Continuous Optimization Methods for the Quadratic Assignment Problem

Tao Huang

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Statistics and Operations Research.

Chapel Hill
2008

Approved by:

Jon W. Tolle

J. Scott Provan

David S. Rubin

Vidyadhar G. Kulkarni

Shu Lu

Abstract

TAO HUANG: Continuous Optimization Methods for
the Quadratic Assignment Problem
(Under the direction of Jon W. Tolle)

In this dissertation we have studied continuous optimization techniques as they are applied in nonlinear 0-1 programming. Specifically, the methods of relaxation with a penalty function have been carefully investigated. When the strong equivalence properties hold, we are guaranteed an integer solution to the original 0-1 problem. The quadratic assignment problem (QAP) possesses such properties and consequently we have developed an algorithm for the QAP based on the method of relaxation using the quadratic penalty function. In our algorithm we have applied two pre-conditioning techniques that enables us to devise a scheme to find a good initial point and hence obtain good solutions to the QAP. Furthermore, we have shown how quadratic cuts can be used to improve on the current solutions. Extensive numerical results on several sets of QAP test problems (including the QAPLIB) have been reported and these results show our algorithm produces good solutions for certain classes of problems in a small amount of time.

To Ping, Teddy, and Ethan

Acknowledgements

For many pursuing a doctoral degree is probably a pure academic endeavor, requiring prolonged and dedicated efforts in a graduate program. My own pursuit has crossed beyond that. Jiggling between a full-time job, family responsibilities and school has proved very challenging to me. Along my journey there were many to whom I felt truely indebted; if it were not for them I would have never come to where I am now. As I come to the end of this journey, I would like to take this opportunity to express my sincere gratitude to them.

Foremost, I would like to thank Him for answering many of my prayers and for being a source of ideas and guidance. I am grateful to my wife, Ping, and sons, Teddy and Ethan, for their patience, understanding, and sacrifice during many lost evenings and weekends. I only feel relieved now I can finally fulfill my overdue promise to take my son, Teddy, to soccer practice. I am also indebted to my parents for their silent and unconditional support during the entire process.

I thank my advisor, Dr. Jon Tolle, for the opportunity to work on such an interesting, although very challenging, problem, and for his patience and guidance that helped me complete this dissertation. My thanks also go to the rest of my dissertation committee members: Drs. J.S. Provan, D.S. Rubin, V.G. Kulkarni, and S. Lu for their time and suggestions.

Last but not the least, I offer my deepest thanks to my friends whose names I do not list; their encouragement and support have been invaluable.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Consider a situation where we have N facilities and N locations, and we are to assign each facility to a location. For a possible assignment, there is a cost associated with each pair of facilities which is proportional to both the flow and the distance between the pair. There is also a cost of placing a facility at a certain location. Our objective is to assign the facilities to the locations such that the total cost is minimized. Specifically, suppose we are given a set $\mathcal{N} = \{1, 2, \ldots, N\}$, a flow matrix $F = (f_{ij}) \in \mathbb{R}^{N\times N}$, a distance matrix $D = (d_{kl}) \in \mathbb{R}^{N\times N}$, and a fixed cost matrix $C = (c_{ik}) \in \mathbb{R}^{N\times N}$, where f_{ij} represents the cost associated with transporting the commodities over a unit distance from facility i to facility j, d_{kl} is the distance from location k to location l, and c_{ik} is the cost of assigning facility i to location k. The problem can be formulated as follows.

$$\underset{\pi\in\Pi_N}{\text{minimize}} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij}d_{\pi(i)\pi(j)} + \sum_{i=1}^{N} c_{i\pi(i)} \tag{1.1}$$

where Π_N is the set of all the permutations of $\mathcal{N}$. The product $f_{ij}d_{\pi(i)\pi(j)}$ is the cost of simultaneously assigning facility i to location $\pi(i)$ and facility j to location $\pi(j)$. Although in the context of facility to location assignments the distance matrix D is almost always symmetric, in a more general setup no assumption is made on the symmetry of the matrices F and D.

Problem (1.1) is known as the quadratic assignment problem (or QAP for short). It was originally introduced in 1957 by Koopmans and Beckmann [70]. Koopmans and Beckmann derived the QAP as a mathematical model to study the assignment of a set of economic activities to a set of locations. Since then the QAP has been a subject of extensive research in combinatorial optimization, not only due to its theoretical and practical

importance, but also because of its complexity. With the advent of high performance computing capability, we have seen a large number of formerly intractable combinatorial problems that have become practically solvable in the last decade. Unfortunately, the QAP is not among those problems. In general, QAP instances of size $N \geq 20$ are still considered intractable. The QAP remains one of the most challenging combinatorial problems from both a theoretical and a practical point of view.

For recent comprehensive surveys of various aspects of the QAP, the reader is referred to Anstreicher [4], Burkard and Çela [18], Burkard et al. [19], Çela [27], Finke et al. [33], and Pardalos et al. [92].

To facilitate the discussion, we will denote a QAP instance with a flow matrix F, distance matrix D, and cost matrix C by $\mathrm{QAP}(F, D, C)$. When $C = 0$, we denote the instance by $\mathrm{QAP}(F, D)$.

1.1.1 Applications

The QAP initially occurred as a facility location problem, which remains one of its major applications. In addition, the QAP has found applications in other areas such as scheduling [40], manufacturing [51], parallel and distributed computing [13], combinatorial data analysis [63], ergonomics [81, 96], archaeology [71], and sports [61]. In the following we briefly describe two applications that will shed insight into the applicability and significance of the QAP.

Steinberg Wiring Problem. This early application is due to Steinberg [106] who gave a detailed account of computer backboard wiring problems. Given a set $E = \{E_1, E_2, \ldots, E_m\}$ of m electronic components, we require E_i to be connected to E_j by W_{ij} wires. Hence we have a symmetric connection matrix $W = (w_{ij})$ with its diagonal elements $w_{ii} = 0$, $i = 1, 2, \ldots, m$. In addition, there are r points $P_1, P_2, \ldots, P_r$ on the backboard, where $r \geq m$. Let d_{kl} be some distance measure that may be interpreted as the length of wire needed to connect two electronic components if one is placed at P_k and the other is placed at P_l. Hence we obtain a symmetric distance matrix $D = (d_{kl})$ with zeros down the diagonal. The most commonly used distance measures between two points are the Euclidean distance and Manhattan distance, i.e., the ℓ_2-norm and ℓ_1-norm respectively of a vector that emanates from one point and ends at the other. The goal is to place the m electronic components at the r points on the backboard such that the total

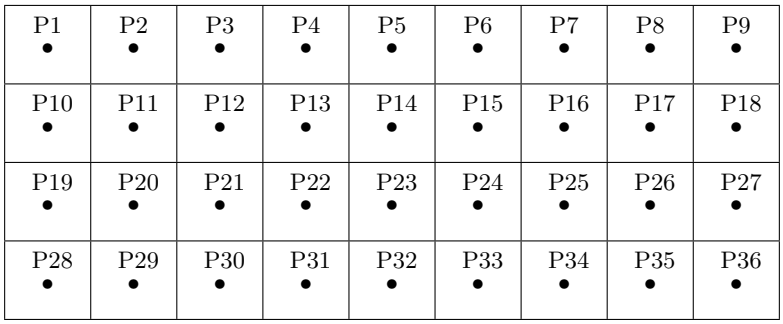


Figure 1.1 Backboard Configuration[1]

wire length needed is minimized. By introducing $r - m$ *fictitious* electronic components with no wires running to or between them, we get r components that have to be placed at r points. Hence the backboard wiring problem becomes a quadratic assignment problem; the connection matrix W becomes the flow matrix in problem (1.1).

As illustrated in Figure 1.1, Steinberg showed a section of the backboard of a modified Univac Solid-State Computer, on which 34 electronic components $E_1, E_2, \ldots, E_{34}$ were to be placed and connected. The dots $P_1, P_2, \ldots, P_{36}$ indicate the possible positions where the electronic components must be placed. The positions form a two-dimensional grid and any two adjacent dots are at a distance of 1 unit both horizontally and vertically. Steinberg provided the connection matrix $W = (w_{ij})$, for $1 \leq i < j \leq 34$. In this example, two fictitious components E_{35} and E_{36} are added so that in every assignment two positions will be empty.

Hospital Layout. Elshafei [32] at the Institute of National Planning in Cairo, Egypt investigated a hospital layout problem in which several clinics of a public hospital in Cairo were to be located so as to minimize the total distance that the patients had to travel to receive treatment in the clinics. The problem resulted from a rapid increase in the number of patients in the Outpatient department at the time. As a result, the department became overcrowded with an average daily number of patients exceeding 700 and those patients having to move among its 17 clinics. The location of the clinics relative to one another was criticized for causing too much travel by the patients and

[1]Copyright ©1961 Society for Industrial and Applied Mathematics. Reprinted with permission from [106].

Table 1.1 Hospital Layout – Facilities and Their Functions[2]

Facility	Function	Facility	Function
1	Receiving and Recording	11	X-Ray
2	General Practitioner	12	Orthopedic
3	Pharmacy	13	Psychiatric
4	Gynecological and Obstetric	14	Squint
5	Medicine	15	Minor Operations
6	Pediatric	16	Minor Operations
7	Surgery	17	Dental
8	Ear, Nose and Throat	18	Dental Surgery
9	Urology	19	Dental Prosthetic
10	Laboratory		

serious delays. Since the flow of the patients was confined between the receiving and recording room and the 17 clinics, the study focused exclusively on the relative location of these 18 facilities. All the facilities needed roughly the same area except for the Minor Operations which occupied nearly twice as much space as any other facility. Hence the Minor Operations section was split into two *pseudo* facilities that had to be placed side by side. As in Table 1.1, there were 19 facilities to be considered in the layout planning.

As Elshafei stated, the estimates of the patient flows between the clinics were available on a yearly basis. Thus a symmetric flow matrix was obtained by averaging the flows between each pair of facilities. The flow between the two pseudo facilities was assigned a very large number such that their adjacency was forced. The distances between the locations could be reasonably measured by tracing the paths of the patients moving from one location to another; the resulting distance matrix therefore was symmetric. Thus the hospital layout problem was formulated as a quadratic assignment problem. In Table 1.2 the lower triangular part shows the flows between the facilities and the upper triangular part shows the distances between the locations.

Other applications of the QAP abound in the literature. For more references to different applications of the QAP, the reader is referred to Burkard et al. [19], Çela [27], Pardalos et al. [92], and Padberg and Rajal [90].

[2]Copyright ©1977 Palgrave Macmillan. Reprinted with permission from [32].

Table 1.2 Hospital Layout – Flow and Distance Matrices[2]

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19
F01	0	12	36	28	52	44	110	126	94	63	130	102	65	98	132	132	126	120	126
F02	76687	0	24	75	82	75	108	70	124	86	93	106	58	124	161	161	70	64	70
F03	0	40951	0	47	71	47	110	73	126	71	95	110	46	127	163	163	73	67	73
F04	415	4118	3848	0	42	34	148	111	160	52	94	148	49	117	104	109	111	105	111
F05	545	5767	2524	256	0	42	125	136	102	22	73	125	32	94	130	130	136	130	136
F06	819	2055	3213	0	0	0	148	111	162	52	96	148	49	117	152	152	111	105	111
F07	135	1917	2072	0	0	0	0	46	46	136	47	30	108	51	79	79	46	47	41
F08	1368	2746	4225	0	0	0	0	0	69	141	63	46	119	68	121	121	27	24	36
F09	819	1097	566	0	47	0	196	0	0	102	34	45	84	23	80	80	69	64	51
F10	5630	5712	0	829	1655	926	1538	0	1954	0	64	118	29	95	131	131	141	135	141
F11	0	0	0	128	287	161	196	301	418	0	0	47	56	54	94	94	63	46	24
F12	3432	0	404	0	0	0	0	0	0	282	1686	0	100	51	89	89	46	40	36
F13	9082	0	9372	0	42	0	0	0	0	0	0	0	0	77	113	113	119	113	119
F14	1503	268	0	0	0	0	0	0	0	0	0	0	0	0	79	79	68	62	51
F15	0	0	972	0	0	0	0	0	0	0	0	0	0	0	0	10	113	107	119
F16	0	1373	0	0	0	0	0	0	0	0	0	0	0	0	99999	0	113	107	119
F17	13732	268	13538	0	226	0	0	0	0	0	226	0	0	0	0	0	0	6	24
F18	1368	0	1368	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
F19	1783	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1.2 Alternative Formulations

The quadratic assignment problem is often stated as (1.1). This is because problem (1.1) readily reveals the combinatorial structure of the QAP. However, problem (1.1) is rarely used in posing solution methods for the QAP. Instead, alternative formulations have been suggested that allow for different solution approaches. In the following we discuss three alternative formulations that are often cited in the literature.

1.2.1 Quadratic 0-1 Programming Formulation

We observe that there is a one-to-one correspondence between Π_N, the set of all the permutations of $\mathcal{N}$, and the set of $N \times N$ permutation matrices $X = (x_{ij})$. To be specific, we have

$$x_{ij} = \begin{cases} 1 & \text{if facility } i \text{ is assigned to location } j, \text{ i.e., } \pi(i) = j; \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

X is a permutation matrix if and only if

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1, \ldots, N, \tag{1.3a}$$

$$\sum_{i=1}^{N} x_{ij} = 1, \quad j = 1, \ldots, N, \tag{1.3b}$$

$$x_{ij} \in \mathbb{B}, \quad i, j = 1, \ldots, N, \tag{1.3c}$$

where $\mathbb{B} = \{0, 1\}$.

(1.3a) and (1.3b) are referred to as the *assignment constraints* and the coefficient matrix of the assignment constraints referred to as the *assignment matrix*. With the assignment constraints, we can formulate the QAP as the following quadratic 0-1 programming problem:

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} f_{ij} d_{kl} x_{ik} x_{jl} + \sum_{i=1}^{N}\sum_{k=1}^{N} c_{ik} x_{ik} \\ &\text{subject to} && \text{(1.3a)–(1.3c).} \end{aligned} \tag{1.4}$$

Problem (1.4) is also called the Koopmans-Beckmann formulation because it was initially used by Koopmans and Beckmann [70].

The equivalence between problems (1.1) and (1.4) is obvious. The term $f_{ij}d_{kl}x_{ik}x_{jl}$ contributes to the objective function only if $x_{ik} = x_{jl} = 1$, i.e., if facility i is assigned to location k and facility j to location l. The term c_{ik} contributes to the objective function only if facility i is assigned to location k.

Note that the sum of the constraints in (1.3a) is equal to the sum of the constraints in (1.3b). Hence the assignment constraints in (1.3a) and (1.3b) are linearly dependent. It is easy to see that after deleting one of the assignment constraints, the remaining assignment constraints are linear independent. Therefore, we can write (1.4) in a more compact form using the matrix-vector notation:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}x^TQx + c^Tx \\ &\text{subject to} && Lx = b \\ & && x \in \mathbb{B}^n, \end{aligned} \tag{1.5}$$

where $Q \in \mathbb{R}^{n\times n}$ with $n = N^2$ is symmetric, $c \in \mathbb{R}^n$, $L \in \mathbb{R}^{m\times n}$ with $m = 2N - 1$, and $b \in \mathbb{R}^m$ is a vector of all ones. Note that the assumption on the symmetry of Q is valid because we can always substitute $\frac{1}{2}(Q + Q^T)$ for Q to achieve the symmetry if Q is asymmetric. In this formulation Q and L have special structures. More generally, (1.5) with arbitrary symmetric Q and L with full row rank is called a quadratic 0-1 programming problem.

1.2.2 Trace Formulation

For two $N \times N$ matrices A and B, we have

$$\operatorname{tr}(AB^T) = \langle A, B\rangle = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}b_{ij}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix and $\langle\cdot,\cdot\rangle$ the *inner product* of two matrices. Given a permutation $\pi \in \Pi_N$, the associated permutation matrix X as defined in (1.2), and a $N \times N$ matrix D, XD is equivalent to a rearrangement of the rows of D in the order of π and DX^T a rearrangement of the columns of D in the order of π. Therefore, we have

$$\begin{aligned} &XDX^T = \big(d_{\pi(i)\pi(j)}\big), \\ &\operatorname{tr}(FXD^TX^T) = \langle F, XDX^T\rangle = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij}d_{\pi(i)\pi(j)}, \end{aligned} \tag{1.6}$$

and also

$$\text{tr}(CX^T) = \sum_{i=1}^{N} c_{i\pi(i)}. \tag{1.7}$$

The following are some of the well-known properties of the traces of two matrices A and B:

$$\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B), \tag{1.8a}$$

$$\text{tr}(AB) = \text{tr}(BA), \quad \text{and} \tag{1.8b}$$

$$\text{tr}\, A = \text{tr}\, A^T. \tag{1.8c}$$

From (1.8a), (1.6), and (1.7), the QAP of the form (1.1) can be written as

$$\begin{aligned} &\text{minimize} && \text{tr}\left[(FXD^T + C)X^T\right] \\ &\text{subject to} && X \in \mathcal{P}_N \end{aligned} \tag{1.9}$$

where $\mathcal{P}_N$ is the set of all $N \times N$ permutation matrices. Furthermore, using the properties (1.8b) and (1.8c) we will show that if either $F = F^T$ or $D = D^T$ then

$$\text{tr}(FXD^TX^T) = \text{tr}(FXDX^T). \tag{1.10}$$

For $D = D^T$, (1.10) holds trivially. For $F = F^T$ but $D \neq D^T$, it follows that

$$\begin{aligned} \text{tr}(FXD^TX^T) &= \text{tr}(FXD^TX^T)^T \\ &= \text{tr}(XDX^TF^T) \\ &= \text{tr}(XDX^TF) \\ &= \text{tr}(FXDX^T). \end{aligned}$$

Similarly we can show that if either $F = F^T$ or $D = D^T$ then

$$\text{tr}(F^TXDX^T) = \text{tr}(FXDX^T). \tag{1.11}$$

(1.10) and (1.11) suggest that we can reformulate a QAP instance in which either the flow matrix or the distance matrix but not both are asymmetric into an equivalent instance in which both matrices are symmetric. For example, in a case where F is symmetric but D is not, we substitute $\bar{D} = \frac{1}{2}(D + D^T)$ for D. In Section 4.3.2 we will use the above result to symmetrize F or D when one or the other is asymmetric.

1.2.3 Kronecker Product

In the Koopmans-Beckmann formulation in (1.4), we can rewrite

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} f_{ij}d_{kl}x_{ik}x_{jl}$$

as

$$x^T \begin{pmatrix} f_{11}d_{11} & \cdots & f_{1N}d_{11} & & & f_{11}d_{1N} & \cdots & f_{1N}d_{1N} \\ \vdots & \ddots & \vdots & \cdots & & \vdots & \ddots & \vdots \\ f_{N1}d_{11} & \cdots & f_{NN}d_{11} & & & f_{N1}d_{1N} & \cdots & f_{NN}d_{1N} \\ & \cdot & & \cdot & & & \cdot & \\ & \cdot & & & \cdot & & \cdot & \\ & \cdot & & & & & \cdot & \\ f_{11}d_{N1} & \cdots & f_{1N}d_{N1} & & & f_{11}d_{NN} & \cdots & f_{1N}d_{NN} \\ \vdots & \ddots & \vdots & \cdots & & \vdots & \ddots & \vdots \\ f_{N1}d_{N1} & \cdots & f_{NN}d_{N1} & & & f_{N1}d_{NN} & \cdots & f_{NN}d_{NN} \end{pmatrix} x$$

$$= x^T \begin{pmatrix} d_{11}F & \cdots & d_{1N}F \\ \vdots & \ddots & \vdots \\ d_{N1}F & \cdots & d_{NN}F \end{pmatrix} x = x^T(D \otimes F)x$$

with $x = \mathrm{vec}(X) = \begin{pmatrix} x_{11}\cdots x_{N1} & \cdots & x_{1N}\cdots x_{NN} \end{pmatrix}^T$. Here $\mathrm{vec}(\cdot)$ denotes the vector formed by the columns of a matrix and $\otimes$ denotes the Kronecker product. Thus the QAP can be formulated as

$$\begin{aligned} (1.12) \qquad & \text{minimize} && \mathrm{vec}(X)^T(D \otimes F)\,\mathrm{vec}(X) + \mathrm{vec}(C)^T\,\mathrm{vec}(X) \\ & \text{subject to} && X \in \mathcal{P}_N. \end{aligned}$$

Note that the Hessian of the objective function of the QAP is $Q = 2D \otimes F$. One of the useful properties of a Kronecker product [48] is that the eigenvalues of the Kronecker product $D \otimes F$ are the N^2 eigenvalues formed from all possible products of the eigenvalues of F and D. This will be used to compute the smallest and largest eigenvalues of Q in Chapter 4.

1.3 Theoretical Complexity

Sahni and Gonzalez [103] have established the complexity of exactly and approximately solving the QAP. Before we present their results, we introduce the following definition of an ϵ-approximation algorithm and an ϵ-approximation solution for the QAP.

Definition 1.1 *Given $\epsilon > 0$, an algorithm is said to be an ϵ-approximation algorithm for the QAP if for every instance QAP(F, D),*

$$\left| \frac{Z^*(F, D) - \hat{Z}(F, D)}{Z^*(F, D)} \right| \leq \epsilon,$$

where $Z^*(F, D) > 0$ is the optimal objective value and $\hat{Z}(F, D)$ is the objective value of the solution obtained by the algorithm. The solution obtained by an ϵ-approximation algorithm is said to be an ϵ-approximation solution for the QAP.

Theorem 1.1 (Sahni and Gonzalez [103], 1976) *The quadratic assignment problem is NP-complete. For an arbitrary $\epsilon > 0$, there does not exist a polynomial-time ϵ-approximation algorithm for the QAP unless $P = NP$.*

Although there exist polynomially solvable cases [19, 27] of the QAP where special structural conditions hold on the coefficient matrices, Theorem 1.1 implies that in general it is unlikely that a polynomial-time algorithm for the QAP will be found. Furthermore, even finding an ϵ-approximation solution to the QAP is a very difficult problem.

1.3.1 Other NP-Complete Problems as the Special Cases

Several well-known NP-complete problems can be formulated as special cases of the QAP. While the QAP is not a tractable problem, in practice no one would want to use algorithms developed for the QAP to solve those NP-complete problems since the specialized algorithms for solving those problems are far more efficient than any solution method for the QAP. However, the relationship between the QAP and those problems sheds light upon the inherent complexity of the QAP. In the following we present three well-known combinatorial optimization problems as special cases of the QAP.

The Traveling Salesman Problem. We are given a set of N cities and pairwise distances between them, and our task is to find the shortest tour that visits each city exactly once. Let $\{1, 2, \ldots, N\}$ represent the N cities and the $N \times N$ matrix $D = (d_{ij})$ the distances between the cities. We can formulate the traveling salesman problem as the QAP with the distance matrix equal to D and the flow matrix being the adjacency matrix of a cycle on N vertices.

The Graph Partitioning Problem. Consider an edge weighted graph $G = (V, E)$ with $|V| = N$ and an integer k that divides N. We are to partition V into k sets of equal cardinality and meanwhile minimize the total weight of the edges cut by the partition. We can formulate this problem as the QAP with the distance matrix equal to the weighted adjacency matrix of G and the flow matrix being -1 multiplying the adjacency matrix of a graph consisting of k disjoint complete subgraphs each of which has $\frac{N}{k}$ vertices. Note that 1) the sum of the elements of the weighted adjacency matrix of G remains constant, and 2) by formulating the problem as the QAP we effectively maximize the sum of the elements of the weighted adjacency matrix of a union of k disjoint subgraphs of G, each of which is induced by $\frac{N}{k}$ vertices of G. Hence the QAP formulation is equivalent to minimizing the total weight of the edges cut by the partition.

