0 & 5 & 0 & 11 \\ 6 & 0 & 5 & 10 & 0 & 1 & 10 & 0 \end{bmatrix}$$

Problem 5 ($n = 10$):

$$\begin{bmatrix}
0 & 6 & 4 & 5 & 8 & 9 & 3 & 4 & 1 & 3 \\
2 & 0 & 9 & 5 & 4 & 6 & 3 & 4 & 5 & 9 \\
1 & 3 & 0 & 4 & 7 & 6 & 5 & 7 & 5 & 4 \\
3 & 2 & 3 & 0 & 9 & 6 & 5 & 8 & 1 & 3 \\
4 & 2 & 2 & 3 & 0 & 8 & 5 & 4 & 7 & 5 \\
2 & 1 & 2 & 3 & 5 & 0 & 8 & 4 & 8 & 6 \\
3 & 2 & 3 & 4 & 1 & 2 & 0 & 5 & 6 & 4 \\
4 & 2 & 1 & 2 & 3 & 4 & 4 & 0 & 8 & 6 \\
1 & 1 & 4 & 3 & 5 & 5 & 3 & 8 & 0 & 9 \\
2 & 4 & 5 & 3 & 2 & 3 & 4 & 5 & 4 & 0
\end{bmatrix}$$

Problem 6 ($n = 12$):

$$\begin{bmatrix}
0 & 4 & 5 & 4 & 7 & 9 & 7 & 6 & 8 & 8 & 9 & 7 \\
2 & 0 & 8 & 4 & 9 & 10 & 6 & 7 & 5 & 10 & 7 & 6 \\
1 & 3 & 0 & 10 & 3 & 8 & 4 & 7 & 9 & 6 & 8 & 4 \\
2 & 4 & 3 & 0 & 8 & 5 & 7 & 9 & 10 & 5 & 9 & 5 \\
0 & 2 & 1 & 3 & 0 & 9 & 8 & 6 & 7 & 10 & 8 & 7 \\
1 & 3 & 2 & 4 & 3 & 0 & 9 & 10 & 9 & 8 & 6 & 7 \\
3 & 0 & 1 & 2 & 3 & 4 & 0 & 8 & 5 & 6 & 4 & 9 \\
3 & 2 & 3 & 2 & 1 & 2 & 3 & 0 & 8 & 6 & 10 & 7 \\
4 & 0 & 2 & 1 & 2 & 3 & 2 & 5 & 0 & 7 & 4 & 9 \\
2 & 5 & 3 & 1 & 0 & 3 & 2 & 4 & 2 & 0 & 6 & 4 \\
0 & 1 & 2 & 3 & 2 & 4 & 2 & 3 & 4 & 2 & 0 & 10 \\
7 & 3 & 2 & 6 & 4 & 3 & 2 & 5 & 2 & 1 & 2 & 0
\end{bmatrix}$$

BIBLIOGRAPHY

Barr, R.S., Glover, F. and Klingman, D.[1977], "The alternating basis algorithm for assignment problem," *Mathematical Programming,* **13,** 1-13.

Bazaraa, M.S. and Sherali, H.D.[1980], "Bender's partitioning scheme applied to a new formulation of the quadratic assignment problem," *Naval Res. Logist. Quart.,* **27,** 29-41.

Bertsekas, D.P.[1981], "A new algorithm for the assignment problem," *Mathematical Programming,* **21,** 152-171.

Bradley, G.H.[1975], "Survey of deterministic networks," *AIIE Transactions,* **7,** 222-234.

Burkard, R.E.[1984], "Quadratic assignment problems," *European Journal of Operations Research,* **15,** 283-289.

Burkard, R.E. and Derigs, U.[1980], *Assignment and Matching Problems: Solution Methods with FORTRAN-Programs,* Springer: Berlin.

Burkard, R.E. and Stratmann, K.H.[1978], "Numerical investigations on quadratic assignment problems," *Naval Res. Logist. Quart.,* **25,** 129-148.

Camerini, P.M., Galbiati, G. and Maffioli, F.[1980], "Complexity of spanning tree problems: part I," *European Journal of Operations Research,* **5,** 346-352.

Chandy, K.M. and Lo, T.[1974], "The capacitated minimum spanning tree," *Networks,* **3,** 173-181.