The Maximum Clique Problem. Suppose we have a graph $G = (V, E)$ with $|V| = N$. We would like to find a subset V_1 of the largest cardinality such that $V_1 \subseteq V$ and V_1 induces a clique in G, i.e., all the vertices of V_1 are connected by edges in G. Let us consider the QAP with the distance matrix equal to the adjacency matrix of G and the flow matrix given as -1 multiplying the adjacency matrix of a graph consisting of a clique of size k and $N - k$ isolated vertices. Note that a clique of size k exists only if the QAP has an optimal objective value of $-k^2 + k$, i.e., the sum of the elements of the adjacency matrix of a clique of size k. The maximum clique can be found by solving a series of N QAP instances for $k = 1, 2, \ldots, N$.

1.4 Published QAP Test Problems

Due to the increased research activities in the QAP, several sets of QAP test problems have been proposed by the research community. These test problems provide a platform on which various solution methods for the QAP can be evaluated or compared. In the following we will discuss four sets of QAP test problems that are accessible to the research community. We will use some of these instances to test our proposed algorithm for the QAP in Chapter 5.

QAPLIB [22] Instances. The QAPLIB was first published in 1991 to provide a unified test base for the QAP. This is the most commonly used set of test problems among the researchers of the QAP. It consists of 137 QAP instances with the sizes ranging from

10 to 256. 4 out of the 137 instances have a size greater than 100. A complete list of the QAPLIB instances with the objective values of the optimal or best known solutions is given in Table B.1 in Appendix B.1. The QAPLIB instances and their solution status have been made available via its website at `http://www.seas.upenn.edu/qaplib/`. The online version is updated on a regular basis and contains the most current information of the solution status. As of this writing, except for those with optima known by construction, the largest instances that have been optimally solved have a size of 36.

Instances by Drezner, Hahn and Taillard [30]. Drezner, Hahn and Taillard proposed a total of 142 QAP instances with the sizes ranging from 12 to 729 and whose characteristics are complementary to the QAPLIB instances. According to Drezner, Hahn and Taillard, the QAPLIB instances are difficult for the exact methods but are solved relatively easily by the heuristic methods within a fraction of one percent of the optimal or best known solutions. The instances proposed by Drezner, Hahn and Taillard were constructed in such a way that they are difficult for the heuristic methods but instances of a size up to 75 are solved relatively easily by the exact methods in a reasonable amount of time.

One of the measures for the difficulty of QAP instances is the landscape ruggedness [3], defined as the normalized variance of the difference between two neighboring solutions. The variance is normalized such that the ruggedness is between 0 and 100, with a larger value for a more difficult problem. For some of the instances proposed by Drezner et al., the ruggedness is nearly 100.

Instances by Palubeckis [91]. Palubeckis proposed a set of 10 QAP instances with the sizes ranging from 20 to 200. Limited computational results show that these instances are more difficult for the heuristics than some of the well-known QAPLIB instances.

Instances by Stützle and Fernandes [108]. Stützle and Fernandes proposed a total of 644 instances with the sizes ranging from 50 to 500. This set of instances with more varied characteristics than the other sets were constructed mainly for study of the performance of the heuristic methods.

1.5 Practical Complexity

The advances in computer hardware in the last decade have brought about significant progress in solving some of the most difficult combinatorial optimization problems. The traveling salesman problem is an example. TSP instances with thousands of cities can be solved in practice [27]. However, the QAP is an exception. It is generally considered computationally difficult [6, 16, 22] to solve the QAP of modest sizes, say $N \geq 20$. The QAP instances with $N = 30$ is roughly the current practical limit for the exact solution methods [4, 6]. Several problems from QAPLIB with about $N = 30$ that had been open for decades were only recently solved. For the solution of the QAPLIB instance nug30 an average of 650 computers were used over a one-week period, providing the equivalent of almost 7 years of computation on a single HP9000 C3000 workstation [6]. The CPU time used to solve the QAPLIB instance tho30 is equivalent to over 15 years of computation on a single C3000. Table 1.3 lists the years in which the problems were exactly solved and the CPU time spent for nearly a dozen recently solved large QAP instances from the QAPLIB.

Table 1.3 Recently Solved Large QAPLIB Instances

Problem	Year Solved[3]	CPU Days[4]
kra30a	2000	99
kra30b	2000	1527
kra32	2000	5536
nug27	2000	113
nug28	2000	722
nug30	2000	3999
ste36a	2001	18
ste36b	1999	60
ste36c	1999	200
tai25a	2003	394
tho30	2000	8997

Note that most of these successes were achieved in the distributed computing environments where massive networks of computers were utilized to meet the daunting

[3]These data are excerpted from the QAPLIB website.

[4]These data are due to Anstreicher [4].

computational demand.

1.6 Asymptotic Behavior

The QAP exhibits an interesting asymptotic behavior. As outlined below, if certain probabilistic conditions on the problem data are satisfied, the ratio of its best and worst objective values approaches 1 as the problem size goes to infinity. On one hand, this behavior suggests that the error of any heuristic method vanishes as the problem size tends to infinity. That is, if the problem size is large enough, the QAP becomes a trivial problem in the sense that every heuristic method finds an almost optimal solution. On the other hand, as the problem size increases, all the feasible solutions to the QAP are clustered between two level sets of the objective function that become arbitrarily close. This renders an exact algorithm such as the branch and bound method very inefficient. In this situation the branch and bound method tends to enumerate a majority of the nodes in the branch and bound tree unless arbitrarily tight lower bounds are available. This behavior will also cause difficulty for the continuous optimization methods that we use in this research. Due to the clustering of the feasible points as the problem size increases, our algorithm using the continuous optimization methods may not be able to "distinguish" between the good and bad solutions and hence may fail to find a good solution.

In an early work Burkard and Fincke [20] studied the difference between the best and worst objective values of some special cases of the QAP. They first considered the planar QAP, where the distance matrix consists of pairwise distances between the points chosen independently and uniformly from the unit square in the plane and the flows are independent random variables on $[0, 1]$. Then they investigated the QAP where both the flows and the distances are independent random variables on $[0, 1]$. In both cases they showed that the difference between the best and worst objective values approaches 0 with a probability tending to 1 as the problem size goes to infinity.

Later Burkard and Fincke [21] investigated the asymptotic behavior of a generic combinatorial optimization problem. They considered a sequence P_N, $N = 1, 2, \ldots$, of minimization problems with a sum objective function as described below. For each positive integer N, P_N was given by specifying a ground set $\mathcal{S}_N$, a set $\mathcal{F}_N \subseteq 2^{\mathcal{S}_N}$ of feasible solutions, and a nonnegative cost function $c_N : \mathcal{S}_N \mapsto \mathbb{R}^+$. The objective function

$f : \mathcal{F}_N \mapsto \mathbb{R}$ was defined as

$$f(X) = \sum_{x \in X} c_N(x) \tag{1.13}$$

for all $X \in \mathcal{F}_N$. Burkard and Fincke showed that the ratio of the best and worst objective values of (1.13) converges to 1 in probability as $N \to \infty$. Szpankowski [109] improved the order of convergence by showing that the convergence holds almost surely. In the almost sure convergence the probability that the ratio of the best and worst objective values approaches 1 is equal to 1.

Çela [27] showed that the QAP is a special case of combinatorial optimization problems with a sum objective function. More specifically, it was shown that the ground set is $\mathcal{S}_N = \{(i,j,k,l) : 1 \leq i,j,k,l \leq N\}$. Each permutation π of $\mathcal{N}$ corresponds to a feasible solution X_π as a subset of $\mathcal{S}_N$, $X_\pi = \{(i,j,\pi(i),\pi(j)) : 1 \leq i,j \leq N\}$. The set $\mathcal{F}_N$ of the feasible solutions consists of all feasible solutions X_π for $\pi \in \Pi_N$. The cost function is given as $c_N(i,j,k,l) = f_{ij}d_{kl}$. Based on the work of Burkard and Fincke [21] and Szpankowski [109] for general combinatorial problems, Çela obtained the following theorem.

Theorem 1.2 (Çela [27], 1998) *Consider a sequence of QAP(F^N, D^N) with $N \times N$ flow and distance matrices $F^N = (f_{ij}^N)$ and $D^N = (d_{ij}^N)$. Assume that for $M > 0$ the f_{ij}^N and, distinctly, the d_{ij}^N are identically and independently distributed random variables on $[0, M]$ with finite first, second, and third moments. Let $Z_{\min}$ and $Z_{\max}$ denote the optimal and worst objective values of QAP(F^N, D^N), respectively:*

$$Z_{\min} = \min_{\pi \in \Pi_N} \sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij}^N d_{\pi(i)\pi(j)}^N$$

$$Z_{\max} = \max_{\pi \in \Pi_N} \sum_{i=1}^{N} \sum_{j=1}^{N} f_{ij}^N d_{\pi(i)\pi(j)}^N$$

Then the following holds almost surely:

$$\lim_{N \to \infty} \frac{Z_{\min}}{Z_{\max}} = 1.$$

Several other authors obtained different analytical forms of the asymptotic behavior of the QAP by imposing slightly different probabilistic conditions. The readers are referred to Frenk, Houweninge and Rinnooy Kan [37] and Rhee [99, 100] for more details.

Frenk, Houweninge and Rinnooy Kan [37] conduct computational experiments to empirically verify the asymptotic property of the QAP. Their results confirm that the ratio converges to 1 relatively quickly. For the examples they use, the ratio falls within 0.1 of its theoretical value from approximately $N = 50$ onwards.

1.7 Synopsis of the Remaining Chapters

In Chapter 2 we will survey the literature on the solution approaches for nonlinear 0-1 programming. As special cases of nonlinear 0-1 programming, we will also survey the solution methods specific for quadratic 0-1 programming and the QAP. In Chapter 3 we will study the methods of relaxation with penalty functions. We will discuss the properties of weak and strong equivalence of the relaxation using two penalty functions. The asymptotic properties of the relaxation using the quartic penalty function will also be discussed. Based upon the method of relaxation using the quadratic penalty function, we will propose an algorithm for the QAP in Chapter 4. We will describe two important techniques used in our algorithm, i.e., the convex transformation of the objective function of the QAP and the pre-conditioning the Hessian of the objective function. Then we will show how quadratic cuts can be applied to improve the solutions to the QAP. In Chapter 5 we will describe the implementation details of our algorithm for the QAP. Then we will show and discuss the numerical results of our algorithm on the QAPLIB instances and other published QAP test problems. We will draw conclusions and discuss possible extensions of our algorithm for the QAP to quadratic 0-1 programming and general nonlinear 0-1 programming in Chapter 6.

Chapter 2

Nonlinear 0-1 Programming

In this chapter we will present a brief overview of formulations of nonlinear 0-1 programming and its main solution approaches reported in the literature. The quadratic assignment problem, the focus of this dissertation, is a special case of quadratic 0-1 programming which in turn is a subclass of nonlinear 0-1 programming. Hence in principle we could solve the quadratic assignment problem as a nonlinear 0-1 programming problem. We will also give a brief summary of the extensive literature in the solution techniques for quadratic 0-1 programming. Last we will discuss the methods to solve the quadratic assignment problem.

2.1 Formulations

To simplify the notation but without loss of generality, we consider a nonlinear 0-1 programming problem of the following form

$$\begin{aligned} (2.1) \qquad & \text{minimize} && f(x) \\ & \text{subject to} && g(x) \leq 0 \\ & && x \in \mathbb{B}^n \end{aligned}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$, and $\mathbb{B} = \{0, 1\}$. Since any equality constraint can be replaced by two equivalent inequality constraints, this form includes the cases with equality constraints and the QAP can be viewed as a special case of it.

Problem (2.1) is important in its own right. For a list of references to the applications of nonlinear 0-1 programming, see Balas and Mazzola [9], Boros and Hammer [15], and Hansen et al. [60]. In addition, an integer or mixed-integer nonlinear programming problem with bounded variables can be transformed into an equivalent nonlinear 0-1

programming problem by binary expansions. In particular, we can replace an integer variable y with bounds $l \leq y \leq u$ by the following:

$$y = l + x_1 + 2x_2 + 4x_3 + \cdots + 2^{K-1}x_K$$

where $x_i \in \mathbb{B}$ for $i = 1, \ldots, K$ and K is given by

$$K = \lfloor \log_2(u - l) \rfloor + 1.$$

It is shown in Hammer and Rudeanu [58] that any function in 0-1 variables can be reduced to a polynomial in the same variables. Indeed, if we set

$$\begin{aligned} (2.2) \qquad & \Delta_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \\ & \quad = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n), \quad \text{and} \\ (2.3) \qquad & \eta_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n), \end{aligned}$$

it is easy to verify that

$$(2.4) \qquad f(x_1, \ldots, x_n) = x_i \Delta_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) + \eta_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

Applying (2.4) recursively, we can reduce (2.1) to the following polynomial 0-1 programming problem:

$$(2.5) \qquad \begin{aligned} \text{minimize} \quad & f(x) = \sum_{k=1}^{p_0} c_k \prod_{j \in \mathcal{N}_k} x_j \\ \text{subject to} \quad & \sum_{k=1}^{p_i} a_{ik} \prod_{j \in \mathcal{N}_{ik}} x_j \leq b_i, \quad i = 1, 2, \ldots, m \\ & x \in \mathbb{B}^n \end{aligned}$$

where $\mathcal{N}_k \subseteq \bar{\mathcal{N}} = \{1, 2, \ldots, n\}, k = 1, 2, \ldots, p_0$ and $\mathcal{N}_{ik} \subseteq \bar{\mathcal{N}}, k = 1, 2, \ldots, p_i, i = 1, 2, \ldots, m$. Note that the property $x_j^2 = x_j, \forall j$, implies all variables have a power of 1 in the terms in which they appear. (2.5) is also called multilinear 0-1 programming.

Moreover, Rosenberg [101] has shown that any unconstrained polynomial 0-1 programming problem $\text{minimize}_{x \in \mathbb{B}^n} f(x)$ can be reduced to the quadratic case. We replace all occurrences of a product of two variables $x_i x_j$ appearing in the terms of order greater than two in $f(x)$ by a new variable x_{n+l} and then add the following corrective term in the objective function

$$\mu(x_i x_j + (3 - 2x_i - 2x_j)x_{n+l})$$

for some $\mu > \tilde{f} - \underline{f}$ where $\tilde{f}$ is the value of $f(x)$ at some $x \in \mathbb{B}^n$ and $\underline{f}$ is a lower bound on $f(x)$ on $\mathbb{B}^n$, given as $\underline{f} = \sum_{k=1}^{p_0} \min(0, c_k)$. Note that $(x_i x_j + (3 - 2x_i - 2x_j)x_{n+l}) = 0$ for $x_{n+l} = x_i x_j$ and $(x_i x_j + (3 - 2x_i - 2x_j)x_{n+l}) \geq 1$ otherwise. Hence any solution for which $x_{n+l} \neq x_i x_j$ cannot be optimal. Applying the above reduction recursively leads to an unconstrained quadratic 0-1 programming problem. Note that such a reduction scheme does not apply to constrained cases. Also the number of new variables increases rapidly with the order of the multilinear terms.

2.2 General Solution Approaches

Various solution approaches have been proposed for nonlinear 0-1 programming problems. We will follow the outline of Hansen et al. [60] and give a brief introduction to the four main approaches, i.e., linearization, algebraic, enumerative, and cutting-plane methods, of which enumerative methods appear to be the most efficient [60]. The purpose of the survey in nonlinear 0-1 programming is not to conduct an exhaustive review on the subject but rather to shed light on the study of the QAP. Hence, we will focus on the basic ideas of the four approaches while skipping the elaborate details of their variants and extensions. For more details of these approaches, the reader is referred to Balas and Mazzola [8, 9], Boros and Hammer [15], Hansen [59], and Hansen et al. [60]. We will postpone discussing the continuous optimization approach in which 0-1 problems are transformed into equivalent continuous optimization problems via relaxation and using penalty functions until Chapter 3.

2.2.1 Linearization Methods

Fortet [35] (see also Watters [115][1]) has shown that a polynomial 0-1 programming problem can be reduced to a linear 0-1 programming problem by replacing each distinct product $\prod_{j \in \mathcal{N}_k} x_j$ by a new 0-1 variable x_{n+k} and adding two new constraints

$$\sum_{j \in \mathcal{N}_k} x_j - x_{n+k} \leq |\mathcal{N}_k| - 1, \quad \text{and} \tag{2.6}$$

$$-\sum_{j \in \mathcal{N}_k} x_j + |\mathcal{N}_k| x_{n+k} \leq 0. \tag{2.7}$$

[1]According to Hansen et al. [60] Watters independently rediscovered the linearization.

The constraint (2.6) implies that $x_j = 0$ for some $j \in \mathcal{N}_k$ if $x_{n+k} = 0$ and $x_{n+k} = 1$ if $x_j = 1$ for all $j \in \mathcal{N}_k$. The constraint (2.7) implies that $x_{n+k} = 0$ if and only if $x_j = 0$ for some $j \in \mathcal{N}_k$ and $x_{n+k} = 1$ if and only if $x_j = 1$ for all $j \in \mathcal{N}_k$.

The above scheme can potentially introduce a large number of new variables and new constraints. Glover and Woolsey [46] proposed several rules to obtain smaller sets of constraints that are equivalent to (2.6) and (2.7). In another paper [47], they introduced new continuous variables $y_k \in [0, 1]$ which automatically take the value 0 or 1 in any feasible solution instead of the 0-1 variables x_{n+k}. Nevertheless, the increase in the number of variables and number of constraints makes solution techniques based on their approaches problematic at best.

2.2.2 Algebraic Methods

The algebraic methods proposed in the literature usually apply to the unconstrained cases. The most well-known algebraic method is the *Basic Algorithm* [56, 58]. The method exploits the local optimality condition as follows. Setting $i = 1$ in (2.4) we have the following

$$f(x_1, \ldots, x_n) = x_1 \Delta_1(x_2, \ldots, x_n) + \eta_1(x_2, \ldots, x_n) \tag{2.8}$$

where $\Delta_1(\cdot)$ and $\eta_1(\cdot)$ are defined in (2.2) and (2.3). For an optimal solution $(x_1^*, \ldots, x_n^*)$, we have

$$f(x_1^*, x_2^*, \ldots, x_n^*) \leq f(\bar{x}_1^*, x_2^*, \ldots, x_n^*)$$

where $\bar{x}_1^* = 1 - x_1^*$. Hence

$$(x_1^* - \bar{x}_1^*)\Delta_1(x_2^*, \ldots, x_n^*) \leq 0$$

is a necessary condition for optimality. This condition can be used to eliminate x_1 from (2.8) as follows. We define

$$\psi_1(x_2, \ldots, x_n) = \begin{cases} \Delta_1(x_2, \ldots, x_n) & \text{if } \Delta_1(x_2, \ldots, x_n) < 0; \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that minimizing $f(x_1, \ldots, x_n)$ is equivalent to minimizing

$$\psi_1(x_2, \ldots, x_n) + \eta_1(x_2, \ldots, x_n).$$

Continuing such an elimination process for $x_2, \ldots, x_{n-1}$ yields a sequence of function $\psi_1, \psi_2, \ldots, \psi_{n-1}$. The minimum of f can be traced back from the minimum x_n^* of $\psi_{n-1}(x_n) + \eta_{n-1}(x_n)$.

Extensions to this method allow for the solution of constrained problems through the transformation of the constrained problems into the equivalent unconstrained problems using penalty functions [58].

The efficiency of this approach depends critically on how $\psi_1, \psi_2, \ldots, \psi_{n-1}$ are obtained. Computationally, determining these functions is generally intractable for problems of realistic sizes [15].

2.2.3 Enumerative Methods

One of the earliest enumerative methods is a *lexicographical enumeration* algorithm proposed by Lawler and Bell [75] for constrained nonlinear 0-1 programming problems with a monotone objective function. Later Mao and Wallingford [78, 29] extended their algorithm to the cases with a general objective function. These algorithms are equivalent to branch-and-bound methods with rigid and problem-independent branching rules; i.e., the branching variables are chosen according to some *a priori* ordering. Many branch-and-bound algorithms for unconstrained and constrained nonlinear 0-1 programming were proposed in the late 1960s and the 1970s. The branch-and-bound algorithms obtain better results than the lexicographical enumeration algorithms by choosing the branching variables according to rules based on the problem structures. The reader is referred to Hansen et al. [60] for a list of references to branch-and-bound algorithms and a framework of these algorithms.

A branch-and-bound algorithm uses a branching rule to split to problem to be solved into smaller subproblems. The most common branching rules are depth-first search and best-first search. In the former, we branch on the deepest node in the branch-and-bound tree if possible; backtracking consists of finding the last branching variable for which a branch remains unexplored. In the latter, the subproblem with the smallest lower bound on the objective value is selected. Bounds are computed on the objective value or constraint left-hand side values and used in tests to curtail the enumeration. A penalty is computed for an unfixed variable as an increment that may be added to a bound if it is fixed at 0 or 1. Tests based on the penalties are performed to determine the variables that must take a value of 0 or 1 in all feasible or optimal solutions.

Once the framework has been laid out, a particular branch-and-bound algorithm is based upon how the bounds and penalties are computed. Many schemes to compute bounds and penalties for nonlinear 0-1 programming have been proposed. The quality of the bounds and penalties usually improves with the amount of computation required to obtain them. There is a trade-off between the quality of the bounds and penalties and the expense required to obtain them.

2.2.4 Cutting-Plane Methods

The original cutting-plane method for constrained nonlinear 0-1 programming is due to Granot and Hammer [50]. In their algorithm, the nonlinear 0-1 programming problem is transformed into an equivalent generalized covering problem. The latter problem is solved as a linear 0-1 programming problem. Unfortunately, the number of generalized covering constraints thus obtained can be very large, making the linear 0-1 programs difficult to solve. To overcome such difficulty, Granot and Granot [49] proposed a cutting-plane algorithm in which only a small set of generalized covering inequalities are generated in the initial generalized covering problem. The initial generalized covering problem is a relaxation of the original 0-1 problem, referred to as the *generalized covering relaxation* (GCR). In this algorithm, the GCR is first solved and the optimal solution x^* obtained. If x^* is infeasible to the original problem, a new generalized covering inequality is generated for each constraint violated by x^*. The new inequalities are added to the GCR and the resulting GCR is re-solved. Otherwise, the original problem has been solved.

The algorithm of Granot and Granot can be improved in several ways. First, the GCR can be solved by a heuristic. Second, some inequalities can be discarded to maintain a moderate size of the GCR. Finally, stronger cuts can be derived for the GCR. The papers of Balas and Mazzola [8, 9] contain the details of this algorithm.

2.3 Quadratic 0-1 Programming

In this section, we will survey the solution methods for quadratic 0-1 programming problems. In particular, we will consider the linearly constrained cases of the following form:

$$\begin{aligned} (2.9) \qquad & \text{minimize} && \tfrac{1}{2}x^T Q x + c^T x \\ & \text{subject to} && Ax \leq b \\ & && x \in \mathbb{B}^n \end{aligned}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric matrix, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^m$. When Q is not symmetric, we can always replace Q by $\bar{Q} = \frac{1}{2}(Q + Q^T)$. Note that by the same argument as in nonlinear 0-1 programming (2.9) includes the cases where equality constraints $Ax = b$ are present. In the following, we will not consider the methods for solving unconstrained quadratic 0-1 programming problems, which the reader will find abound in the literature. Neither will we discuss algorithms that specialize in the quadratic knapsack problems, an important application of quadratic 0-1 programming. We will outline the main solution approaches for the quadratic assignment problem in section 2.4.

Existing methods for solving (2.9) primarily fall into two classes. In the first class, the problem is transformed into an equivalent linear 0-1 integer problem and then solved while in the second class the nonlinear objective function is dealt with directly through some enumerative scheme. For the linearization techniques, the transformations for general nonlinear 0-1 programming discussed in section 2.2.1 are readily applicable to quadratic 0-1 programming problems. In addition, Glover [45] proposed a perhaps most compact mixed integer linear reformulation of quadratic 0-1 programming problems. As McBride and Yormark [80] indicated, the linearization techniques do not yield very efficient solution methods for the problem (2.9) due to the increase in the number of variables and number of constraints. For example, in the worst case where all the cross-product quadratic terms are present, using the linearization technique in (2.6) and (2.7) requires $n(n-1)/2$ additional 0-1 variables and $n(n-1)$ additional constraints. Although the alternative linearizations yield more compact formulations, convergence is still slow.

The enumeration methods have proved to be more attractive for solving (2.9). Among the earlier enumerative schemes are Mao and Wallingford [78], and Laughhunn [73]. Mao and Wallingford extended Lawler and Bell's lexicographical enumeration al-

gorithm [75] to handle quadratic 0-1 programming problems whose objective function is not monotonic. Laughhunn proposed a Balas-type [7] algorithm using some simple bounding penalties based on the objective function. As McBride and Yormark [80] also pointed out, these earlier enumerative methods are not effective for problems of practical size. More recent research has favored branch-and-bound methods whose performance hinges on good lower bounds on the objective function. Hence obtaining tight bounds on the objective function has been the focus in the study of branch-and-bound methods. The most frequently used approaches include linearization techniques, Lagrangian decomposition methods, and, more recently, semidefinite relaxations.

To obtain a bound via linearization, the techniques described in section 2.2.1 can be used. However, those techniques do not produce good bounds [2]. Adams and Sherali [2] proposed a linearization scheme specifically for quadratic 0-1 programming problems. It was shown that their linearization yielded predominantly tighter bounds than the other linearizations found in the literature.

Lagrangian decomposition methods provide a different approach for computing lower bounds. Michelon and Maculan [83] applied Lagrangian decomposition to study nonlinear integer programming problems. Solving the dual Lagrangian relaxation requires the solution of a continuous nonlinear programming problem and an integer linear programming problem at each iteration. Later, Michelon and Maculan [84] studied the Lagrangian decomposition for the problem (2.9). They showed that the solution to the continuous quadratic programming problem can be expressed in a closed form. Hence the dual problem was easier to solve in that only one subproblem had to be solved in order to compute the objective function of the dual problem. Elloumi et al. [31] presented a different decomposition for quadratic 0-1 programming with linear constraints and showed that it compared favorably to the linearization of Adams and Sherali [2].

A quite recent method, often yielding tight bounds, is the semidefinite relaxation for equality constrained quadratic 0-1 programming [95]. In this approach, the linear equality constraints are replaced by the squared norm constraints to obtain an equivalent quadratically constrained 0-1 problem. The Lagrangian dual problem of the quadratically constrained problem is formed and the Lagrangian is homogenized to obtain a pure quadratic form. Then the products of variables are replaced by new variables which form a symmetric, rank one, and positive semidefinite matrix. Using hidden semidefinite constraints in the Lagrangian dual problem, a semidefinite program is obtained. Taking the

Lagrangian dual of the resulting semidefinite program leads to the semidefinite relaxation of the original problem. Some promising results have been reported for this approach.

2.4 Methods for the Quadratic Assignment Problem

Branch-and-bound and *cutting-plane* methods are primarily the two types of algorithms that have been used to solve the quadratic assignment problem to optimality. Branch-and-bound algorithms have been the more successful of the two for the QAP, outperforming cutting-plane algorithms whose running time is simply too long [27]. In the following we will briefly discuss cutting-plane methods. We will describe in detail branch-and-bound methods for the QAP in section 2.4.1.

There are two classes of cutting-plane methods: traditional cutting-plane methods and polyhedral cutting-plane or *branch-and-cut* methods. Traditional cutting-plane algorithms have been developed by Balas and Mazzola [9], Bazaraa and Sherali [11, 12], and Kaufmann and Broeckx [69]. These algorithms make use of mixed integer linear programming formulations for the QAP which are well suited for Benders' decomposition. They either solve the QAP to optimality or compute a lower bound. The computational experience for polyhedral cutting-plane methods is still very limited, due to lack of good understanding of the combinatorial structure of the QAP polytope [27]. Recently, encouraging results [90, 65] in polyhedral cuts have been obtained, leading one to believe that polyhedral cutting-plane algorithms can be used to solve reasonably sized QAP instances in the future.

2.4.1 Branch-and-Bound Methods

In a branch-and-bound method for the QAP, the algorithm starts with an empty permutation with no facility assigned to any location and successively expands it to a full permutation in which all the facilities are assigned to the locations. The important components for a branch-and-bound algorithm for the QAP are the *branching rule*, *selection rule*, and *lower bounding technique*.

There are three common types of branching rules: *single assignment* [44, 74], *pair assignment* [39, 72, 88], and *relative positioning* [85]. In the single assignment branching, the algorithm assigns a single, not yet allocated facility i to an unoccupied location j

at each branching step. The pair assignment rule allocates a pair of facilities to a pair of locations at each branching step. In the relative positioning branching, the levels of branch-and-bound tree do not correspond to the assignments of facilities to locations. The partial permutations at each level are determined in terms of the distances between facilities, i.e., their relative positions. Numerical results show that the single assignment branching rule outperforms the pair assignment or relative positioning branching rules [19, 92]. The choice of the facility-location pair (i, j) is made according to the selection rule. Several rules have been proposed by different authors [10, 17, 79]. The appropriate rule usually depends on the bounding technique employed.

2.4.2 Lower Bounds

The performance of branch-and-bound algorithms depends critically on the quality of the lower bounds. Many efforts have been made to derive tight and yet computationally efficient lower bounds. In this section we will briefly describe five bounding techniques for the QAP: Gilmore-Lawler and related lower bounds, lower bounds based on linear programming relaxations, eigenvalue based lower bounds, lower bounds based on semidefinite relaxations, and convex quadratic programming bounds.

Gilmore-Lawler and Related Lower Bounds. One of the first lower bounds for the QAP was derived by Gilmore [44] and Lawler [74]. For QAP(F, D) of size N, we define a $N \times N$ matrix $C = (c_{ij})$ by

$$c_{ij} \triangleq \min_{\substack{\pi \in \Pi_N: \\ \pi(j)=i}} \sum_{k=1}^{N} f_{i\pi(k)} d_{jk}, \quad i, j = 1, \ldots, N.$$

It is well known that the entries c_{ij} can be easily computed by sorting vectors F_i and D_j in increasing and decreasing order respectively, where F_i denotes the i-th row of matrix F and D_j denotes the j-th row of matrix D. It is easy to see that the following holds for $\pi \in \Pi_N$:

$$Z(F, D, \pi) = \sum_{i=1}^{N} \sum_{j=1}^{N} f_{\pi(i)\pi(j)} d_{ij} \geq \sum_{i=1}^{N} F_{\pi(i)} D_i^T \geq \sum_{i=1}^{N} c_{\pi(i)i}. \tag{2.10}$$

From (2.10) we have

$$\min_{\pi \in \Pi_N} Z(F, D, \pi) \geq \min_{\pi \in \Pi_N} \sum_{i=1}^{N} c_{\pi(i)i} \triangleq GLB,$$

where GLB is the Gilmore-Lawler lower bound for QAP(F, D). After matrix C is computed, it takes $O(n^3)$ time to compute GLB by solving a linear assignment problem. Hence the overall complexity for computing the Gilmore-Lawler bound is $O(n^3)$.

The Gilmore-Lawler bound is one of the simplest and cheapest bounds to compute, but unfortunately this bound is not tight and, in general, the gap between the Gilmore-Lawler bound and the optimal objective value increases quickly as the problem size increases. Various reduction schemes have been proposed to improve the quality of the Gilmore-Lawler bound by transforming the coefficient matrices F and D. Li et al. [77] proposed an improvement on the bound via a variance reduction scheme. Reduction schemes based on reformulations [24, 25] and dual formulations [54, 55] have also been proposed.

Lower Bounds Based on Linear Programming Relaxations. Several authors have proposed mixed integer linear programming (MILP) formulations for the QAP. In the context of lower bound computations two MILP formulations are of importance. Frieze and Yadegar [38] replaced the products $x_{ik}x_{jl}$ of 0-1 variables by continuous variables y_{ijkl} and introduced extra linear constraints to obtain an MILP formulation for the QAP. To obtain a lower bound, Frieze and Yadegar derived a Lagrangian relaxation of their MILP formulation and solved it approximately by applying subgradient optimization techniques. They showed that the resulting bounds are better than the Gilmore-Lawler bounds.

Based upon the MILP formulation of Frieze and Yadegar, Adams and Johnson [1] obtained a slightly more compact MILP formulation:

$$
\begin{aligned}
(2.11) \qquad \text{minimize} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} f_{ij}d_{kl}y_{ijkl} \\
\text{subject to} \quad & \text{(1.3a)–(1.3c)} \\
& \sum_{i=1}^{N} y_{ijkl} = x_{jl}, \quad j,k,l = 1,\ldots,N \\
& \sum_{j=1}^{N} y_{ijkl} = x_{ik}, \quad i,k,l = 1,\ldots,N \\
& y_{ijkl} = y_{klij}, \quad i,j,k,l = 1,\ldots,N \\
& y_{ijkl} \geq 0, \quad i,j,k,l = 1,\ldots,N.
\end{aligned}
$$

They showed that the continuous relaxation of (2.11) is tighter than the continuous relaxation of the formulation of Frieze and Yadegar. Moreover, many of the previously published lower bounding techniques can be explained based upon the Lagrangian dual of this relaxation.

Eigenvalue Based Lower Bounds. These bounds are based on the relationship between the objective value of the QAP in the trace formulation and the eigenvalues of the flow and distance matrices. If designed and implemented prudently, these bounding techniques can produce bounds of better quality than the Gilmore-Lawler bounds or bounds based on linear programming relaxations. However, the eigenvalue based bounds are expensive to compute and hence are not appropriate for use as bounding techniques in branch-and-bound algorithms. Eigenvalue based bounds for the QAP with symmetric matrices are proposed by several authors [33, 52, 53, 98].

Lower Bounds Based on Semidefinite Relaxations. Semidefinite programming (SDP) relaxations for the QAP were studied by Karisch [68], Zhao [117], and Zhao et al. [118]. In their papers, interior-point methods or cutting-plane methods are used to solve the SDP relaxations to obtain lower bounds for the QAP. These solution methods require at least $O(n^6)$ time per iteration. In terms of quality the bounds they obtained are competitive with the best existing lower bounds for the QAP. For many QAPLIB instances, such as those of Hadley et al. [53], Nugent et al. [88] and Taillard [110, 111], semidefinite relaxations provide the best existing bounds. However, the prohibitively high computation requirements makes the use of such approaches impractical.

Convex Quadratic Programming Bounds. The quadratic programming bound [5] for QAP(F, D) is defined as the optimal objective value of the following quadratic programming problem:

$$\text{(QPB)} \qquad \begin{array}{ll} \text{minimize} & \operatorname{vec}(X)^T Q \operatorname{vec}(X) + \gamma \\ \text{subject to} & Xe = X^T e = e \\ & X \geq 0, \end{array}$$

where $Q = (D \otimes F) - (I \otimes S) - (T \otimes I)$. The matrices S and T and constant γ are obtained from the spectral decompositions of $V^T F V$ and $V^T D V$, where V is an $N \times (N-1)$ matrix

with orthonormal columns such that $e^T V = 0$. The objective function of (QPB) is convex on the nullspace of the equality constraints, so computing the quadratic programming bound requires solving a convex quadratic programming problem. In [16] the Frank-Wolfe (FW) algorithm [36, 82] is proposed to approximately solve (QPB). Although the FW algorithm is known for its poor asymptotic performance, in this context it is attractive because the computation required at each iteration is dominated by the solution of a single, small linear assignment problem. It is worth noting that this bounding technique was used to solve several very difficult QAP instances in Table 1.3 to optimality.

Chapter 3

Methods of Relaxation with a Penalty Function

3.1 Introduction

In Chapter 2 we outlined several solution approaches to nonlinear 0-1 programming problems that use discrete optimization techniques. In this chapter we will discuss a class of algorithms for a nonlinear 0-1 programming problem that are based upon the continuous optimization techniques. In this class of algorithms the integrality constraints on the variables of the 0-1 optimization problem are relaxed to obtain a continuous optimization problem. A penalty function is introduced to force the solutions to the continuous relaxation to be integer. Then the continuous optimization problem is solved to obtain an optimal or good integer solution to the original 0-1 problem.

Consider the nonlinear 0-1 programming problem of (2.1). We replace $\mathbb{B}^n$ with $\mathcal{X}$ and obtain the following *relaxation*:

$$\begin{aligned} (3.1) \qquad & \text{minimize} && f(x) \\ & \text{subject to} && g(x) \leq 0 \\ & && x \in \mathcal{X} \end{aligned}$$

where $\mathcal{X}$ is some path-connected set with $\mathbb{B}^n \subset \mathcal{X} \subseteq \mathbb{R}^n$ and $f(x)$ is bounded below on $\mathcal{X}$. Note that the global optimal solution to (3.1) provides a lower bound for the optimal solution to (2.1). In general this lower bound is not equal to the minimum of (2.1); the global solution to (3.1) may not be integer. In continuous optimization approaches, it is standard to introduce a suitable penalty term to the objective function to force the optimal solutions to the continuous relaxation to be integer. Thus we consider the

following *relaxation with a penalty function*:

$$\begin{aligned} (3.2) \qquad & \text{minimize} && f(x) + \mu\Phi(x) \\ & \text{subject to} && g(x) \leq 0 \\ & && x \in \mathcal{X} \end{aligned}$$

where $\Phi : \mathbb{R}^n \mapsto \mathbb{R}$ and $\mu > 0$ is the penalty parameter.

In order to find an optimal solution, or at least a good integer solution, to (2.1) by solving (3.2) the following properties are desirable. For all values of μ greater than some finite μ_0:

(A1) $\Phi(x) = 0$ for $x \in \mathbb{B}^n$ and $\Phi(x) > 0$ for $x \in \mathcal{X} \backslash \mathbb{B}^n$.
(A2) Every 0-1 feasible solution to (2.1) is a local minimum of (3.2).
(A3) The global minimum of (3.2) occurs at a 0-1 point.
(A4) The converse of (A2) is true; i.e., every local minimum of (3.2) is a 0-1 feasible solution to (2.1).

Given an appropriate value of μ_0, the following implications are a consequence of these properties. Property (A1) ensures that (2.1) and (3.2) have the same objective values at $x \in \mathbb{B}^n$. With property (A2) every 0-1 feasible solution to (2.1) can be found by solving its relaxation if we start close enough to the 0-1 solution. Note that this property does not preclude the existence of noninteger local solution to (3.2). Property (A3) together with (A1) enables a global optimization algorithm, if available, for the continuous relaxation to find the (global) optimum of the original 0-1 problem. Property (A4) guarantees that we can always find a 0-1 feasible solution to (2.1) by solving (3.2). Note that in absence of property (A4), property (A3) is needed for a global solution to (3.2) to be optimal for (2.1) whereas properties (A2) and (A4) together imply property (A3). Based upon whether property (A4) is satisfied or not, we have the following two definitions.

Definition 3.1 (*Weak Equivalence*)

A nonlinear 0-1 programming problem (2.1) and its relaxation (3.2) are said to be weakly equivalent if properties (A1)–(A3) hold for all $\mu > \mu_0$.

Definition 3.2 (*Strong Equivalence*)

A nonlinear 0-1 programming problem (2.1) and its relaxation (3.2) are said to be strongly equivalent if properties (A1), (A2), and (A4) hold for all $\mu > \mu_0$.

To solve (3.2) globally and hence obtain the optimal solution to (2.1), it suffices that only weak equivalence holds. In this case, there are two major difficulties with a continuous optimization approach. First, we need to determine the threshold value μ_0 such that for all $\mu > \mu_0$ weak equivalence holds. Except for some special cases, μ_0 is generally unknown for a nonlinear 0-1 programming problem. Second, even in the cases where μ_0 is known, we still need to solve a continuous global optimization problem that, when there are multiple local solutions, can be extremely difficult.

Hence we have to be content with finding a good 0-1 solution to (2.1) by solving (3.2) locally. We face several challenges. First, we have the same issue of determining the threshold value μ_0 as above. In the cases where μ_0 is unknown, we need to solve a sequence of optimization problems of form (3.2) with an increasing μ until we find a 0-1 solution or determine that the algorithm fails to find a 0-1 solution. Second, in the cases where only weak equivalence holds, there are local minima to (3.2) which are not 0-1 feasible solutions to (2.1). Thus to guarantee that a continuous optimization approach finds a 0-1 solution to (2.1), an algorithm for solving (3.2) must avoid the non-integer local minima. Because of the existence of non-integer local minima in the above cases, choosing $\Phi(x)$ such that strong equivalence holds is usually desirable. Under strong equivalence, we can always find a 0-1 feasible solution to (2.1) by solving (3.2). However, only a certain class of nonlinear 0-1 programming problems with a specific penalty function possess the strong equivalence properties.

We introduce two penalty functions that have been used in the literature for 0-1 problems. It is obvious that the 0-1 constraints on the variables $x \in \mathbb{B}^n$ are equivalent to

$$\begin{cases} x^T(e-x) = 0 \\ 0 \le x \le e \end{cases}$$

or

$$x \circ (e-x) = 0$$

where $\circ$ denotes the Hadamard product or component-wise product of two vectors, and e is the vector of all ones. Setting $\Phi(x) = x^T(e-x)$ and $\mathcal{X} = \{x \in \mathbb{R}^n : 0 \le x \le e\}$, we obtain the relaxation:

$$\begin{aligned} (3.3) \qquad & \text{minimize} && f(x) + \mu x^T(e-x) \\ & \text{subject to} && g(x) \le 0 \\ & && 0 \le x \le e. \end{aligned}$$

Alternatively, we can set $\Phi(x) = \|x \circ (e - x)\|^2$ to obtain the following:

$$\begin{aligned} (3.4) \qquad & \text{minimize} && f(x) + \mu\|x \circ (e - x)\|^2 \\ & \text{subject to} && g(x) \leq 0 \\ & && x \in \mathcal{X} \end{aligned}$$

where $\|\cdot\|$ denotes the ℓ_2 or Euclidean norm of a vector, $\mathbb{B}^n \subset \mathcal{X} \subseteq \mathbb{R}^n$ and $f(x)$ is bounded below on $\mathcal{X}$. It will be shown that weak equivalence holds between (2.1) and (3.3) under some mild conditions on $f(x)$ whereas weak equivalence between (2.1) and (3.4) does not hold in general. Furthermore, it can be shown that strong equivalence holds for (3.3) when (2.1) represents the QAP.

We refer to $\Phi(x) = x^T(e - x)$ in (3.3) as the *quadratic penalty function* and $\Phi(x) = \|x \circ (e - x)\|^2$ in (3.4) as the *quartic penalty function*. It is easy to see that the quadratic penalty function is concave and the quartic penalty function is neither concave nor convex. We note that almost all of the literature on continuous optimization approaches via relaxation and a penalty function has focused on studying the quadratic penalty function. The paper of Davydov and Sigal [28] seems to be the sole study employing the quartic penalty function in their continuous relaxation. They used a different penalty parameter for each component of the penalty term; i.e., their penalty term is $\|\mu \circ x \circ (e - x)\|^2$ with $\mu \in \mathbb{R}^n$ being a positive vector.

In the following sections, the results from both the literature and this research on several relevant topics on the methods of relaxation with a penalty function are presented and discussed. In section 3.2, the general conditions under which weak equivalence holds are established. Section 3.3 presents the weak equivalence properties in the context of relaxation using the quadratic penalty function. Two special cases of quadratic 0-1 programming problems in which μ_0 is known are also discussed. In section 3.4 strong equivalence is shown to hold for the QAP relaxation using the quadratic penalty function. We also show how to compute an estimate of μ_0 for which strong equivalence holds. Section 3.5 outlines a smoothing algorithm developed by Ng [87] and its drawbacks are discussed. We describe asymptotic properties of the quartic penalty function in section 3.6. Then the chapter is concluded with a summary of the methods of relaxation with a penalty function.

3.2 General Weak Equivalence

Giannessi and Niccolucci [41] studied the problem of minimizing a function $f(x)$ on a set, say $\bar{\mathcal{Z}}$, and the related problem of minimizing $f(x)$ with a penalty function on a larger set in which $\bar{\mathcal{Z}}$ is embedded. The latter problem is a relaxation of the former one. They established the conditions under which the two problems attain the same global minimum in terms of both the optimal objective value and the set of minimum points. Note that in their original study no other assumptions than compactness are made on $\bar{\mathcal{Z}}$. In fact $\bar{\mathcal{Z}}$ is not necessarily a discrete set. Since the focus of this section is on nonlinear 0-1 programming of which the QAP is a special case, we will adapt their results to the context of nonlinear 0-1 programming and state the conditions under which weak equivalence holds.

Setting $\mathcal{R} = \{x \in \mathbb{R}^n : g(x) \leq 0\}$ and $\mathcal{Z} = \mathbb{B}^n$, we rewrite (2.1) as

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in \mathcal{R} \cap \mathcal{Z} \end{aligned} \tag{3.5}$$

and (3.2) as

$$\begin{aligned} &\text{minimize} && f(x) + \mu\Phi(x) \\ &\text{subject to} && x \in \mathcal{R} \cap \mathcal{X} \end{aligned} \tag{3.6}$$

where $\mathcal{X}$ is define in (3.1).

Theorem 3.1 (Giannessi and Niccolucci [41], 1976, modified) *For problems (3.5) and (3.6), suppose that $\mathcal{R} \subseteq \mathbb{R}^n$ is closed, $\mathcal{X}$ is compact, $\mathcal{Z} \subset \mathcal{X} \subset \mathbb{R}^n$, and the following conditions hold:*

(i) *f is bounded below on $\mathcal{X}$ and there exists an open set $\mathcal{A} \supset \mathcal{Z}$ and real numbers $\alpha, L > 0$ such that, for each $x, y \in \mathcal{A}$, f satisfies the following Hölder condition:*

$$|f(x) - f(y)| \leq L\|x - y\|^{\alpha}.$$

(ii) *Φ is continuous on $\mathcal{X}$, and $\Phi(x) = 0$ for each $x \in \mathcal{Z}$ and $\Phi(x) > 0$ for each $x \in \mathcal{X} \setminus \mathcal{Z}$.*

(iii) *For each $z \in \mathcal{Z}$, there exists a neighborhood $S(z)$ of z and a real number $\epsilon(z) > 0$ such that*

$$\Phi(x) \geq \epsilon(z)\|x - z\|^{\alpha}, \quad \textit{for each } x \in S(z) \cap (\mathcal{X} \setminus \mathcal{Z}).$$

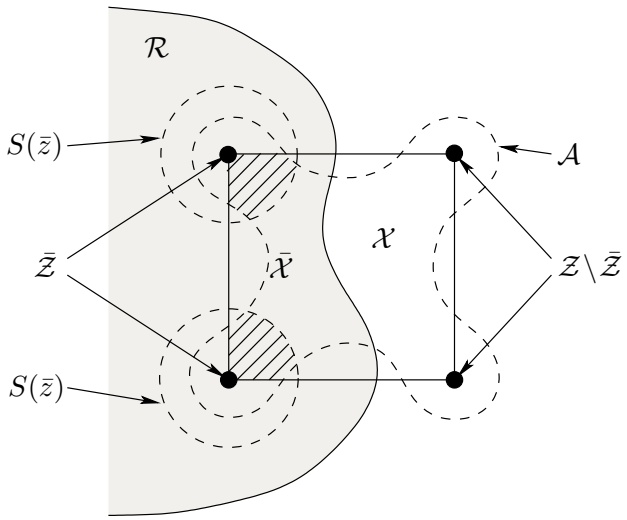


Figure 3.1 Proof of Weak Equivalence

Then there exists μ_0 such that for each $\mu > \mu_0$ problems (3.5) and (3.6) are weakly equivalent.