Charnes, A. and Cooper, W.W.[1963], "Deterministic equivalents for optimizing and satisficing under chance constraints," *Operations Research,* **11,** 18-39.

Cherlton, D. and Tarjan, R.E.[1976], "Finding minimum spanning trees," *SIAM J. on Compututing,* **5,** 724-742.

Christofides, N., Mingozzi, A. and Toth, P.[1980], "Contributions to the quadratic assignment problem," *European Journal of Operations Research,* **4,** 243-247.

Christofides, N., Mingozzi, A. and Toth, P.[1981], "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations," *Mathematical Programming,* **20,** 255-282.

Cook, S.A.[1971], "The complexity of theorem proving procedure," *Proc. 3rd ACM Symp. on the Theory of Computing,* 151-158.

Dijkstra, E.W.[1959], "A note on two problems in connection with graphs," *Numer. Math.,* **1,** 269-271.

Edwards, C.S.[1980]. "A branch and bound algorithm for the Koopmans-Beckmann quadratic assignment problem," *Mathematical Programming Study,* **13,** 35-52.

Elshafei, A.N.[1977], "Hospital layout as a quadratic assignment problem," *Operations Research Quarterly,* **1,** 167-179.

Frieze, A.M. and Yadegar, J.[1983], "On the quadratic assignment problem," *Discrete Applied Mathematics,* **5,** 89-98.

Garey, M.R. and Johnson, D.S.[1979], *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H.Freeman and Company: San Francisco, p.218, pp.56-60.

Gavett, J.W. and Plyter, N.V.[1966], "The optimal assignment of facilities to locations by branch and bound," *Operations Research,* **14,** 210-232.

Gavish, B.[1982], "Topological design of centralized computer networks -- formulations and algorithms," *Networks,* **12,** 355-377.

Gavish, B. and Srikanth, K.[1983], "An optimal solution method for large-scale multiple traveling salesman problem," *Working Paper No. 8027,* The University of Rochaster.

Geofferion, A.M.[1974], "Lagrangean relaxation for integer programming," *Mathematical Programming Study,* **2,** 82-114.

Gilmore, P.C.[1962], "Optimal and sub-optimal algorithms for the quadratic assignment problem," *SIAM J. on Applied Mathematics,* **10,** 305-313.

Goffin, J.L.[1977], "On the rates of subgradient optimization methods," *Mathematical Programming,* **13,** 329-347.

Gomory, R.E. and Hu, T.C.[1961], "Multi-terminal network flows," *SIAM J. on Applied Mathematics,* **9,** 551-571.

Hansen, P.[1975], "Fonctions d'evaluation et penalites pour les programmes quadratiques en variables zero-un," In: B.Roy, ed., *Combinatorial Programming: Methods and Applications,* Reidel: Dordrecht, 361-370.

Hansen, P.[1979], "Methods of nonlinear 0-1 programming," *Annals of Discrete*

Mathematics, **5,** 53-70.

Hatch, R.S.[1975], "Bench marks comparing transportation codes based on simple primal-dual algorithms," *Operations Research,* **23,** 1167-1172.

Haymond, R.E., Jarvis, J.P. and Shier, D.R.[1980], "Computational Methods for minimum spanning tree problems," *Technical Report #354,* Department of Mathematical Science, Clemson University.

Held, M. and Karp, R.[1970], "The traveling salesman problem and minimum spanning trees," *Operations Research,* **18,** 1138-1162.

Held, M. and Karp, R.[1971], "The traveling salesman problem and minimum spanning trees: part II," *Mathematical Programming,* **1,** 6-25.

Held, M., Wolfe, P. and Crowder, H.[1974], "Validation of subgradient optimization," *Mathematical Programming,* **6,** 62-88.

Hung, M.S. and Rom, W.O.[1980], "Solving the assignment problem by relaxation," *Operations Research,* **28,** 969-982.

Ishii, H., Shiode, S. and Nishida, T.[1981], "Chance constrained spanning tree problem," *Journal of the Operations Research Society of Japan,* **24,** 147-157.

Ishii, H., Shiode, S., Nishida, T. and Namasuya, Y.[1981], "Stochastic spanning tree problem," *Discrete Applied Mathematics,* **3,** 263-273.