Proof. We only need to show that for each $\mu > \mu_0$ properties (A2) and (A3) are satisfied. Figure 3.1 illustrates the sets that will appear in this proof. We introduce the sets $\bar{\mathcal{X}} = \mathcal{R} \cap \mathcal{X}$, $\bar{\mathcal{Z}} = \mathcal{R} \cap \mathcal{Z}$, and $\bar{S}(z) = \mathcal{A} \cap S(z)$ for $z \in \mathcal{Z}$. It will be shown that for each $\bar{z} \in \bar{\mathcal{Z}}$, the function

$$F_{\bar{z}}(x) = \frac{f(\bar{z}) - f(x)}{\Phi(x)}$$

is bounded on $\bar{S}(\bar{z}) \cap (\bar{\mathcal{X}} \backslash \bar{\mathcal{Z}})$.

For all $x \in \bar{S}(\bar{z}) \cap (\bar{\mathcal{X}} \backslash \bar{\mathcal{Z}})$, we have

$$\Phi(x) \geq \epsilon(\bar{z}) \|x - \bar{z}\|^{\alpha} \quad \text{and}$$
$$|f(x) - f(\bar{z})| \leq L \|x - \bar{z}\|^{\alpha}.$$

Thus

$$|F_{\bar{z}}(x)| \leq \frac{L}{\epsilon(\bar{z})} \triangleq \beta(\bar{z}) < \infty.$$

Consequently we have

$$\frac{f(\bar{z}) - f(x)}{\Phi(x)} \leq \beta(\bar{z}).$$

Let $\mu > \beta \triangleq \max_{\bar{z}\in\bar{\mathcal{Z}}}\{\beta(\bar{z})\}$. For each $\bar{z} \in \bar{\mathcal{Z}}$, we have

$$f(x) + \mu\Phi(x) > f(x) + \beta(\bar{z})\Phi(x) \geq f(\bar{z}), \quad \text{for each } x \in \bar{S}(\bar{z}) \cap (\bar{\mathcal{X}} \setminus \bar{\mathcal{Z}}). \tag{3.7}$$

Therefore each $\bar{z} \in \bar{\mathcal{Z}}$ is a local minimum for $\mu > \beta$, which verifies property (A2).

Now we set

$$S = \left[\bigcup_{\bar{z}\in\bar{\mathcal{Z}}} \bar{S}(\bar{z}) \right] \cap \bar{\mathcal{X}} \supset \bar{\mathcal{Z}}$$

and

$$\mathcal{X}_0 = \bar{\mathcal{X}} \setminus S = \bar{\mathcal{X}} \cap \left[\mathbb{R}^n \setminus \bigcup_{\bar{z}\in\bar{\mathcal{Z}}} \bar{S}(\bar{z}) \right].$$

Hence $\mathcal{X}_0$ is compact. Furthermore, Φ is continuous and positive on $\mathcal{X}_0$. Thus

$$M_\Phi = \inf_{x\in\mathcal{X}_0} \Phi(x) = \min_{x\in\mathcal{X}_0} \Phi(x) > 0.$$

Since f is bounded below on $\bar{\mathcal{X}}$, we have

$$M_f = \inf_{x\in\bar{\mathcal{X}}} f(x) > -\infty.$$

Let

$$\gamma = \frac{f(\bar{z}^*) - M_f}{M_\Phi} \tag{3.8}$$

where $f(\bar{z}^*) = \min_{\bar{z}\in\bar{\mathcal{Z}}} f(\bar{z})$. Obviously $0 \leq \gamma < \infty$. For $\mu > \gamma$ and $x \in \mathcal{X}_0$,

$$f(x) + \mu\Phi(x) > M_f + \gamma M_\Phi = f(\bar{z}^*).$$

That is, the global minimum of (3.6) cannot occur on $\mathcal{X}_0$. By (3.7) the global minimum of (3.6) must be attained at some $\bar{z} \in \bar{\mathcal{Z}}$; i.e., property (A3) is satisfied. Setting $\mu_0 = \max\{\beta, \gamma\}$ completes the proof. ∎

Remark 3.1 In the original proof in Giannessi and Niccolucci [41], γ is defined as

$$\gamma = \frac{\sup_{x\in\bar{\mathcal{X}}} f(x) - M_f}{M_\Phi} < \infty,$$

which requires that f be also bounded above. They have

$$f(x) + \mu\Phi(x) > M_f + \gamma M_\Phi \geq f(\bar{z}), \quad \text{for each } x \in \mathcal{X}_0,$$

for each $\bar{z} \in \bar{\mathcal{Z}}$ and $\mu > \gamma$. This is unnecessary since in order to prove weak equivalence, we only need to show that the global minimum does not occur on $\mathcal{X}_0$. Hence, we defined γ as in (3.8) and relaxed the upper bound on f.

Remark 3.2 Alternatively we can define $\gamma = (f(\bar{z}) - M_f)/M_\Phi$ for some $\bar{z} \in \bar{\mathcal{Z}}$ with $f(\bar{z}) < \infty$. Thus for $\mu > \gamma$ and $x \in \mathcal{X}_0$,

$$f(x) + \mu\Phi(x) > M_f + \gamma M_\Phi = f(\bar{z}) \geq f(\bar{z}^*).$$

In theory such a definition of γ will generally result in a larger γ.

Remark 3.3 The compactness of $\mathcal{X}$ can be relaxed. In this case we define $M_\Phi = \inf_{x \in \mathcal{X}_0} \Phi(x) > 0$. Note that the infimum may not be attained on $\mathcal{X}_0$.

3.3 Relaxation Using the Quadratic Penalty Function

In this section we will present the results of Giannessi and Niccolucci [41] which show that not only problems (2.1) and (3.3) are weakly equivalent but also the relaxation using the quadratic penalty function (3.3) has a strictly concave objective function for $\mu > \mu_0$. We will also discuss two special cases of nonlinear 0-1 programming in which μ_0 can be efficiently computed.

Before we proceed, we will survey other relevant research on the subject in the literature. One of the earliest work was due to Raghavachari [97]. Raghavachari studied the linear 0-1 programming problem where $f(x)$ and $g(x)$ are linear functions in (2.1) and (3.3). It was shown that every 0-1 feasible solution to $\{x \in \mathbb{R}^n : Ax \leq b, 0 \leq x \leq e\}$ is an extreme point of the polyhedron. Using the well-known property (e.g., [62, 112]) that a strictly concave function on a compact convex set attains its minima at extreme points, Raghavachari established that (2.1) is equivalent to a concave minimization problem of the form (3.3) for a sufficiently large μ. Kalantari and Rosen [66] derived a lower bound for μ_0 such that for $\mu > \mu_0$ the equivalence in Raghavachari [97] holds. Furthermore, Kalantari and Rosen [67] and Borchardt [14] independently studied linearly constrained nonlinear 0-1 programming problems. They showed that the above equivalence result and lower bound for μ_0 obtained in linear 0-1 programming generalize to the cases with

a concave nonlinear objective function and linear constraints. They also described how to extend their results to 0-1 programming problems with a general nonlinear objective function and linear constraints. However, in both Kalantari and Rosen [67] and Borchardt [14], computing the lower bound for μ_0 requires solving a continuous global minimization problem of a concave objective function over a polytope, which itself is an intractable problem. Li [76] proposed a scheme of solving nonlinear 0-1 programming problems, where both $f(x)$ and $g(x)$ are nonlinear in (2.1), by solving a sequence of problems

$$\begin{aligned} \text{minimize} \quad & f(x) + \mu x^T(e - x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & x^T(e - x) \leq \tfrac{1}{\mu} \\ & 0 \leq x \leq e \end{aligned}$$

for an increasing penalty parameter μ. Li chose the initial μ such that $\mu x^T(e - x)$ is of the same order of magnitude as $f(x)$. The constraint $x^T(e - x) \leq \frac{1}{\mu}$ was meant to accelerate the convergence to a 0-1 point and overcome the difficulty of μ becoming too large. Li also noted that the proposed method heavily depends on the initial point and it may converge to a non-integer local minimum.

Now we state the main results on problem (2.1) and its relaxation using the quadratic penalty function. Giannessi and Niccolucci [41] proved that conditions (ii) and (iii) of Theorem 3.1 hold for the quadratic penalty function. Furthermore, they showed that under a mild condition the objective function $f(x) + \mu x^T(e - x)$ in (3.3) becomes strictly concave for a sufficiently large μ. They obtained the following theorem.

Theorem 3.2 (Giannessi and Niccolucci [41], 1976) *For problems (2.1) and (3.3), suppose that f satisfies condition (i) of Theorem 3.1 with $\alpha = 1$; i.e., f is bounded below on $\mathcal{X} = \{x \in \mathbb{R}^n : 0 \leq x \leq e\}$ and Lipschitz continuous on an open set $\mathcal{A} \supset \mathcal{Z} = \mathbb{B}^n$. Furthermore, $f \in \mathcal{C}^2(\mathcal{X})$. Then there exists μ_0 such that for each $\mu > \mu_0$, problems (2.1) and (3.3) are weakly equivalent and (3.3) has a strictly concave objective function.*

Now for a nonlinear 0-1 programming problem (2.1) that satisfies the conditions in Theorem 3.2, we can obtain a relaxation of form (3.3) whose objective function is strictly concave. If $\{x \in \mathbb{R}^n : g(x) \leq 0\}$ is a convex set, e.g., a polyhedron when $g(x)$ is linear,

we can exploit the extensive theory and methods for minimization of concave functions to globally solve (3.3) and hence find the optimal solution to (2.1). To this end, we need to determine the threshold value μ_0 in Theorem 3.2 *a priori*. In the following we will discuss two cases of quadratic 0-1 programming where μ_0 can be determined. We will describe how to compute μ_0 for the QAP as a special case of quadratic 0-1 programming in section 3.4.

In unconstrained quadratic 0-1 programming problems, we set $\mu_0 = \max\left\{\frac{1}{2}\bar{\lambda}, 0\right\}$, where $\bar{\lambda}$ is the largest eigenvalue of the Hessian of the quadratic objective function. In this case the Hessian is constant, so $\bar{\lambda}$ can be computed efficiently. For $\mu > \mu_0$, the objective function of the relaxation using the quadratic penalty function is strictly concave. Since the only constraints in the relaxation are $0 \leq x \leq e$, the minima of the relaxation coincide with the vertices of the unit hypercube. The global minimum occurs at one of the vertices. Note that unconstrained quadratic 0-1 programming exhibits the strong equivalence properties.

We now consider the linearly constrained quadratic 0-1 programming problems. In general, setting μ_0 to $\max\left\{\frac{1}{2}\bar{\lambda}, 0\right\}$ in (3.3) does not guarantee that the global minimum is attained at a vertex of the unit hypercube because there are other vertices that are non-integer. The global minimum may occur at a non-integer vertex. We illustrate this by the following example.

Example 3.1 *Consider the following quadratic 0-1 programming problem with a convex objective function*

$$\begin{aligned} & \text{minimize} && \left(x_1 - \tfrac{1}{2}\right)^2 + x_2^2 \\ & \text{subject to} && 1.1x_1 + x_2 = 1 \\ & && x_1, x_2 \in \mathbb{B}. \end{aligned} \tag{3.9}$$

We plot the contours of the objective function and the constraint in figure 3.2. The constraint $1.1x_1 + x_2 = 1$ *crosses line* $x_2 = 0$ *at* $(0.9091, 0)$. *The only feasible integer solution is* $(0, 1)$*; point* $(1, 0)$ *is not feasible. The minimum of the relaxed problem obtained by replacing* $x_1, x_2 \in \mathbb{B}$ *with* $0 \leq x_1, x_2 \leq 1$ *is at* $(0.724, 0.2036)$. *The only eigenvalue of the Hessian of the objective function of (3.9) is* 2. *It is easy to see that for* $\mu > \frac{1}{2}\bar{\lambda} = 1$, $\left(x_1 - \frac{1}{2}\right)^2 + x_2^2 + \mu\sum_{i=1}^{2}(x_i - x_i^2)$ *is strictly concave. We substitute* $1 - 1.1x_1$ *for* x_2 *and*

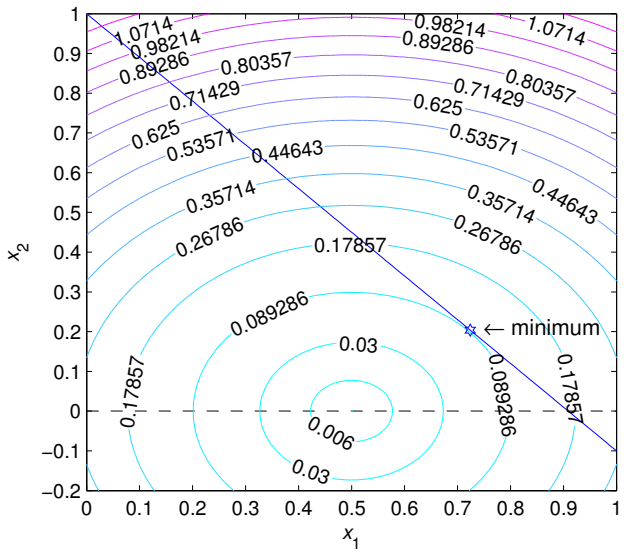


Figure 3.2 A Simple Problem with a Convex Quadratic Objective Function

obtain the following one-dimensional relaxation:

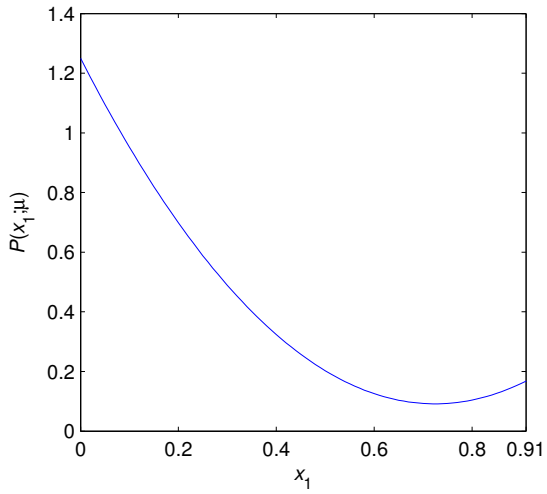
$$\begin{array}{lll} \text{minimize} & P(x_1;\mu) \triangleq 2.21(1-\mu)x_1^2 - (3.2-2.1\mu)x_1 + 1.25 \\ x_1 & \\ \text{subject to} & 0 \le x_1 \le 0.9091. \end{array} \tag{3.10}$$

Figure 3.3 illustrates the objective function of problem (3.10) parameterized by μ. As we can see, for $\mu = 3$, $P(x_1;\mu)$ is strictly concave. However, the global minimum occurs at $x_1 \approx 0.9091$, which is non-integer. We observe that $P(0;\mu) = P(0.9091;\mu) = 1.25$ for $\mu \approx 13.0919$ and the global minimum occurs at $x_1 = 0$ for $\mu > 13.0919$. Figure 3.3(d) shows that the global minimum is at $x_1 = 0$ for $\mu = 20$.

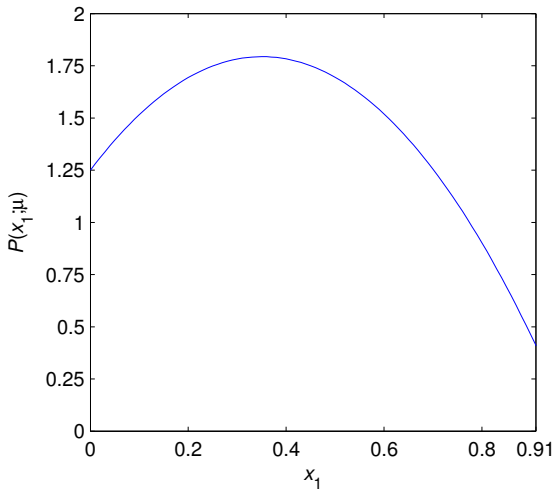
Zhu [119] studied the quadratic 0-1 programming problem of form (2.9). The relaxation using the quadratic penalty function is

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}x^TQx + c^Tx + \mu x^T(e-x) \\ \text{subject to} & Ax \le b \\ & 0 \le x \le e. \end{array} \tag{3.11}$$

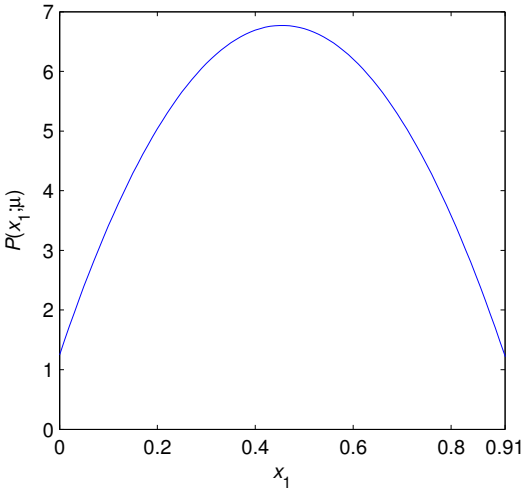
Zhu obtained the following result that gives an estimate of μ_0 in Theorem 3.2 for linearly constrained quadratic 0-1 programming problems.



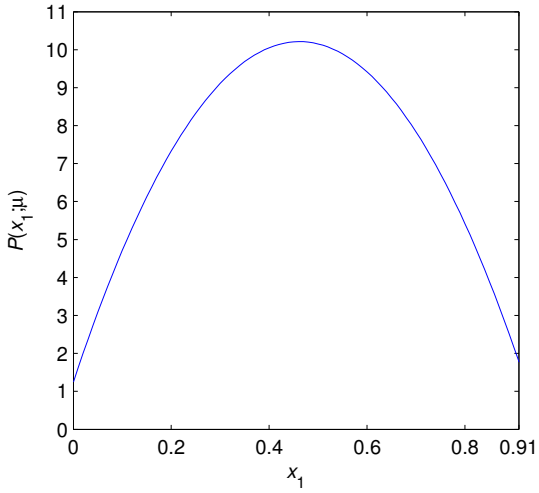
(a) $\mu = 0$



(b) $\mu = 3$



(c) $\mu \approx 13.0919$



(d) $\mu = 20$

Figure 3.3 Parameterized Objective Function in One-Dimensional Relaxation

Theorem 3.3 (Zhu [119], 2003) *Suppose that (2.9) is feasible and all the components of A and b are integral. Let*

$$\mu_0 = \tfrac{1}{2}\bar{\lambda} + 2(\bar{f} - \underline{f}) \max_{i=1,\ldots,m} \|A_i\|_\infty \tag{3.12}$$

where $\bar{\lambda}$ is the largest eigenvalue of Q, $\bar{f}$ is the upper bound on the objective value of (2.9), $\underline{f}$ is the lower bound on the objective value of (3.11) for $\mu = \frac{1}{2}\bar{\lambda}$, and A_i is the i-th row of matrix A. Then for each $\mu > \mu_0$, (2.9) and (3.11) are weakly equivalent and (3.11) is a quadratic programming problem with a strictly concave objective function.

Remark 3.4 The upper bound $\bar{f}$ is approximated by the sum of all the positive components of $\frac{1}{2}Q$ and c with $\bar{f} = 0$ if there is no positive component. To compute the lower bound $\underline{f}$, Zhu recommends a polynomial time algorithm of Pardalos and Rodgers [93].

We would like to point out that the estimate of μ_0 by (3.12) has two drawbacks. First, (3.12) may introduce an unnecessarily large penalty parameter μ for (3.11) because the lower bound and upper bound estimates $\underline{f}$ and $\bar{f}$ are not tight. Second, the value of μ_0 computed by (3.12) is sensitive to the scaling of the linear constraint coefficients. The following example illustrates the second drawback.

Example 3.2 *For the quadratic 0-1 programming problem (3.9), we multiply the coefficients of the linear constraint by* 100 *and omit the constant term in the objective function. Thus we have the following equivalent problem:*

$$\begin{aligned} &\text{minimize} && x_1^2 + x_2^2 - x_1 \\ &\text{subject to} && 110x_1 + 100x_2 = 100 \\ & && x_1, x_2 \in \mathbb{B} \end{aligned} \tag{3.13}$$

Obviously (3.13) satisfies the integrality condition on the components of A and b in Theorem 3.3. We have $\bar{\lambda} = 2$, $\bar{f} = 2$, and $\underline{f} = 0$. Thus $\mu_0 = \frac{1}{2}(2) + 2(2-0)(110) = 441$. In fact, we can scale the coefficients of the linear constraint to arbitrarily large integers, hence resulting in an arbitrarily large μ_0. Note that figure 3.3 shows that $\mu > 13.0919$ is sufficient for the problem in Example 3.1.

In summary, Theorem 3.2 establishes weak equivalence between a nonlinear 0-1 programming problem (2.1) and its relaxation using the quadratic penalty function (3.3).

Although the global minimum of (3.3) for $\mu > \mu_0$ is attained at a 0-1 solution, in general there are non-integer local minima to (3.3). For the cases where the constraints of (3.3) form a convex set and μ_0 in Theorem 3.2 can be determined, we can attempt to globally solve the concave minimization problem and find the optimal solution to the original 0-1 problem. However, the global concave minimization problem is generally NP-hard. Alternatively, we can locally solve the relaxation, but an algorithm that searches for a local minimum may end up at a non-integer local minimum of the relaxation and hence fail to find an integer feasible solution to the original 0-1 problem. In example 3.1, the global minimum is at $(x_1, x_2) = (0, 1)$ for $\mu > 13.0919$, but a local algorithm starting close to $(x_1, x_2) = (1, 0)$ will get trapped in the local minimum at $(x_1, x_2) \approx (0.9091, 0)$, as is obvious in figure 3.3(d).

In the next section we will discuss the relaxation using the quadratic penalty function for the QAP. The QAP possesses nice properties that preclude the above difficulty.

3.4 Strong Equivalence of the QAP Relaxation

Pardalos and Rosen [94] considered the QAP of the form:

$$\begin{aligned} &\text{minimize} && x^T Q x \\ &\text{subject to} && Lx = b \\ & && x \in \mathbb{B}^n \end{aligned} \tag{3.14}$$

and its relaxation:

$$\begin{aligned} &\text{minimize} && x^T (Q - \mu I) x \\ &\text{subject to} && Lx = b \\ & && x \geq 0, \end{aligned} \tag{3.15}$$

where μ is such that $Q - \mu I$ is negative definite. They showed that (3.14) and (3.15) are equivalent in the sense that the feasible solutions to (3.14) coincide with the minima of (3.15) and the objective values of (3.14) and (3.15) at each 0-1 feasible point differ by a constant.

In this research we consider the QAP of form (1.5) and its relaxation using the

quadratic penalty function:

$$
\begin{aligned}
(3.16) \qquad & \text{minimize} && \tfrac{1}{2}x^T Q x + c^T x + \mu x^T (e - x) \\
& \text{subject to} && Lx = b \\
& && x \geq 0.
\end{aligned}
$$

Since the assignment constraints $Lx = b$ and the nonnegativity constraints $x \geq 0$ combined imply upper bounds $x \leq e$, we do not explicitly write the upper bounds on x in (3.16). Note that the difference between (3.15) and (3.16) is in the linear terms in the objective function of (3.16). Before we present the main theorem, we need the following lemma.

Lemma 3.4 *The set of the feasible solutions to the QAP*

$$\bar{\mathcal{Z}} = \{x \in \mathbb{B}^n : Lx = b\}$$

coincide with the set of extreme points or vertices of the following polyhedral set

$$\bar{\mathcal{X}} = \{x \in \mathbb{R}^n : Lx = b, x \geq 0\}.$$

Proof. It is well-known (refer to Proposition A.1 in Appendix A.1) that the assignment matrix L in (1.5) is totally unimodular. Consequently, for every basic matrix B of L, $\det B = \pm 1$. If x_B are the basic variables, we have

$$Bx_B = b.$$

By Cramer's rule, the j-th component of x_B

$$(x_B)_j = \frac{\det B_j}{\det B}, \quad \forall j.$$

Where B_j is obtained from B by replacing its j-th column with b. If we expand the determinant of B_j by the cofactors of the j-th column, we have $\det B_j$ being integer valued since b is a vector of all ones. Hence $(x_B)_j$ is an integer; i.e., the basic solutions to $Lx = b$ are integer valued. It follows that the basic feasible solutions to $Lx = b, x \geq 0$ are also integer valued, and so are the extreme points of set $\bar{\mathcal{X}}$. Since $Lx = b, x \geq 0$ implies $x \leq e$, we have $x \in \mathbb{B}^n$ at the extreme points. ∎

Theorem 3.5 *The QAP (1.5) and its relaxation using the quadratic penalty function (3.16) are strongly equivalent for $\mu > \mu_0 = \frac{1}{2}\bar{\lambda}$, where $\bar{\lambda}$ is the largest eigenvalue of Q.*

Proof. For $\mu > \frac{1}{2}\bar{\lambda}$, the objective function of (3.16) is strictly concave. By Lemma 3.4, the 0-1 feasible solutions of (1.5) coincide with the extreme points of the feasible region of (3.16). Since a strictly concave function over a closed convex set attains its minima at the extreme points, by definition 3.2, (1.5) and (3.16) are strongly equivalent. ∎

We can select some $\mu > \mu_0 = \frac{1}{2}\bar{\lambda}$ so (1.5) and (3.16) are strongly equivalent. Alternatively, the following proposition provides an inexpensive method to compute an estimate of μ_0.

Proposition 3.6 (Bazaraa and Sherali [12], 1982) *Suppose $Q = (q_{ij}) \in \mathbb{R}^{n\times n}$ is symmetric and $q_{ij} \geq 0$ for $i, j = 1, \ldots, n$. Let $\mu_0 = \frac{1}{2}\|Q\|_\infty$, where $\|Q\|_\infty = \max\limits_{i=1,\ldots,n} \sum\limits_{j=1}^{n} |q_{ij}|$. Then $Q - 2\mu I$ is negative definite for $\mu > \mu_0$.*

Interested readers are referred to Appendix A.2 for the proof of Proposition 3.6.

By Theorem 3.5, we will always find a 0-1 feasible solution to the QAP (1.5) by locally solving its relaxation (3.16). That is, a 0-1 feasible solution is guaranteed for the QAP via the method of relaxation using the quadratic penalty function. However, a solution thus obtained depends on the starting point. It is likely that starting from an arbitrary point the method fails to find a good 0-1 solution to the QAP. In section 3.5, we will discuss an algorithm that attempts to address the issue of a suitable starting point.

3.5 A Smoothing Algorithm

In the method of relaxation using the quadratic penalty function for nonlinear 0-1 programming, our goal is to find a good, if not optimal, 0-1 solution to (2.1) by locally solving (3.3). If only weak equivalence holds, we require the algorithm that solves (3.3) to avoid non-integer local minima so we can find a 0-1 feasible solution to (2.1). In the case where strong equivalence holds, a 0-1 feasible solution to (2.1) is guaranteed. In either case, the local 0-1 solution obtained by solving (3.3) heavily depends on the starting point. Hence we need a scheme to start from a point that will lead to a good 0-1 solution.

To our knowledge, Ng [87] is the first of such work to address the issue of a suitable starting point for the method of relaxation using the quadratic penalty function. Ng developed a smoothing algorithm for the following nonlinear 0-1 programming problem

$$\begin{aligned} (3.17) \qquad & \text{minimize} && f(x) \\ & \text{subject to} && Ax = b \\ & && x \in \mathbb{B}^n. \end{aligned}$$

In the proposed algorithm the relaxation of (3.17) of the following form

$$(3.18) \qquad \begin{aligned} & \underset{x}{\text{minimize}} && P(x;\mu,\gamma) \triangleq f(x) + \mu x^T(e-x) - \gamma \sum_{i=1}^{n} \big(\log x_i + \log(1-x_i)\big) \\ & \text{subject to} && Ax = b \end{aligned}$$

is solved sequentially with an increasing μ and a decreasing γ. The initial μ is chosen to be very small and the initial γ to be sufficiently large. The barrier term $-\gamma \sum_{i=1}^{n} \big(\log x_i + \log(1-x_i)\big)$ serves two purposes. It replaces the bound constraints $0 \le x \le e$ in (3.3) and it has a smoothing effect; i.e., for a sufficiently large γ the multiple local minima have disappeared and $P(x;\mu,\gamma)$ becomes unimodal.

Ng [87] noted that with the initial μ and γ thus chosen, the algorithm effectively starts from the analytic center of the feasible region $\bar{\mathcal{X}} = \{\, x \in \mathbb{R}^n : Ax = b, 0 \le x \le e \,\}$, given as the optimal solution to the following problem:

$$(3.19) \qquad \begin{aligned} & \text{minimize} && -\sum_{i=1}^{n} \big(\log x_i + \log(1-x_i)\big) \\ & \text{subject to} && Ax = b. \end{aligned}$$

More specifically Ng proved the following theorem.

Theorem 3.7 (Ng [87], 2002) *Suppose $\bar{\mathcal{X}} \neq \varnothing$, $x^*(\gamma)$ is the optimal solutions to (3.18) for $\mu = 0$, and x^* is the optimal solution to (3.19). Then $\lim_{\gamma\to\infty} x^*(\gamma) = x^*$.*

The basic idea of the smoothing algorithm is that starting from a "neutral" point, i.e., the analytic center of the feasible region, and by gradually imposing integrality on the variables, the algorithm has a better chance to follow a continuous trajectory to the global or at least a good solution to (3.17). The following example shows how the smoothing algorithm works.

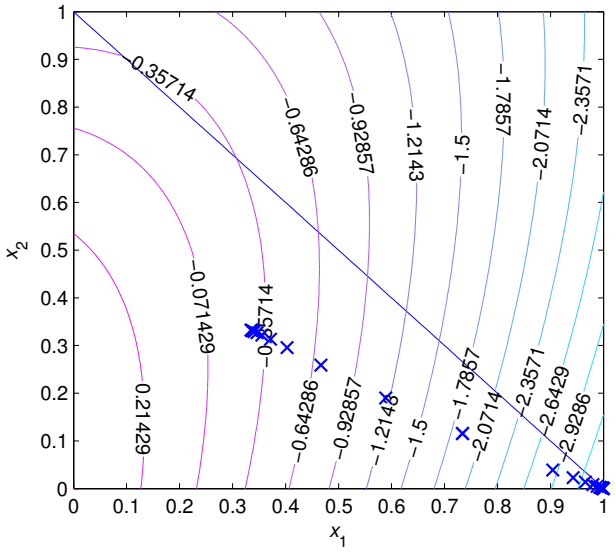


Figure 3.4 Illustration of Smoothing Algorithm in Two-Dimensional Subspace

Example 3.3 *Consider the following quadratic 0-1 programming problem*

$$
\begin{aligned}
(3.20) \qquad & \text{minimize} && f(x_1, x_2, x_3) = -3x_1^2 - 2\left(x_2 - \tfrac{1}{2}\right)^2 + x_3^2 \\
& \text{subject to} && x_1 + x_2 + x_3 = 1 \\
& && x_1, x_2, x_3 \in \mathbb{B}.
\end{aligned}
$$

There are three feasible solutions to (3.20) whose objective values are $f(1,0,0) = -3.5$, $f(0,1,0) = -0.5$, *and* $f(0,0,1) = 0.5$ *respectively. Obviously* $(x_1, x_2, x_3) = (1,0,0)$ *is the optimal solution to (3.20). For the illustration purpose, we replace* $x_1, x_2, x_3 \in \mathbb{B}$ *by* $0 \le x_1, x_2, x_3 \le 1$ *and then* x_3 *by* $1 - x_1 - x_2$. *Hence we obtain the following 2-dimensional relaxation (without adding a penalty term):*

$$
\begin{aligned}
(3.21) \qquad & \text{minimize} && f(x_1, x_2) = -2x_1^2 + 2x_1x_2 - x_2^2 - 2x_1 + \tfrac{1}{2} \\
& \text{subject to} && x_1 + x_2 \le 1 \\
& && 0 \le x_1, x_2 \le 1.
\end{aligned}
$$

We plot the contour of the objective function and the constraint of (3.21) in figure 3.4. Now we apply the smoothing algorithm to (3.20). We set the initial μ *to* 0.01 *and the initial* γ *to* 100. *In two successive iterations,* μ *is increased by a ratio of* 1.15 *and* γ *decreased by a ratio of* 0.6. *The algorithm starts at the analytic center of the feasible region,* $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. *The trajectory of the iterates are projected onto the* x_1-x_2 *plane as*

indicated by the symbols "×" in figure 3.4. Note that the smoothing algorithm does find the optimal solution to (3.20).

We would like to point out that the performance of the smoothing algorithm is very sensitive to the parameters chosen. These parameters include the initial values of μ and γ, and the ratios of their increase or decrease. Second, the choice of the ratio of increase for μ relative to the ratio of decrease for γ also affects the performance of the algorithm. Thirdly, a suitable set of parameters for the algorithm is often problem-dependent. These drawbacks considerably restrict the applicability of the smoothing algorithm.

3.6 Asymptotic Properties of the Quartic Penalty Function

In this section we will first show that for a nonlinear 0-1 programming problem its relaxation using the quartic penalty function does not exhibit weak equivalence for any finite penalty parameter μ. Then we will define asymptotic equivalence between a nonlinear 0-1 programming problem and its relaxation with a penalty function. Last we will prove that the QAP relaxation using the quartic penalty function possesses the weak asymptotic equivalence properties but not the strong asymptotic equivalence properties.

Proposition 3.8 *For a nonlinear 0-1 programming problem (2.1) with an objective function $f \in \mathcal{C}^2$, weak equivalence does not hold for its relaxation using the quartic penalty function.*

Proof. Let us consider the relaxation of the form (3.4). Setting $\mathcal{R} = \{x \in \mathbb{R}^n : g(x) \leq 0\}$ in (3.4), we have

$$\begin{aligned} (3.22) \qquad & \underset{x}{\text{minimize}} \quad f(x) + \mu \|x \circ (e - x)\|^2 \triangleq P(x; \mu) \\ & \text{subject to} \quad x \in \mathcal{R} \cap \mathcal{X} \end{aligned}$$

where $\mathcal{X}$ is defined in (3.1). If weak equivalence were to hold, it would have to be the case that for every instance of (2.1) every feasible solution is a local minimum of (3.22) for some sufficiently large μ. On the contrary, we will show that for some instances there exists a feasible solution which cannot be a local minimum of (3.22) for any finite μ.

Indeed, it is clearly the case that in some instances there will be a point $z \in \mathbb{B}^n \cap \mathcal{R}$ for which there exist $s \in \mathbb{R}^n$ and $0 < \bar{\alpha} < \infty$ such that

$$z + \alpha s \in \mathcal{R} \cap \mathcal{X} \text{ for } 0 < \alpha \leq \bar{\alpha}$$

and

$$\nabla f(z)^T s < 0.$$

By the Taylor's formula, we have

$$f(z + \beta s) = f(z) + \beta \nabla f(z)^T s + \tfrac{1}{2}\beta^2 s^T \nabla^2 f(z + \xi\beta s) s, \quad 0 < \xi < 1, \tag{3.23}$$

where $0 < \beta \leq \bar{\alpha}$. The following holds

$$\nabla f(z)^T s + \tfrac{1}{2}\beta s^T \nabla^2 f(z + \xi\beta s) s < \sigma \nabla f(z)^T s \tag{3.24}$$

for $0 < \sigma < 1$ if and only if

$$\tfrac{1}{2}\beta s^T \nabla^2 f(z + \xi\beta s) s < (\sigma - 1)\nabla f(z)^T s.$$

Note that $(\sigma - 1)\nabla f(z)^T s > 0$. Let

$$M = \sup_{0<\beta\leq\bar{\alpha}} \left\{ \tfrac{1}{2} s^T \nabla^2 f(z + \xi\beta s) s \right\},$$

and

$$\beta_0 = \begin{cases} \bar{\alpha} & \text{if } M \leq 0 \\ \min\left(\bar{\alpha}, (\sigma - 1)\nabla f(z)^T s / M\right) & \text{otherwise} \end{cases}.$$

Thus, (3.24) holds for $0 < \beta < \beta_0$. By (3.23), we have

$$f(z + \beta s) < f(z) + \sigma\beta \nabla f(z)^T s \tag{3.25}$$

for $0 < \beta < \beta_0$. Furthermore, we have

$$\|(z + \beta s) \circ (e - z - \beta s)\|^2 = \beta^2 \sum_{i=1}^{n} \left[(1 - 2z_i - \beta s_i) s_i\right]^2.$$

Let

$$\bar{M} = \sup_{0<\beta<\beta_0} \sum_{i=1}^{n} \left[(1 - 2z_i - \beta s_i) s_i\right]^2.$$

Note that $\bar{M} > 0$. Defining $\beta_1 = \min\left(\beta_0, -\sigma\nabla f(z)^T s/(\mu\bar{M})\right)$, we have that for $0 < \beta < \beta_1$,

$$\mu\beta^2 \sum_{i=1}^{n} \left[(1 - 2z_i - \beta s_i)s_i\right]^2 \leq \mu\beta^2\bar{M} < -\sigma\beta\nabla f(z)^T s$$

or

$$\mu\|(z + \beta s) \circ (e - z - \beta s)\|^2 < -\sigma\beta\nabla f(z)^T s. \tag{3.26}$$

Adding (3.25) and (3.26), we get

$$\begin{aligned} P(z + \beta s; \mu) &= f(z + \beta s) + \mu\|(z + \beta s) \circ (e - z - \beta s)\|^2 \\ &< f(z) = P(z; \mu) \end{aligned}$$

for $0 < \beta < \beta_1$. Thus z cannot be a (local) minimum of (3.22). ∎

Now we consider the following asymptotic properties between nonlinear 0-1 programming problem (2.1) and its relaxation with a penalty function (3.2):

(B1) $\Phi(x) = 0$ for $x \in \mathbb{B}^n$ and $\Phi(x) > 0$ for $x \in \mathcal{X}\backslash\mathbb{B}^n$.

(B2) Every 0-1 feasible solution to (2.1) is a limit point of some sequence of (local) minima $\hat{x}(\mu)$ of (3.2) as $\mu \to \infty$.

(B3) The sequence of the global minima $x^*(\mu)$ of (3.2) has the limit at a 0-1 point.

(B4) The limit point of any sequence of (local) minima $\hat{x}(\mu)$ of (3.2) as $\mu \to \infty$ is a 0-1 feasible solution to (2.1).

Similar to definitions (3.1) and (3.2), we define the following.

Definition 3.3 (*Weak Asymptotic Equivalence*)

A nonlinear 0-1 programming problem (2.1) and its relaxation (3.2) are said to be weakly asymptotically equivalent if properties (B1)–(B3) hold.

Definition 3.4 (*Strong Asymptotic Equivalence*)

A nonlinear 0-1 programming problem (2.1) and its relaxation (3.2) are said to be strongly asymptotically equivalent if properties (B1), (B2), and (B4) hold.

To show that only weak asymptotic equivalence holds for the QAP and its relaxation using the quartic penalty function, we consider the following relaxation of the QAP:

$$\begin{aligned} (3.27) \qquad & \underset{x}{\text{minimize}} \quad \tfrac{1}{2}x^TQx + c^Tx + \mu\|x \circ (e-x)\|^2 \\ & \text{subject to} \quad Lx = b, \end{aligned}$$

The above form is equivalent to setting $\mathcal{X} = \mathbb{R}^n$ in (3.4). We will first show that for each 0-1 feasible solution to the QAP, there is a sequence of local minima of (3.27) that converge to the 0-1 point as $\mu \to \infty$. We will also show that there exists a sequence of local minima of (3.27) that converge to a noninteger point as $\mu \to \infty$.

Before we prove the main theorem, we need the following three lemmas. Lemma 3.9 gives the bounds on the norms of two matrices.

Lemma 3.9 *Given $n \times n$ symmetric matrices Q and D with D being positive definite, we let $H = \frac{1}{2\mu}Q + D$ and $W = -(LH^{-1}L^T)^{-1}$. Furthermore, let $\mu_0 > 0$ be such that for $\mu \geq \mu_0$, H is positive definite. Then, for $\mu \geq \mu_0$, we have*

$$(3.28) \qquad \|H^{-1}\| < \tau_1 \quad \textit{and}$$

$$(3.29) \qquad \|W\| < \tau_2$$

where $\tau_1 < \infty$ and $\tau_2 < \infty$ are independent of μ.

Proof. Before we proceed, we need the following notation. Let $p_k(z)$ denote a polynomial of degree k in x, i.e.,

$$p_k(z) = a_0 + a_1 z + a_2 z^2 + \cdots + a_k z^k.$$

Note that $\lim_{z\to 0} p_k(z) = a_0$. Similarly we will also use the notation $q_k(z)$ and $r_k(z)$. Now we have, for $\mu \geq \mu_0$,

$$\begin{aligned} H^{-1} &= \left(\frac{1}{2\mu}Q + D\right)^{-1} \\ &= \frac{\text{adj}\left(\frac{1}{2\mu}Q + D\right)}{\det\left(\frac{1}{2\mu}Q + D\right)} \end{aligned}$$

where adj$(\cdot)$ denotes the adjoint of a matrix, i.e., $[\text{adj } A]_{ij} = A_{ji}$ and A_{ji} denotes the (j,i)-cofactor of A. Note that each component of adj $\left(\frac{1}{2\mu}Q + D\right)$ is a polynomial of degree

$n-1$ in $\dfrac{1}{\mu}$. Hence we have

$$H^{-1} = \frac{[p_{n-1}^{(ij)}\left(\frac{1}{\mu}\right)]}{p_n\left(\frac{1}{\mu}\right)} \tag{3.30}$$

where we use the superscript (ij) to denote the (i,j)-th component of the adjoint. Thus

$$\begin{aligned}\|H^{-1}\| &= \frac{1}{p_n\left(\frac{1}{\mu}\right)}\|[p_{n-1}^{(ij)}\left(\frac{1}{\mu}\right)]\| \\ &< \frac{1}{p_n\left(\frac{1}{\mu}\right)}\sum_{i=1}^{n} p_{n-1}^{(ii)}\left(\frac{1}{\mu}\right) \triangleq \zeta_1(\mu)\end{aligned}$$

where the inequality follows from the fact that the eigenvalues of H are all positive. Since $p_n\left(\frac{1}{\mu}\right) > 0$ for $\mu \geq \mu_0$, $\zeta_1(\mu)$ is continuous on $\Omega \triangleq \{\mu : \mu_0 \leq \mu < \infty\}$. Furthermore,

$$\lim_{\mu\to\infty} \zeta_1(\mu) = \frac{1}{a_0}\sum_{i=1}^{n} a_0^{(ii)}$$

where a_0 and $a_0^{(ii)}$ denote the constant terms in $p_n\left(\frac{1}{\mu}\right)$ and $p_{n-1}^{(ii)}\left(\frac{1}{\mu}\right)$, respectively. It is easy to see that $\zeta_1(\mu)$ is bounded on Ω. Let $\tau_1 = \sup_{\mu\in\Omega} \zeta_1(\mu)$ and thus

$$\|H^{-1}\| < \tau_1$$

which proves (3.28).

Now we rewrite W as follows

$$W = -(LH^{-1}L^T)^{-1} = -(LD^{-1}(I + \frac{1}{2\mu}QD^{-1})^{-1}L^T)^{-1}.$$

Note that $LH^{-1}L^T$ is invertible since L has a full row rank and so is $I + \frac{1}{2\mu}QD^{-1}$ since $\det(I + \frac{1}{2\mu}QD^{-1})\det D = \det H \neq 0$. Using a similar argument as in (3.30), $(I+\frac{1}{2\mu}QD^{-1})^{-1}$ is a rational function in $\frac{1}{\mu}$ whose numerator is a matrix of polynomials of degree $n-1$ and whose denominator is a polynomial of degree n. Note that premultiplying $(I + \frac{1}{2\mu}QD^{-1})^{-1}$ by LD^{-1} and postmultiplying it by L^T preserves the degree of the polynomials in the numerator and denominator. Hence we have

$$LD^{-1}(I + \frac{1}{2\mu}QD^{-1})^{-1}L^T = \frac{[q_{n-1}^{(ij)}\left(\frac{1}{\mu}\right)]}{q_n\left(\frac{1}{\mu}\right)}$$

and furthermore

$$(LD^{-1}(I+\frac{1}{\mu}QD^{-1})^{-1}L^T)^{-1} = q_n(\frac{1}{\mu})\frac{[r^{(ij)}_{(n-1)^2}\left(\frac{1}{\mu}\right)]}{r_{n(n-1)}\left(\frac{1}{\mu}\right)}.$$

Therefore,

$$\begin{aligned}\|W\| &= \frac{q_n(\frac{1}{\mu})}{r_{n(n-1)}\left(\frac{1}{\mu}\right)}\|[r^{(ij)}_{(n-1)^2}\left(\frac{1}{\mu}\right)]\| \\ &< \frac{q_n(\frac{1}{\mu})}{r_{n(n-1)}\left(\frac{1}{\mu}\right)}\sum_{i=1}^{n} r^{(ii)}_{(n-1)^2}\left(\frac{1}{\mu}\right) \triangleq \zeta_2(\mu).\end{aligned}$$

$\zeta_2(\mu)$ is bounded on Ω and let $\tau_2 = \sup_{\mu\in\Omega}\zeta_2(\mu)$. Thus we have

$$\|W\| < \tau_2$$

which completes the proof. ∎

We have the following lemma that defines the relationship between a feasible solution to the QAP and the local minima of (3.27). Note that in the proofs of Lemmas 3.10 and 3.11 that follow, we use λ and $\tilde{\lambda}$ to denote Lagrange multipliers.