Karp, R.M.[1972], "Reducibility among combinatorial problems." In: R.E. Miller and J.W. Thatcher (eds.), *Complexity of Computer Computations,* Plenum Press: New York, 85-103.

Kershenbaum, A. and Van Slyke, R.[1972], "Computing minimum spanning trees efficiently," *Proc. ACM Nat. Conf.*, 518-527.

Kolbin, V.V.[1971], *Stochastic Programming,* in: R.V. Gamkrelidze, ed., *Progress in Mathematics,* Volume 11, translated by J.S. Wood, Plenum Press: New York, pp. 12-14.

Koopmans, T.C. and Beckmann, M.J.[1957], "Assignment problems and the location of economic activities," *Econometrica,* **25,** 52-76.

Krarup, J. and Pruzan, P.M.[1978], "Computer-aided layout design," *Mathematical Programming Study,* **9,** 75-94.

Kruskal, Jr.[1956], "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math. Soc.,* **7,** 48-50.

Kuhn, H.W.[1955], "The Hungarian method for the assignment problem," *Naval Res. Logist. Quart.,* **12,** 83-97.

Lawler, E.L.[1962], "The quadratic assignment problem," *Management Science,* **19,** 586-590.

Nijenhuis, A. and Wilf, H.S.[1978], *Combinatorial Algorithms for Computers and Calculators,* (2nd ed.), Academic Press: New York. 283-284.

Nugent, C.E., Vollmann, T.E. and Ruml, J.[1968], "An experimental comparison of techniques for the assignment of facilities to locations," *Operations Research,* **16,** 150-173.

Phillips, D.T. and Garcia-Diaz, A.[1981], *Fumdamentals of network analysis,* Prentice-Hall: Englewood Cliffs, N.J., 435-436.

Prim, R.C.[1957], "Shortest connection networks and some generations," *Bell System Tech. J.,* **36,** 1389-1401.

Ross, G.J.S.[1969], "Algorithm AS13: minimum spanning tree," *Appl. Statistics.* **1,** 103-104.

Roucairol, C.[1979], "A reduction method for quadratic assignment problems," *Operations Research,* Verfahren 32, 183-187.

Sahni, S. and Gonzalez, T.[1976], "P-complete approximation problem," *J. ACM,* **23,** 555-565.

Seppanen, J.J.[1970], "Algorithm 399: spanning tree," *Comm. ACM,* **10,** 621-622.

Silver, R.[1960], "An algorithm for the assignment problem," *Commun. Assoc. Comput.,* **3,** 603-606.

Steinberg, L.[1961], "The backboard wiring problem: a placement algorithm," *SIAM Review,* **3,** 37-50.

Whitney, V.K.M.[1972], "Algorithm 422: minimal spanning tree," *Comm. ACM,* **4,** 273-274.

Yao, A.C.[1975], "An $O(|E| \log \log |V|)$ algorithm for finding spanning trees," *Inform. Process. Lett.,* **4,** 21-23.

CURRICULUM VITAE

Name: Weixuan Xu

Permanent adress: The Institute of Applied Mathematics
Academia Sinica
Beijing, China.

Degree and date to be conferred: Ph.D., 1984.

Date of birth: July 29, 1941.

Place of birth: Liuzhow, China.

Secondary education: The first middle school of Guangzhow, 1959
Guangzhow, China.

Collegiate institutions attended	Dates	Degree	Date of Degree
University of Sun Yat-sen Guangzhow, China	1959-1964	B.A.	1964
Chinese University of Technology & Science Graduate School Beijing, China	1964-1968	N/A	1968 (graduated)
University of Maryland	1980-1984	Ph.D.	1984.

Major: Applied Mathematics.

Minor: Operations Research.

Professional positions hels:

Research Associate, 1984-
Institute of Applied Mathematics
Academia Sinica
Beijing, China.

Graduate teaching assistant, 1980-1984
University of Maryland, Department of Mathematics
College Park, Maryland 20742.

Research Associate, 1979-1980
Institute of Applied Mathematics
Academia Sinica
Beijing, China.

Research assistant, 1977-1979
Institute of Applied Mathematics
Academia Sinica
Beijing, China.

Teacher, 1972-1977
The Middle School Affliated to the Xinyu railway
Jianxi, China.