Lemma 3.10 *Let $\hat{x}$ be a feasible solution to the QAP. The QAP relaxation using the quartic penalty function (3.27) has local minima $x^*(\mu)$ in some neighborhood of $\hat{x}$ for sufficiently large μ. Furthermore,*

$$\lim_{\mu\to\infty} x^*(\mu) = \hat{x}.$$

Proof. Consider the following first order optimality conditions for (3.27).

$$\begin{aligned}&Qx + c + 2\mu(X - 3X^2 + 2X^3)e + L^T\tilde{\lambda} = 0 \\ &Lx = b\end{aligned}$$

where X is an $n\times n$ diagonal matrix formed from x and $\tilde{\lambda}\in\mathbb{R}^m$ is the vector of Lagrange multipliers. Replacing $\tilde{\lambda}$ with $2\mu\lambda$ and then dividing the first system of equations through by 2μ, we get

$$F(x,\lambda) = \begin{bmatrix} \frac{1}{2\mu}(Qx+c) + (X - 3X^2 + 2X^3)e + L^T\lambda \\ Lx - b \end{bmatrix} = 0.$$

The Jacobian of $F(x,\lambda)$ is as follows.

$$J(x,\lambda) = \begin{bmatrix} \frac{1}{2\mu}Q + I - 6X + 6X^2 & L^T \\ L & 0 \end{bmatrix} = \begin{bmatrix} H(x) & L^T \\ L & 0 \end{bmatrix}.$$

where $H(x) = \frac{1}{2\mu}Q + I - 6X + 6X^2$. Note that $J(x,\lambda)$ is independent of λ. We let

$$x^0 = \hat{x},$$
$$\lambda^0 = 0.$$

Since $\hat{x}$ is a feasible solution to the QAP, we have

$$F(x^0,\lambda^0) = \begin{bmatrix} \frac{1}{2\mu}w \\ 0 \end{bmatrix},$$
$$\|F(x^0,\lambda^0)\| = \frac{1}{2\mu}\|w\|,$$

where $w = Q\hat{x} + c$.

Let $\mathcal{B}((x^0,\lambda^0),r)$ denote a neighborhood of (x^0,λ^0) with a radius $r > 0$. For $(x,\lambda^1),(y,\lambda^2) \in \mathcal{B}((x^0,\lambda^0),r)$,

$$J(x,\lambda^1) - J(y,\lambda^2) = \begin{bmatrix} 6(X^2 - Y^2 + Y - X) & 0 \\ 0 & 0 \end{bmatrix}.$$

$$\begin{aligned} \|J(x,\lambda^1) - J(y,\lambda^2)\| &= 6\|X^2 - Y^2 + Y - X\| \\ &\le 6\|X - Y\|\|X + Y - I\|. \end{aligned}$$

Note that for $i = 1,\ldots,n$,

$$\begin{aligned} &-r < x_i - \hat{x}_i < r \quad \text{and} \quad -r < y_i - \hat{x}_i < r, \\ &-(2r+1) \le 2\hat{x}_i - 2r - 1 < x_i + y_i - 1 < 2\hat{x}_i + 2r - 1 \le 2r + 1, \\ &|x_i + y_i - 1| < 2r + 1. \end{aligned}$$

Hence,

$$\|J(x,\lambda^1) - J(y,\lambda^2)\| < 6N(2r+1)\|x - y\|.$$

That is, $J(x,\lambda)$ is Lipschitz continuous on $\mathcal{B}((x^0,\lambda^0),r)$ with the Lipschitz constant $\gamma \triangleq 6N(2r+1)$. Let $\hat{X}$ be an $n \times n$ diagonal matrix formed from $\hat{x}$ and we have

$$H(x^0) = \frac{1}{2\mu}Q + I - 6\hat{X} + 6\hat{X}^2 = \frac{1}{2\mu}Q + I.$$

To simplify the notation, we denote $H(x^0)$ by H in the following. Note that

$$\begin{aligned}\det J(x^0,\lambda^0) &= \det \begin{bmatrix} I & 0 \\ -LH^{-1} & I \end{bmatrix} \begin{bmatrix} H & L^T \\ L & 0 \end{bmatrix} \begin{bmatrix} I & -H^{-1}L^T \\ 0 & I \end{bmatrix} \\ &= \det \begin{bmatrix} H & 0 \\ 0 & -LH^{-1}L^T \end{bmatrix} = \det H \det(-LH^{-1}L^T).\end{aligned}$$

Let $\mu_0 > 0$ be such that H is positive definite for $\mu \geq \mu_0$. Thus, $J(x^0,\lambda^0)$ is nonsingular for $\mu \geq \mu_0$. Furthermore, we have

$$J(x^0,\lambda^0)^{-1} = \begin{bmatrix} H^{-1}(I + L^TWLH^{-1}) & -H^{-1}L^TW \\ -WLH^{-1} & W \end{bmatrix}$$

where $W = -(LH^{-1}L^T)^{-1}$. Therefore, we have that for $\mu \geq \mu_0$,

$$\begin{aligned}\|J(x^0,\lambda^0)^{-1}\| &\leq \|H^{-1}(I + L^TWLH^{-1})\| + \|-H^{-1}L^TW\| + \|-WLH^{-1}\| + \|W\| \\ &\leq \|H^{-1}\| + \|H^{-1}\|^2\|L\|_F^2\|W\| + 2\|H^{-1}\|\|L\|_F\|W\| + \|W\| \\ &= \|H^{-1}\| + \left(\|H^{-1}\|\|L\|_F + 1\right)^2 \|W\| \\ &\leq \tau_1 + \left[\tau_1\sqrt{N(2N-1)} + 1\right]^2 \tau_2 \triangleq \beta,\end{aligned}$$

which follows from lemma 3.9 and the observation $\|L\|_F = \sqrt{N(2N-1)}$ with $\|\cdot\|_F$ denoting the Frobenius norm of a matrix. Also we have

$$\begin{aligned}\|J(x^0,\lambda^0)^{-1}F(x^0,\lambda^0)\| &\leq \|J(x^0,\lambda^0)^{-1}\|\|F(x^0,\lambda^0)\| \\ &\leq \frac{\beta}{2\mu}\|w\| \triangleq \eta.\end{aligned}$$

As a result,

$$\alpha = \beta\gamma\eta = \frac{1}{2\mu}6N(2r+1)\beta^2\|w\| \leq \frac{1}{2}$$

for a sufficiently large μ. Note that for $r \geq \frac{-1+\sqrt{1+\frac{4}{3N\beta}}}{4} > 0$,

$$r \geq \frac{1}{6N(2r+1)\beta} = \frac{1}{\beta\gamma} \geq \frac{1-\sqrt{1-2\alpha}}{\beta\gamma} \triangleq r_0.$$

Therefore, by the Kantorovich theorem [64, 89], the system $F(x,\lambda)$ has a unique solution in the closure of $\mathcal{B}((x^0,\lambda^0),r_0)$. Moreover, as $\mu \to \infty$, $\alpha \to 0$ and $r_0 \to 0$. Since for a sufficiently large μ, the Hessian $H(x)$ at such a solution is positive definite, the solution is a minimum. ∎

Now we present the result regarding the noninteger local minima of (3.27) in the following.

Lemma 3.11 *For the QAP with $N \geq 5$, its relaxation using the quartic penalty function (3.27) has a noninteger local minimum $x^*(\mu)$ in some neighborhood of $\tilde{x} = \frac{1}{N}e$ for a sufficiently large μ. Furthermore,*

$$\lim_{\mu \to \infty} x^*(\mu) = \tilde{x}.$$

Proof. The proof is very similar to that of lemma 3.10. We have

$$F(x, \lambda) = \begin{bmatrix} \frac{1}{2\mu}(Qx + c) + (X - 3X^2 + 2X^3)e + L^T\lambda \\ Lx - b \end{bmatrix}$$

and

$$J(x, \lambda) = \begin{bmatrix} H(x) & L^T \\ L & 0 \end{bmatrix}.$$

We define x^0 and λ^0 as follows:

$$x^0 = \tilde{x} = \frac{1}{N}e,$$

$$\lambda^0 = \begin{bmatrix} -\kappa_1(N)d \\ 0 \end{bmatrix} \in \mathbb{R}^m$$

where $\kappa_1(N) = \frac{1}{N} - \frac{3}{N^2} + \frac{2}{N^3}$ and $d \in \mathbb{R}^N$ is a vector of ones. From the structure of L, we have

$$L^T\lambda^0 = -\kappa_1(N)e,$$

$$Le = Nb.$$

Let $\tilde{X}$ be an $n \times n$ diagonal matrix formed from $\tilde{x}$. Then

$$\frac{1}{2\mu}(Qx^0 + c) + (\tilde{X} - 3\tilde{X}^2 + 2\tilde{X}^3)e + L^T\lambda^0 = \frac{1}{2\mu}\left(\frac{1}{N}Qe + c\right) = \frac{1}{2\mu}w$$

and

$$Lx^0 - b = \frac{1}{N}Le - b = 0$$

where $w = \frac{1}{N}Qe + c$. Therefore

$$F(x^0, \lambda^0) = \begin{bmatrix} \frac{1}{2\mu}w \\ 0 \end{bmatrix}$$

and

$$\|F(x^0, \lambda^0)\| = \frac{1}{2\mu}\|w\|.$$

At $x^0 = \frac{1}{N}e$, we have

$$H(x^0) = \frac{1}{2\mu}Q + I - 6\tilde{X} + 6\tilde{X}^2 = \frac{1}{2\mu}Q + \kappa_2(N)I$$

where $\kappa_2(N) = 1 - \frac{6}{N} + \frac{6}{N^2}$. For $N \geq 5$, $\kappa_2(N) > 0$ and hence $\kappa_2(N)I$ is positive definite. The rest of the proof follows as in Lemma 3.10. ∎

We have the following theorem.

Theorem 3.12 *For $N \geq 5$, the QAP (1.5) and its relaxation using the quartic penalty function (3.27) are weakly asymptotically equivalent but not strongly asymptotically equivalent.*

Proof. The proof follows from Definitions 3.3 and 3.4, and Lemmas 3.10 and 3.11. ∎

3.7 Summary

In this chapter, we first described the relaxation with a penalty function for a nonlinear 0-1 programming problem. Then we introduced two definitions: weak equivalence and strong equivalence. We have shown that under mild conditions weak equivalence holds for a nonlinear 0-1 programming problem and its relaxation using the quadratic penalty function and the relaxation has a strictly concave objective function. Furthermore, strong equivalence holds for the QAP and its relaxation using the quadratic penalty function. A smoothing algorithm that attempts to address the issue of the starting point for the relaxation using the quadratic penalty function was described and its drawbacks were discussed. We showed that weak equivalence does not hold for a nonlinear 0-1 programming problem and its relaxation using the quartic penalty function for any finite penalty parameter. Then we defined weak asymptotic equivalence and strong asymptotic equivalence and showed that only weak asymptotic equivalence holds for the QAP relaxation using the quartic penalty function.

In this research, we apply the method of relaxation with a penalty function to solve the QAP. By the strong equivalence properties of the QAP, we can solve its relaxation

using the quadratic penalty function for some finite penalty parameter and always find an integer solution to the QAP. On the other hand, the QAP relaxation using the quartic penalty function possesses weak asymptotic equivalence only and it has the following drawbacks:

- By (weak) asymptotic equivalence, for a solution to the relaxation using the quartic penalty function to be close to an integer solution of the QAP, the penalty parameter may have to be very large. This can cause numerical difficulties in solving the relaxation, especially in higher dimensions.

- In absence of strong asymptotic equivalence, the algorithm can terminate at a local solution that is not integer.

Accordingly, the quadratic penalty function is preferred and hence the method of relaxation using the quadratic penalty function will be used in this research. With the relaxation using the quadratic penalty function, we have a concave minimization problem for a sufficiently large penalty parameter. Unless we can globally solve this problem, the solution we obtain depends critically on the starting point. In Chapter 4, we will describe how a good starting point might be found. Assuming we have one, denoted by x^0, we have the following options:

(i) Start from x^0 and solve the QAP relaxation using the quadratic penalty function for a sufficiently large penalty parameter μ.

(ii) Start from x^0 and gradually increase the penalty parameter μ, solving a series of relaxations of the form (3.16). It is to be hoped that the solutions to the relaxations follow some path leading to a good integer solution.

For option (ii), we have noticed the following:

- As we gradually increase the penalty parameter μ, the path the algorithm follows may not be continuous. The iterates may jump when μ increases beyond the threshold value μ_0. This occurs when the Hessian of the objective function at the solution to the relaxation changes from a positive definite matrix to an indefinite one. Hence even if we start from a good initial point, we may still get a mediocre solution.

- By gradually increasing the penalty parameters, we will need to solve many optimization problems before an integer solution can be obtained. This can be perceivably very expensive.

Naturally, one would ask whether the extra computation involved in option (ii) pays off. Our numerical experience has indicated that the significant amount of extra computation expended in solving a series of optimization problems is not justified by any improvement over the solution obtained by solving a single relaxation problem. Hence, we will adopt option (i) in this research. In Chapter 4, we will describe our algorithm for the QAP which is based on option (i).

Chapter 4

Solving the QAP via Relaxation and Quadratic Cuts

4.1 Overview

In Chapter 3 we showed that strong equivalence is a desirable property when we use the continuous relaxation and penalty function techniques to solve a nonlinear 0-1 programming problem. The QAP relaxation with the quadratic penalty function possesses such a property. As we also mentioned in Section 3.7, we will solve a single relaxation problem of form (3.16) for a sufficiently large penalty parameter as described in Theorem 3.5.

By the strong equivalence, we are guaranteed to obtain an integer solution to the QAP by solving (3.16). Nevertheless, two fundamental issues need to be addressed:

1. If we start from an arbitrary point and solve (3.16), chances are that we will only get an "average" solution to the QAP. Thus we need to choose a starting point for (3.16) so we can on the average obtain a "good" solution to the QAP.

2. Since there is no guarantee that we will obtain the optimal solution to the QAP by solving (3.16), if possible, we would like to use a current solution to find a better one.

In the discussions that follow in this chapter, we will denote the objective function of the QAP by $q(x)$, i.e.,

$$q(x) = \tfrac{1}{2}x^T Q x + c^T x.$$

4.1.1 The Algorithmic Framework

In this section we will outline the framework of our proposed algorithm for the QAP. The framework as depicted in Figure 4.1 addresses the two issues brought up in the above.

In step (1) of the framework, we solve the following *quadratic programming relaxation*:

$$\text{(QPR)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & q(x) \\ \text{subject to} & Lx = b \\ & x \geq 0 \end{array}$$

to obtain a starting point x^0. In step (2), we start from x^0 and solve the QAP relaxation with the quadratic penalty function (3.16) for the penalty parameter μ to obtain an initial solution to the QAP. That is, we solve

$$\text{(RXP)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & q(x) + \mu x^T(e - x) \\ \text{subject to} & Lx = b \\ & x \geq 0. \end{array}$$

Since there is no guarantee that the resulting solution will be optimal to the QAP or even a "good" one, we would like to employ a method to find a better one. Let us denote the *incumbent solution*, or the current best solution, by $\hat{x}$. It is obvious that any solution to the QAP with a smaller objective value, if it exists, must satisfy

$$q(x) \leq q(\hat{x}) - \epsilon \tag{4.1}$$

for some small positive ϵ. This suggests that we add (4.1) to (RXP) and solve the resulting more restricted problem. We refer to (4.1) as a *quadratic cut*. More specifically, we set the incumbent solution $\hat{x}$ in step (3), and in step (4) starting from x^0 we solve the *relaxation with the penalty and quadratic cut*:

$$\text{(RPC)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & q(x) + \mu x^T(e - x) \\ \text{subject to} & x \in \Omega \end{array}$$

where Ω is defined as

$$\Omega = \left\{ x \in \mathbb{R}^n : Lx = b, q(x) \leq q(\hat{x}) - \epsilon, x \geq 0 \right\}. \tag{4.2}$$

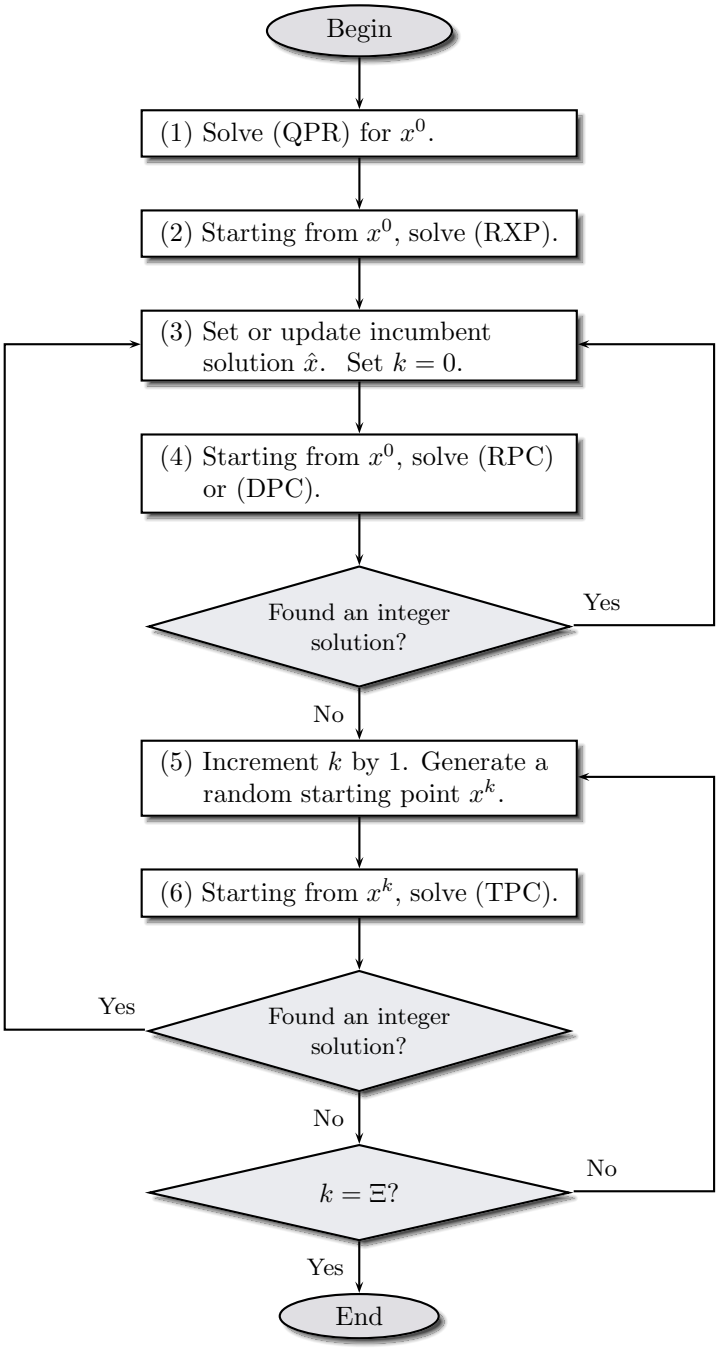


Figure 4.1 Flowchart of the Algorithmic Framework

Alternatively, we can replace the objective function of (RPC) with a slightly different one. That is, in step (4) we start from x^0 and solve the *minimal distance problem with the penalty and quadratic cut*:

$$\text{(DPC)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & P(x;\rho) \triangleq \|x - x^0\|^2 + \rho x^T(e - x) \\ \text{subject to} & x \in \Omega \end{array}$$

where ρ is chosen to be greater than one so that $\nabla_x^2 P(x;\rho)$ is negative definite. From the form of $P(x;\rho)$, $\nabla_x^2 P(x;\rho)$ is either negative definite or positive definite (except for $\rho = 1$). If $\nabla_x^2 P(x;\rho)$ were positive definite, (DPC) would have a unique optimum, which is unlikely to be an integer solution. We will explain why $\|x - x^0\|^2$ is used in the objective function in Section 4.4.

Since the optimal solution to either (RPC) or (DPC) satisfies (4.1), if it is integer it will be a better solution than $\hat{x}$. However, even though the objective functions of (RPC) and (DPC) are concave, the presence of the quadratic cut allows the possibility that an optimal solution to (RPC) or (DPC) may not be at an integer point. That is, strong equivalence does not hold for (RPC) or (DPC). If we obtain an integer solution, we go back to step (3), update $\hat{x}$ and solve (RPC) or (DPC) again. Otherwise, knowing that we are unable to obtain a better solution to the QAP by solving (RPC) or (DPC) from the given starting point, we try different points in Ω to see whether any of them would lead to an improved solution. To this end, we randomly generate a point x^k in Ω in step (5) and in step (6) we solve the *quartic penalty problem with the quadratic cut*:

$$\text{(TPC)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & \|x \circ (e - x)\|^2 \\ \text{subject to} & x \in \Omega \end{array}$$

starting from x^k. We will discuss why we use the quartic penalty function as the objective function in Section 4.4. If we find an integer solution to (TPC), we go back to step (3), update $\hat{x}$, and carry on from there. Otherwise, if we fail to find an integer solution to (TPC), we go back to step (5), generate a different random starting point, and re-solve (TPC) with the new starting point. We define the *number of stalled cuts* as the number of successive times we fail to find an integer solution to (TPC). We terminate when the number of stalled cuts reaches a prescribed number Ξ.

In the following discussions, we will collectively refer to (QPR), (RXP), (RPC), (DPC) and (TPC) as the *relaxation problems*, and (RPC), (DPC) and (TPC) as the

relaxation problems with the quadratic cut.

There are several techniques or components that are indispensable to our proposed algorithm for the QAP. We will describe the convex transformation of the objective function in Section 4.2. In Section 4.3 we will describe in detail the techniques of preconditioning that play a crucial role in our algorithm. We will discuss the considerations in the formulation of the relaxation problems with the quadratic cut in Section 4.4. In Section 4.5 we will show how to compute random starting points for (TPC).

4.2 Convex Transformation of the Objective Function

4.2.1 Motivation

In this section we will motivate the study of the techniques of convex transformations of the QAP that are essential for addressing the two issues described in Section 4.1.

To address the issue of a good starting point, we need to transform the general QAP with a nonconvex objective function to an equivalent problem with a convex objective function. To see why this is the case, note that in our proposed algorithm we start (RXP) from the optimal solution to (QPR). When the objective function is nonconvex, finding the global minimum is generally intractable; there are possibly multiple local minima and *a priori* information about which local minima lead to "good" solutions to the QAP is unavailable. An arbitrary local minimum x^0 does not necessarily lead to a "good" solution to the QAP. On the other hand, if we have a convex objective function (QPR) becomes a convex quadratic programming problem and we can efficiently compute its global minimum. We can then argue that the integer solutions closer to the global minimum tend to have better objective values than the integer solutions farther away from the global minimum.

Another reason to have a convex objective function lies in the use of the quadratic cut. To improve on the incumbent solution, we need to solve the relaxation problems with the quadratic cut. If $q(x)$ is nonconvex, the feasible regions of the relaxation problems with the quadratic cut are nonconvex sets, which make it difficult to solve the problems and even more difficult to find integer solutions to the problems. However, if $q(x)$ is convex, the feasible regions are convex sets and the relaxation problems with the quadratic

cut become more tractable.

4.2.2 Techniques of Convex Transformation

From the discussion in Section 4.2.1, it is clear that we need to have the objective function of the QAP to be convex. Fortunately we can transform a QAP instance with a nonconvex objective function to an equivalent problem with a convex objective function. There have been several forms of convex transformations suggested. White [116] described one convex transformation for the QAP. The quadratic term of the objective function of (1.4) is equivalent to the following:

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} f_{ij}d_{kl}\big[(x_{ik}+x_{jl})^2-(x_{ik}+x_{jl})\big], \tag{4.3}$$

which is obvious because $x_{ik}^2 = x_{ik}$ and $x_{jl}^2 = x_{jl}$ for $x_{ik}, x_{jl} \in \mathbb{B}$. For the QAP, we have $f_{ij} \geq 0$ and $d_{kl} \geq 0$. Thus, in general the function (4.3) is convex and White showed that if certain conditions on the flows and distances are satisfied, it becomes strictly convex.

In a different approach, Hammer and Rubin [57] replaced the objective function $q(x)$ with the following:

$$\begin{aligned} f(x;\gamma) &= \tfrac{1}{2}x^TQx + \tfrac{1}{2}\gamma x^T(x-e) + c^Tx \\ &= \tfrac{1}{2}x^T(Q+\gamma I)x + (c-\tfrac{1}{2}\gamma e)^Tx. \end{aligned} \tag{4.4}$$

Obviously $f(x;\gamma) = q(x)$ for $x \in \mathbb{B}^n$ and hence the transformation does not alter the solutions to the original 0-1 problem. Let $\underline{\lambda}$ denote the smallest eigenvalue of Q. Choosing $\gamma > -\underline{\lambda}$, we assure that $f(x;\gamma)$ is a strictly convex function.

It is easy to see that we do not have to choose the same parameter for all the terms in $x^T(x-e)$. Instead we can use the following slightly more general convex transformation

$$\begin{aligned} f(x;\phi) &= \tfrac{1}{2}x^TQx + \tfrac{1}{2}\sum_{i=1}^{n}\phi_i(x_i^2-x_i) + c^Tx \\ &= \tfrac{1}{2}x^T(Q+\mathrm{diag}(\phi))x + (c-\tfrac{1}{2}\phi)^Tx \end{aligned} \tag{4.5}$$

where $\phi = (\phi_i) \in \mathbb{R}^n$ is such that $Q+\mathrm{diag}(\phi)$ is positive definite. Hence, (4.4) is a special case of (4.5).

The convex transformation (4.5) enables us to obtain an equivalent problem with a strictly convex objective function through perturbing the diagonal elements of the

Hessian. As an additional advantage, (4.5) offers a simple and yet effective way to shift the eigenvalues of the Hessian of the resulting function, which is crucial for our pre-conditioning algorithm in Section 4.3. Transformation (4.3) does not produce a strictly convex function nor does it have the advantage of controlling the eigenvalues as with (4.5). Hence, we will not consider transformation (4.3) any further and we will adopt transformation (4.5) in our algorithm.

4.3 Pre-Conditioning the Hessian of the Objective Function

4.3.1 Motivation

The condition number of a positive definite matrix A (see e.g., [107]) is defined as

$$\text{cond}(A) = \frac{\overline{\lambda}(A)}{\underline{\lambda}(A)},$$

where $\underline{\lambda}(\cdot)$ and $\overline{\lambda}(\cdot)$ denote the smallest eigenvalue and largest eigenvalue of a matrix respectively. If $\text{cond}(A)$ is small (near 1), we say A is well-conditioned. The technique of improving the condition number of a matrix, i.e., bringing it closer to 1, is referred to as *pre-conditioning*.

Using the convex transformation (4.5), we can transform the general QAP to an equivalent problem with a strictly convex objective function. The condition number of the positive definite Hessian of the resulting objective function plays a crucial role in how our algorithm performs. The condition number manifests itself in two ways. First, for the relaxation problems described in Section 4.1, the condition number governs how numerically stable those problems are. With a well-conditioned positive definite Hessian the problems become easier to solve in the sense that we see a higher percentage of success in solving the problems to optimality and, if they are successfully solved, it takes fewer iterations for an optimization algorithm to converge.

Second, with a well-conditioned positive definite Hessian we have a greater chance of finding "good" solutions to the QAP. For a quadratic function whose Hessian is positive definite, its level sets are hyperellipsoids at a common center. The lengths of the axes of a hyperellipsoid are inversely proportional to the square root of the eigenvalues, $\lambda_1, \ldots, \lambda_n$, of the Hessian; i.e., the lengths are proportional to $1/\sqrt{\lambda_1}, \ldots, 1/\sqrt{\lambda_n}$. Hence,

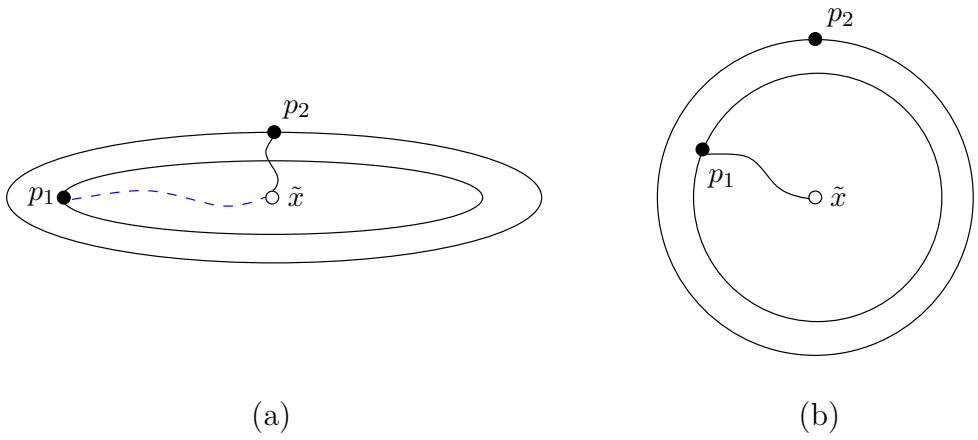


Figure 4.2 Contours of Convex Quadratic Functions

a hyperellipsoid with a well-conditioned Hessian has a shape closer to a hypersphere. To illustrate the implication of this, we consider the unconstrained examples in Figure 4.2. In Figure 4.2(a) and Figure 4.2(b), the level sets passing through two integer points are plotted for a convex quadratic function. $\tilde{x}$ denotes the unconstrained minimum of the convex quadratic function and p_1, p_2 denote two integer points. The objective value of p_1 is less than that of p_2. In general, with the method of relaxation using a penalty function, the algorithm tends to follow a path to the nearest integer point because of the "pull" induced by the penalty term in the objective function. In Figure 4.2(a) where the level sets are ellipses whose major and minor axes are significantly different in length, if we start from $\tilde{x}$, the algorithm tends to follow the solid line from $\tilde{x}$ to p_2 instead of the dashed line to p_1. On the other hand, in Figure 4.2(b) where the level sets are circles, starting from $\tilde{x}$, the algorithm tends to go to p_1 which has a better objective value than p_2. The objective function with a well-conditioned Hessian has contours that are more "circular" in shape and the nearest 0-1 integer point to the unconstrained minimum is more likely to be optimal. Thus, if we start from the unconstrained minimum, we have a better chance of finding a "good" integer solution. This argument can also be applied to the constrained cases.

For these reasons, we would like to have a scheme that performs the convex transformation and, at the same time, pre-conditions the Hessian of the objective function. To this end, we first scale the objective function by $\frac{1}{\sigma}, \sigma > 0$; i.e., the objective function becomes $\frac{1}{\sigma}q(x)$. Note this does not alter the solutions to the QAP but scales the

eigenvalues of Q. Then we add a positive multiple of identity matrix γI to the Hessian as in (4.4) to make it positive definite. Thus, we obtain an equivalent problem with the objective function:

$$\tfrac{1}{2}x^T\left(\frac{1}{\sigma}Q+\gamma I\right)x+\left(\frac{1}{\sigma}c-\tfrac{1}{2}\gamma e\right)^T x.$$

Let $\tilde{Q}=\frac{1}{\sigma}Q+\gamma I$. Suppose σ and γ are chosen such that $\tilde{Q}$ is positive definite and hence we have

$$\text{cond}(\tilde{Q})=\frac{\overline{\lambda}(Q)/\sigma+\gamma}{\underline{\lambda}(Q)/\sigma+\gamma}=1+\frac{(\overline{\lambda}(Q)-\underline{\lambda}(Q))/\sigma}{\underline{\lambda}(Q)/\sigma+\gamma},\quad \frac{\underline{\lambda}(Q)}{\sigma}+\gamma>0. \tag{4.6}$$

Note that for given γ and σ, $\text{cond}(\tilde{Q})$ depends on $(\overline{\lambda}(Q)-\underline{\lambda}(Q))$ as well as $\underline{\lambda}(Q)$. To make $\text{cond}(\tilde{Q})$ dependent on $(\overline{\lambda}(Q)-\underline{\lambda}(Q))$ but not on $\underline{\lambda}(Q)$ alone, we add $-\underline{\lambda}(Q)I$ to Q before we apply scaling; i.e., the objective function becomes:

$$\tfrac{1}{2}x^T\left(\frac{1}{\sigma}(Q-\underline{\lambda}(Q)I)+\gamma I\right)x+\left(\frac{1}{\sigma}(c+\tfrac{1}{2}\underline{\lambda}(Q)e)-\tfrac{1}{2}\gamma e\right)^T x.$$

Let $\tilde{Q}=\frac{1}{\sigma}(Q-\underline{\lambda}(Q)I)+\gamma I$. Choosing σ and γ so that $\tilde{Q}$ is positive definite we have

$$\text{cond}(\tilde{Q})=\frac{(\overline{\lambda}(Q)-\underline{\lambda}(Q))/\sigma+\gamma}{\gamma}=1+\frac{\overline{\lambda}(Q)-\underline{\lambda}(Q)}{\gamma\sigma},\quad \gamma>0. \tag{4.7}$$

Using (4.7) instead of (4.6) for the condition number of $\tilde{Q}$ enables an easier determination of γ and σ which, we will show in Section 4.3.3, can be done analytically. We define the *spread of the eigenvalues* of a matrix, denoted by $\text{sp}(\cdot)$, as the difference between its largest and smallest eigenvalues. We have $\text{sp}(Q)=\overline{\lambda}(Q)-\underline{\lambda}(Q)$. To bound the condition number of $\tilde{Q}$ by some given constant κ, (4.7) suggests the following two-step pre-conditioning scheme:

(i) minimize $\text{sp}(Q)$;
(ii) determine γ and σ in (4.7) such that $\tilde{Q}$ is positive definite and $\text{cond}(\tilde{Q})\leq\kappa$.

4.3.2 Minimizing the Spread of Eigenvalues

In this section we will describe two heuristics that attempt to minimize the spread of the eigenvalues of the Hessian of the objective function. The first algorithm is due to Carter [26], and the second one is based upon the work of Finke, Burkard and Rendl [33].

Carter [26] devised a variation of the modified Cholesky factorization of Gill and Murray [43] that simultaneously attempts to minimize the spread of the eigenvalues for

a general positive semidefinite matrix. His method proceeds as follows. Let Q be the Hessian of a general quadratic function and $\bar{Q}$ be a positive definite matrix derived from Q by modifying its diagonal elements by $\phi_j, j = 1, \ldots, n$. $\bar{Q}$ is factored into LL^T, i.e.,

$$\begin{pmatrix} \bar{q}_{11} & \bar{q}_{21} & \cdot & \cdot & \bar{q}_{n1} \\ \bar{q}_{21} & \bar{q}_{22} & \cdot & \cdot & \bar{q}_{n2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \bar{q}_{n1} & \bar{q}_{n2} & \cdot & \cdot & \bar{q}_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \\ l_{n1} & l_{n2} & \cdot & \cdot & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & \cdot & \cdot & l_{n1} \\ & l_{22} & \cdot & \cdot & l_{n2} \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \\ & & & & l_{nn} \end{pmatrix}.$$

For $j = 1, \ldots, n$,

$$\bar{q}_{jj} = q_{jj} + \phi_j = \sum_{r=1}^{j} l_{jr}^2 = \sum_{r=1}^{j-1} l_{jr}^2 + l_{jj}^2, \tag{4.8}$$

$$\bar{q}_{ij} = q_{ij} = \sum_{r=1}^{j} l_{ir} l_{jr} = \sum_{r=1}^{j-1} l_{ir} l_{jr} + l_{ij} l_{jj}, \quad i = j+1, \ldots, n.$$

Requiring the positive definiteness of $\bar{Q}$, we have the following of the column-wise Cholesky factorization:

$$l_{jj} = \sqrt{\bar{q}_{jj} - \sum_{r=1}^{j-1} l_{jr}^2} > 0$$

$$l_{ij} = \frac{1}{l_{jj}} \Big(q_{ij} - \sum_{r=1}^{j-1} l_{ir} l_{jr} \Big) = \frac{p_{ij}}{l_{jj}}, \quad i = j+1, \ldots, n, \tag{4.9}$$

where $p_{ij} \triangleq q_{ij} - \sum_{r=1}^{j-1} l_{ir} l_{jr}, i = j+1, \ldots, n$. In the modified Cholesky factorization, the key is to determine the l_{jj}. Once they have been determined, the l_{ij}, $i > j$, can be determined from (4.9). Then the ϕ_j can be recovered from (4.8).

Since the goal is to minimize the spread of the eigenvalues of $\bar{Q}$, we need to determine l_{jj} such that $\mathrm{sp}(\bar{Q})$ is minimized. Carter [26] used a simple approximation in which the sum of the eigenvalues is to be minimized. This is valid since the eigenvalues are all positive. The sum of the eigenvalues is equal to the trace of a matrix. Thus we are to minimize

$$\sum_{j=1}^{n} \bar{q}_{jj} = \sum_{j=1}^{n} \sum_{i=1}^{j} l_{ji}^2$$

which is the sum of the squares of all the components in L. Taking the sum by columns instead of by rows, we have

$$\sum_{j=1}^{n} \bar{q}_{jj} = \sum_{j=1}^{n} \sum_{i=j}^{n} l_{ij}^2 \tag{4.10}$$

A greedy heuristic is used to minimize the column sum in (4.10) at each iteration of the column-wise Cholesky factorization. For $j = 1, \ldots, n$, the jth column sum is

$$\sum_{i=j}^{n} l_{ij}^2 = l_{jj}^2 + \sum_{i=j+1}^{n} \frac{p_{ij}^2}{l_{jj}^2} \tag{4.11}$$

We let

$$\hat{l}_{jj} = \sqrt[4]{\sum_{i=j+1}^{n} p_{ij}^2}.$$

Thus at each iteration the value of l_{jj} that minimizes (4.11) is given by

$$l_{jj} = \begin{cases} \hat{l}_{jj} & \text{if } \hat{l}_{jj} > \epsilon; \\ \epsilon & \text{otherwise.} \end{cases}$$

where ϵ is a small positive number. For $j = n$, $\sum_{i=j}^{n} l_{ij}^2$ is minimized by $l_{nn} = 0$. Note that setting $l_{nn} = 0$ does not cause the factorization to fail because we do not need to evaluate (4.9) for $j = n$. However, it results in the smallest eigenvalue of $\bar{Q}$ being equal to zero.

Note that in the modified Cholesky factorization algorithm of Gill and Murray, l_{jj} is chosen such that the off-diagonal elements of L are uniformly bounded. Also the diagonal perturbation ϕ is optimal in the sense that an *a priori* bound on its norm is minimized. Hence the algorithm is numerically stable. In Carter's variant, l_{jj} is chosen to myopically minimize the column sum of L at each iteration. With that modification, Carter relaxes the bound on the off-diagonal elements of L. Furthermore, control of the norm of the diagonal perturbation is also lost. As a result, Carter's algorithm becomes numerically unstable. In Table 4.1 we compare Carter's algorithm with Gill and Murray's modified Cholesky factorization on a set of selected QAPLIB instances. Contrary to what is intended, Carter's algorithm drastically increases the spread of the eigenvalues instead of minimizing it on this set of problems. Hence we will not use Carter's algorithm.

Finke, Burkard, and Rendl [33] developed an algorithm to reduce the spread of the eigenvalues in the context of the QAP. Although their purpose was to obtain improved bounds for the QAP, the approach can be readily adapted for our purpose. We describe their approach as follows. Suppose we are given QAP(F, D, C) where $F, D, C \in \mathbb{R}^{N \times N}$. We assume F and D are symmetric. When either F or D, but not both, are asymmetric, we can symmetrize it as shown in Section 1.2.2. Let $g, h, r, s \in \mathbb{R}^N$ be vectors to be

Table 4.1 Comparison of Carter's and Gill and Murray's Algorithms

Problem	$\mathrm{sp}(\bar{Q})$	
	Carter	Gill & Murray
chr12b	8.026e+08	2.694e+06
esc16a	7.376e+06	4.621e+03
had14	5.686e+09	3.170e+04
had20	5.449e+14	7.782e+04
lipa20b	3.052e+14	3.019e+05
nug12	2.505e+08	1.433e+04
nug15	1.266e+11	2.725e+04
nug20	1.215e+14	5.664e+04
rou15	6.479e+11	4.456e+06
scr20	6.971e+12	1.875e+07
tai10a	9.521e+09	1.809e+06
tai17a	1.302e+14	5.345e+06

determined and define

$$G = ge^T + eg^T \quad \text{and}$$
$$H = he^T + eh^T.$$

Using the trace formulation of the QAP given in Section 1.2.2, we consider the following two reduction rules.

(Rd1) Let $\bar{F} = F - G$, $\bar{D} = D - H$, and $\bar{C} = 2\bar{F}eh^T + 2ge^TD + C$.
(Rd2) Let $\bar{F} = F - R$, $\bar{D} = D - S$, and $\bar{C} = \mathrm{diag}(\bar{F})s^T + r\,\mathrm{diag}(D)^T + C$.

In the above, $R = \mathrm{diag}(r)$, $S = \mathrm{diag}(s)$, $\mathrm{diag}(\bar{F})$, and $\mathrm{diag}(D)$ denote the vectors formed from the diagonals of matrices $\bar{F}$ and D, respectively. The following theorem shows that neither reduction rule (Rd1) nor (Rd2) will alter the solutions to the original QAP.

Theorem 4.1 (Finke, Burkard, and Rendl [33], 1987) *With reduction rule (Rd1) or (Rd2),* $\mathrm{tr}(FXD + C)X^T = \mathrm{tr}(\bar{F}X\bar{D} + \bar{C})X^T$ *for every* $X \in \mathcal{P}_N$*, where* $\mathcal{P}_N$ *is the set of all* $N \times N$ *permutation matrices.*

Proof. We first prove that the reduction results in an equivalent problem for every $X \in \Pi$

in (Rd1). Observing that $Xe = e$, $X^T e = e$, $e^T X = e^T$, and $e^T X^T = e^T$, we have

$$\begin{aligned}
\operatorname{tr}(FXD + C)X^T &= \operatorname{tr}\left[(\bar{F} + G)X(\bar{D} + H) + C\right]X^T \\
&= \operatorname{tr}(\bar{F}X\bar{D} + \bar{F}XH + GXD + C)X^T \\
&= \operatorname{tr}(\bar{F}X\bar{D} + \bar{F}Xhe^T + \bar{F}Xeh^T + ge^TXD + eg^TXD + C)X^T \\
&= \operatorname{tr}(\bar{F}X\bar{D} + \bar{F}Xhe^T + \bar{F}eh^T + ge^TD + eg^TXD + C)X^T. \qquad (4.12)
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
\operatorname{tr}\bar{F}Xhe^TX^T &= \operatorname{tr}\bar{F}Xhe^T = \operatorname{tr}Xhe^T\bar{F} = \operatorname{tr}\bar{F}^Teh^TX^T, \\
\operatorname{tr}eg^TXDX^T &= \operatorname{tr}XDX^Teg^T = \operatorname{tr}XDeg^T = \operatorname{tr}ge^TD^TX^T.
\end{aligned}$$

Thus (4.12) is equal to

$$\begin{aligned}
&\operatorname{tr}\left(\bar{F}X\bar{D} + (\bar{F} + \bar{F}^T)eh^T + ge^T(D + D^T) + C\right)X^T \\
&\qquad = \operatorname{tr}(\bar{F}X\bar{D} + 2\bar{F}eh^T + 2ge^TD + C)X^T.
\end{aligned}$$

Now for reduction rule (Rd2), we have

$$\begin{aligned}
\operatorname{tr}(FXD + C)X^T &= \operatorname{tr}\left[(\bar{F} + R)X(\bar{D} + S) + C\right]X^T \\
&= \operatorname{tr}(\bar{F}X\bar{D} + \bar{F}XS + RXD + C)X^T. \qquad (4.13)
\end{aligned}$$

Note the following identities

$$\begin{aligned}
\operatorname{tr}\bar{F}XSX^T &= \operatorname{tr}\operatorname{diag}(\bar{F})s^TX^T, \\
\operatorname{tr}RXDX^T &= \operatorname{tr}r\operatorname{diag}(D)^TX^T.
\end{aligned}$$

Thus (4.13) becomes

$$\operatorname{tr}(\bar{F}X\bar{D} + \operatorname{diag}(\bar{F})s^T + r\operatorname{diag}(D)^T + C)X^T$$

as required. ∎

Our goal is to minimize the spread of the eigenvalues of the Hessian Q of the QAP objective function. Since the eigenvalues of Q are all possible products of the eigenvalues of F and D as noted in Section 1.2.3, we tend to minimize $\operatorname{sp}(Q)$ if we minimize $\operatorname{sp}(F)$ and $\operatorname{sp}(D)$. Applying both reduction rules (Rd1) and (Rd2), we can

reduce F to $\bar{F} = F - G - R$ and D to $\bar{D} = D - H - S$. C is updated accordingly. Note $\bar{F}$ and $\bar{D}$ are also symmetric. Since there is no simple formula for $\mathrm{sp}(\bar{F})$ in terms of matrix elements, we instead minimize the upper bound for $\mathrm{sp}(\bar{F})$ given by Mirsky [86]:

$$\mathrm{sp}(\bar{F}) \leq m(\bar{F}) \triangleq \Big(2\sum_{i=1}^{N}\sum_{j=1}^{N} \bar{f}_{ij}^2 - \frac{2}{N}(\mathrm{tr}\,\bar{F})^2\Big)^{\frac{1}{2}}.$$

$m(\bar{F})$ is minimized at

$$g_k = \frac{1}{N-2}\Big(\sum_{j=1}^{N} f_{kj} - f_{kk} - y\Big), \quad k = 1, \ldots, N, \tag{4.14a}$$

$$r_k = f_{kk} - 2g_k, \quad k = 1, \ldots, N \tag{4.14b}$$

where $y = \frac{1}{2(N-1)}\Big(\sum_{i=1}^{N}\sum_{j=1}^{N} f_{ij} - \mathrm{tr}\,F\Big)$. Similarly by minimizing $m(\bar{D})$ we have

$$h_k = \frac{1}{N-2}\Big(\sum_{j=1}^{N} d_{kj} - d_{kk} - z\Big), \quad k = 1, \ldots, N, \tag{4.15a}$$

$$s_k = d_{kk} - 2h_k, \quad k = 1, \ldots, N. \tag{4.15b}$$

where $z = \frac{1}{2(N-1)}\Big(\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij} - \mathrm{tr}\,D\Big)$. It can be shown that the reduced matrices $\bar{F}$ and $\bar{D}$ have row and column sums equal to zero as well as zero diagonals. Hence the objective function of $\mathrm{QAP}(\bar{F}, \bar{D}, \bar{C})$ is not convex. Before we proceed, we need the following proposition.

Proposition 4.2 *Given symmetric matrices A and B, let $\bar{A} = A - \alpha I$ and $\bar{B} = B - \beta I$. Then, for $M = \bar{A} \otimes \bar{B}$, $\min\limits_{\alpha,\beta} \mathrm{sp}(M)$ is attained at*

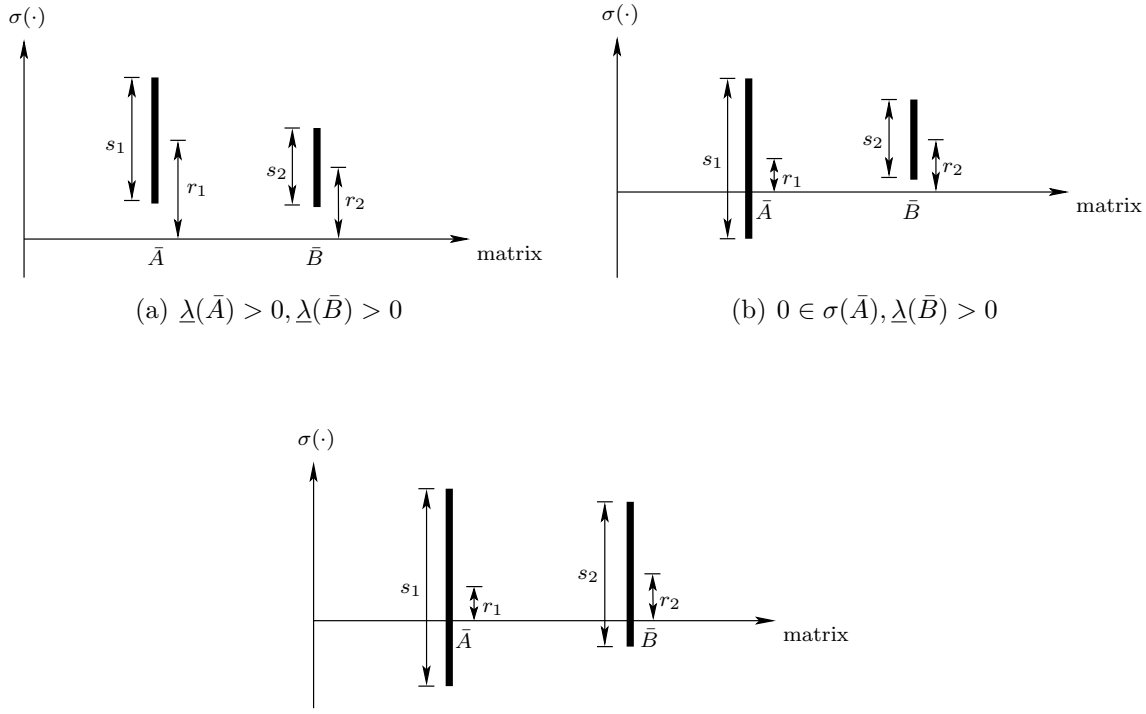
$$\alpha = \tfrac{1}{2}\left(\underline{\lambda}(A) + \overline{\lambda}(A)\right), \tag{4.16a}$$

$$\beta = \tfrac{1}{2}\left(\underline{\lambda}(B) + \overline{\lambda}(B)\right). \tag{4.16b}$$

Proof. Let $\sigma(\bar{A})$ and $\sigma(\bar{B})$ be defined as

$$\sigma(\bar{A}) = \{\lambda \in \mathbb{R} : \underline{\lambda}(\bar{A}) \leq \lambda \leq \overline{\lambda}(\bar{A})\},$$
$$\sigma(\bar{B}) = \{\lambda \in \mathbb{R} : \underline{\lambda}(\bar{B}) \leq \lambda \leq \overline{\lambda}(\bar{B})\}.$$

Varying α and β shift the eigenvalues of $\bar{A}$ and $\bar{B}$ without changing their spreads. Using an argument of symmetry in terms of $\sigma(\bar{A})$ and $\sigma(\bar{B})$ relative to 0, it suffices to consider the three cases depicted in Figure 4.3: a) $\underline{\lambda}(\bar{A}) > 0$ and $\underline{\lambda}(\bar{B}) > 0$; b) $0 \in \sigma(\bar{A})$ and



(a) $\underline{\lambda}(\bar{A}) > 0, \underline{\lambda}(\bar{B}) > 0$

(b) $0 \in \sigma(\bar{A}), \underline{\lambda}(\bar{B}) > 0$

(c) $0 \in \sigma(\bar{A}), 0 \in \sigma(\bar{B})$

Figure 4.3 Spreads of Eigenvalues of $\bar{A}$ and $\bar{B}$

$\underline{\lambda}(\bar{B}) > 0$; and c) $0 \in \sigma(\bar{A})$ and $0 \in \sigma(\bar{B})$. In Figure 4.3, $s_1 = \mathrm{sp}(A)$, $s_2 = \mathrm{sp}(B)$, and r_1 and r_2 are the vertical coordinates of the mid-points of $\sigma(\bar{A})$ and $\sigma(\bar{B})$ respectively. By varying r_1 and r_2, we change the positions of $\sigma(\bar{A})$ and $\sigma(\bar{B})$ relative to 0.

In the case shown in Figure 4.3(a), we have $r_1 > \frac{s_1}{2}$ and $r_2 > \frac{s_2}{2}$, and

$$\begin{aligned}
\overline{\lambda}(M) &= (r_1 + \frac{s_1}{2})(r_2 + \frac{s_2}{2}), \\
\underline{\lambda}(M) &= (r_1 - \frac{s_1}{2})(r_2 - \frac{s_2}{2}), \\
\mathrm{sp}(M) &= r_1 s_2 + s_1 r_2, \\
\inf_{r_1, r_2} \mathrm{sp}(M) &= s_1 s_2.
\end{aligned}$$

In the case shown in Figure 4.3(b), we have $-\frac{s_1}{2} \leq r_1 \leq \frac{s_1}{2}$ and $r_2 > \frac{s_2}{2}$, and

$$\begin{aligned}
\overline{\lambda}(M) &= (r_1 + \frac{s_1}{2})(r_2 + \frac{s_2}{2}), \\
\underline{\lambda}(M) &= (r_1 - \frac{s_1}{2})(r_2 + \frac{s_2}{2}), \\
\mathrm{sp}(M) &= \tfrac{1}{2} s_1 s_2 + s_1 r_2, \\
\inf_{r_1, r_2} \mathrm{sp}(M) &= s_1 s_2.
\end{aligned}$$

In the case shown in Figure 4.3(c), we have $-\frac{s_1}{2} \leq r_1 \leq \frac{s_1}{2}$ and $-\frac{s_2}{2} \leq r_2 \leq \frac{s_2}{2}$, and

$$\begin{aligned}
\overline{\lambda}(M) &= \max\left((r_1 + \frac{s_1}{2})(r_2 + \frac{s_2}{2}), (r_1 - \frac{s_1}{2})(r_2 - \frac{s_2}{2})\right), \\
\underline{\lambda}(M) &= \min\left((r_1 + \frac{s_1}{2})(r_2 - \frac{s_2}{2}), (r_1 - \frac{s_1}{2})(r_2 + \frac{s_2}{2})\right), \\
\mathrm{sp}(M) &= \tfrac{1}{2} s_1 s_2 + \delta
\end{aligned}$$

where $\delta = \frac{1}{2}\max(r_1 s_2 + s_1 r_2, -r_1 s_2 - s_1 r_2) + \frac{1}{2}\max(r_1 s_2 - s_1 r_2, s_1 r_2 - r_1 s_2)$. Note that

$$\delta = \begin{cases}
\max(r_1 s_2, s_1 r_2) \geq 0 & \text{if } r_1 s_2 \geq 0 \text{ and } s_1 r_2 \geq 0 \\
\max(-r_1 s_2, -s_1 r_2) \geq 0 & \text{if } r_1 s_2 \leq 0 \text{ and } s_1 r_2 \leq 0 \\
\max(-r_1 s_2, s_1 r_2) > 0 & \text{if } r_1 s_2 < 0 \text{ and } s_1 r_2 > 0 \\
\max(r_1 s_2, -s_1 r_2) > 0 & \text{if } r_1 s_2 > 0 \text{ and } s_1 r_2 < 0
\end{cases}.$$

Hence,

$$\min_{r_1, r_2} \mathrm{sp}(M) = \tfrac{1}{2} s_1 s_2.$$

Therefore, for all r_1 and r_2, $\min_{\alpha,\beta} \mathrm{sp}(M)$ is attained at $r_1 = r_2 = 0$, which is equivalent to (4.16a) and (4.16b). ∎

By Proposition 4.2, we can apply reduction rule (Rd2) for $R = \alpha I$ and $S = \beta I$ to $\bar{F}$, $\bar{D}$ and $\bar{C}$ before we obtain the Hessian of the objective function $\bar{Q} = 2\bar{D} \otimes \bar{F}$. We summarize the above results in Algorithm 4.1.

We have run the Matrix Reduction algorithm on the QAPLIB instances with the number of locations $N \leq 100$. The results are shown in Appendix B.2. Roughly speaking, the Matrix Reduction algorithm on the average reduces the spread of eigenvalues of the original Hessian by an order of magnitude for the set of instances we have run. However, the algorithm causes fill-in and produces almost completely dense Hessian matrices for the set of instances except lipa20a–lipa90a. For lipa20a–lipa90a, the algorithm in fact produces sparser matrices. When the Hessian is dense and we use the exact Hessian in an optimization algorithm to solve the relaxation problems, we will experience substantial slow-down in the optimization, especially as the problem size increases. We will discuss how to overcome this difficulty in Section 5.1.2.

4.3.3 Bounding the Condition Number of the Hessian

In this section we will describe an algorithm to bound the condition number of the Hessian. We can apply the algorithm to the Hessian of the original objective function of the QAP, or to the modified Hessian as the output of Algorithm 4.1. In either case the Hessian is generally nonconvex. Hence we need to make the Hessian positive definite and bound the condition number of the resulting Hessian $\tilde{Q}$ by some constant $\kappa > 1$. From the discussions in Section 4.3.1 and (4.7) in particular, we have

$$\begin{aligned} \mathrm{cond}(\tilde{Q}) = 1 + \frac{\overline{\lambda}(Q) - \underline{\lambda}(Q)}{\gamma\sigma} \leq \kappa \quad \text{or} \\ \gamma\sigma \geq \frac{\overline{\lambda}(Q) - \underline{\lambda}(Q)}{\kappa - 1} \triangleq \eta, \quad \gamma \geq \gamma_{\min} \text{ and } \sigma_{\min} \leq \sigma \leq \sigma_{\max} \end{aligned}$$

where $\gamma_{\min} > 0$ and $\sigma_{\max} > \sigma_{\min} > 0$ are given. $\gamma_{\min}$ is chosen such that $\tilde{Q}$ is sufficiently positive definite; $\sigma_{\min}$ is typically 1 because we do not want to increase the values by scaling.

We would like both γ and σ to be as small as possible. Thus we choose γ and σ to

Algorithm 4.1 (*Matrix Reduction*).

Input: F, D and $C \in \mathbb{R}^{N\times N}$ where F and D are symmetric matrices.

Output: a symmetric matrix $Q \in \mathbb{R}^{N^2\times N^2}$ with reduced spread of eigenvalues and $c \in \mathbb{R}^{N^2}$.

Compute vectors g, h, r, s and matrices G, H, R, S:

Compute g and r by (4.14a)–(4.14b);

Let $G = ge^T + eg^T$ and $R = \operatorname{diag}(r)$;

Compute h and s by (4.15a)–(4.15b);

Let $H = he^T + eh^T$ and $S = \operatorname{diag}(s)$;

Apply reduction rule (Rd1):

$F \leftarrow F - G$;

$C \leftarrow C + 2Feh^T + 2ge^T D$;

$D \leftarrow D - H$;

Apply reduction rule (Rd2):

$F \leftarrow F - R$;

$C \leftarrow C + \operatorname{diag}(F)s^T + r\operatorname{diag}(D)^T$;

$D \leftarrow D - S$;

Let $\alpha = \frac{1}{2}\left(\underline{\lambda}(F) + \overline{\lambda}(F)\right)$ and $\beta = \frac{1}{2}\left(\underline{\lambda}(D) + \overline{\lambda}(D)\right)$;

$F \leftarrow F - \alpha I$;

$C \leftarrow C + \beta\operatorname{diag}(F)e^T + \alpha e\operatorname{diag}(D)^T$;

$D \leftarrow D - \beta I$;

Compute Q and c:

Let $Q = 2D \otimes F$ and $c = \operatorname{vec}(C)$;

be the optimal solution to the following minimization problem.

$$
\begin{aligned}
(4.17) \qquad & \underset{\gamma,\sigma}{\text{minimize}} && \zeta\gamma + \sigma \\
& \text{subject to} && \gamma\sigma \geq \eta \\
& && \gamma \geq \gamma_{\min} \\
& && \sigma_{\min} \leq \sigma \leq \sigma_{\max}
\end{aligned}
$$

where $\zeta > 0$ is a pre-determined weight for γ relative to σ. We will discuss how to choose ζ later in this section. Note that (4.17) can be solved analytically. In particular, we have three cases to consider: 1) in Figure 4.4(a), $\gamma_{\min}\sigma_{\min} < \eta$ and $\frac{\eta}{\gamma_{\min}} < \sigma_{\max}$; 2) in Figure 4.4(b), $\gamma_{\min}\sigma_{\min} < \eta$ and $\frac{\eta}{\gamma_{\min}} \geq \sigma_{\max}$; and 3) in Figure 4.4(c), $\gamma_{\min}\sigma_{\min} \geq \eta$. In Figure 4.4, the shaded areas are the feasible region of (4.17). We depict the objective function values as parallel isolines. For $t_1 > t_2$, moving in the direction from $\zeta\gamma + \sigma = t_1$ to $\zeta\gamma + \sigma = t_2$ decreases the objective value. Since $\sigma = \frac{\eta}{\gamma}$ is a strictly convex function in γ for $\gamma > 0$, the minimum is attained where one of the isolines touches the feasible region at a single point. In cases 1) and 2), we let $\bar{\gamma}_{\min} = \max(\gamma_{\min}, \frac{\eta}{\sigma_{\max}})$ and $\bar{\gamma} = \sqrt{\frac{\eta}{\zeta}}$. It is easy to verify that the solution to (4.17) is

$$
\gamma^* = \begin{cases} \bar{\gamma}_{\min} & \text{if } \bar{\gamma} \leq \bar{\gamma}_{\min} \\ \bar{\gamma} & \text{if } \bar{\gamma}_{\min} < \bar{\gamma} \leq \frac{\eta}{\sigma_{\min}} \\ \dfrac{\eta}{\sigma_{\min}} & \text{otherwise} \end{cases},
$$

$$
\sigma^* = \frac{\eta}{\gamma^*}.
$$

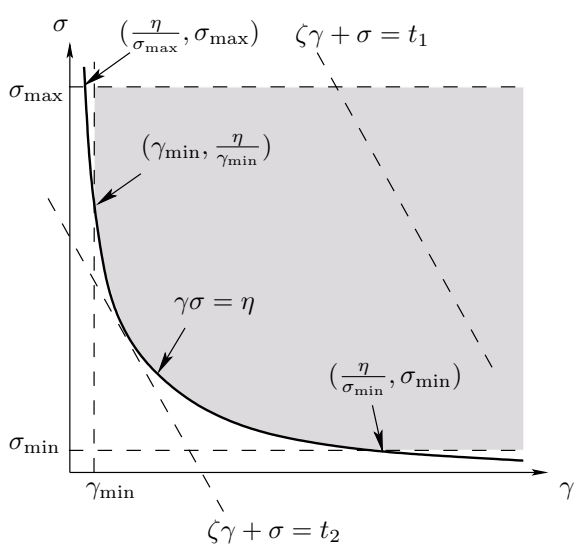
In case 3), the solution to (4.17) is

$$
\gamma^* = \gamma_{\min} \text{ and } \sigma^* = \sigma_{\min}.
$$

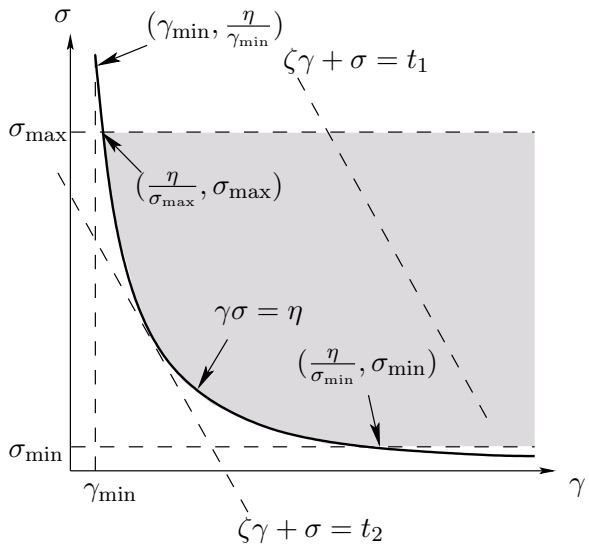
We summarize the above results in Algorithm 4.2.

Now we return to the issue of how to choose the weight ζ in (4.17). Hammer and Rubin [57] have the following result regarding the parameter γ in (4.4) in their study of the convex transformation of quadratic 0-1 programming problems.

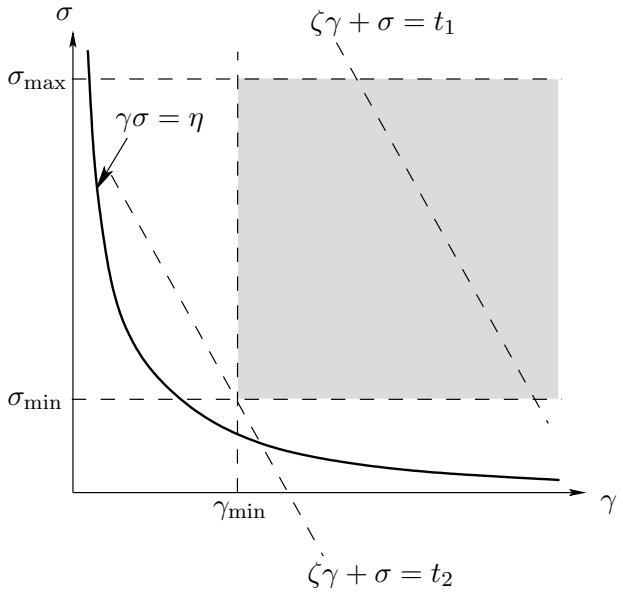
Theorem 4.3 (Hammer and Rubin [57], 1970) *For $i = 1, 2$, let γ_i be such that $f(x; \gamma_i)$ in (4.4) is strictly convex, and $\tilde{x}^i$ denote the optimal solution to a convex minimization problem whose objective function is $f(x; \gamma_i)$. Furthermore, let $p = \frac{1}{2}e$ denote*



(a) $\gamma_{\min}\sigma_{\min} < \eta$, $\frac{\eta}{\gamma_{\min}} < \sigma_{\max}$



(b) $\gamma_{\min}\sigma_{\min} < \eta$, $\frac{\eta}{\gamma_{\min}} \geq \sigma_{\max}$



(c) $\gamma_{\min}\sigma_{\min} \geq \eta$

Figure 4.4 Minimization for γ and σ

Algorithm 4.2 (*Scaling and Shift*).

Input: a symmetric matrix $Q \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$, scalars $\gamma_{\min} > 0$, $\sigma_{\max} > \sigma_{\min} > 0$, $\kappa > 1$, and $\zeta > 0$.

Output: a positive definite matrix $Q \in \mathbb{R}^{n\times n}$ whose condition number is bounded by κ, and $c \in \mathbb{R}^n$.

Let

$$\eta = \frac{\overline{\lambda}(Q) - \underline{\lambda}(Q)}{\kappa - 1};$$

if $\gamma_{\min}\sigma_{\min} < \eta$ **then**

  Let $\bar{\gamma}_{\min} = \max(\gamma_{\min}, \frac{\eta}{\sigma_{\max}})$ and $\bar{\gamma} = \sqrt{\frac{\eta}{\zeta}}$;

  if $\bar{\gamma} \le \bar{\gamma}_{\min}$ **then**

    Let $\gamma = \bar{\gamma}_{\min}$;

  else if $\bar{\gamma}_{\min} < \bar{\gamma} \le \frac{\eta}{\sigma_{\min}}$ **then**

    Let $\gamma = \bar{\gamma}$;

  else

    Let $\gamma = \frac{\eta}{\sigma_{\min}}$;

  end (if)

  Let $\sigma = \frac{\eta}{\gamma}$;

else

  Let $\gamma = \gamma_{\min}$ and $\sigma = \sigma_{\min}$;

end (if)

$Q \leftarrow \frac{1}{\sigma}Q + (\gamma - \frac{\underline{\lambda}(Q)}{\sigma})I$;

$c \leftarrow \frac{1}{\sigma}c + \frac{1}{2}(\frac{\underline{\lambda}(Q)}{\sigma} - \gamma)e$;

the center of the hypercube $0 \leq x \leq e$. *Then*

$$\|\tilde{x}^1 - p\| > \|\tilde{x}^2 - p\|$$

for $\gamma_1 < \gamma_2$.

Remark 4.1 The results in Theorem 4.3 hold for both unconstrained and constrained cases as long as we have convex minimization problems where the unique optimal solutions exist.

Theorem 4.3 essentially shows that as we decrease γ, the minimum of $f(x;\gamma)$ moves in a direction away from the center of the hypercube $0 \leq x \leq e$. We have argued that with a well-conditioned Hessian the integer points closer to the continuous minimum tend to have smaller objective values. Having the minimum move away from the center of the hypercube will make the integer points more "distinguishable" in terms of their objective values. Therefore, we would like γ^* to be small relative to σ^* in the solution to (4.17). To that end we choose $\zeta > 1$. From our empirical results, it is appropriate to set ζ to a value between 2 and 10.

We have run the Scaling and Shift algorithm on the QAPLIB instances with size $N \leq 100$ in two settings. The results are shown in Appendix B.3. In the first setting we have applied the algorithm to the Hessian of the original objective function; in the second setting we have applied the algorithm to the modified Hessian produced by the Matrix Reduction algorithm. In both settings we bound the condition number of the resulting Hessian by $\kappa = 100$. As we can see from the results shown, the magnitudes of σ and γ (which is equal to $\underline{\lambda}$ in Table B.3) are relatively small as compared with the magnitude of the spread of eigenvalues as shown in Table B.2. The values of σ and γ required to satisfy the bound on the condition number of the Hessian in the first setting are roughly two or three times as large as those in the second setting. Furthermore, in the first setting the sparsity of the Hessian is preserved whereas almost complete fill-in occurs in the second setting except for instances lipa20a–lipa90a. The fill-in is caused by the Matrix Reduction algorithm as discussed earlier.

4.4 Formulating the Relaxation Problems with Quadratic Cuts

In motivating the convex transformation in Section 4.2.1, we argued that the integer solutions closer to the minimum x^0 of (QPR) tend to have better objective values than those farther away from x^0. By a similar argument, we would like the optimal solution to (RPC) or (DPC) to be close to x^0 as well as being integral so the solution would be a "good" one to the QAP. In (RPC) the QAP objective function $q(x)$ plays a role in keeping the solution close to x^0 in the sense that moving in a direction away from x^0 is an ascent direction for $q(x)$. We replace $q(x)$ with the penalty term $\|x - x^0\|^2$ in (DPC) to make such a goal explicit. The quadratic cut confines the iterates within $q(x) < q(\hat{x})$ and hence an integer solution to (RPC) or (DPC), if found, must be an improved solution to the QAP.

As we have mentioned, we may get a noninteger solution by solving (RPC) or (DPC) starting from x^0. In that case, we could start from a different point and re-solve (RPC) or (DPC). We have observed the following with such a strategy:

- Since we have already failed to find an integer solution starting from x^0, there is no point in having $q(x)$ or $\|x - x^0\|^2$ in the objective function because their presence may lead the iterates back towards x^0, which in turn can result in a noninteger solution. Thus we drop $q(x)$ or $\|x - x^0\|^2$ from the objective function and start from a point $x^k \in \Omega$, $k \geq 1$, independent of x^0. We would like to either find an integer solution in Ω or determine that x^k leads to a noninteger solution. In the latter case, we try a different x^k and solve the problem repeatedly until we find an integer solution or have failed a prescribed number of times. Clearly, it is crucial that we be able to solve these problems quickly.

- After we drop $q(x)$ or $\|x - x^0\|^2$, the objective function contains only the quadratic penalty function, which is concave. If we solve the resulting optimization problem, convergence to a solution is not as fast as if the objective function were convex or at least locally convex. This suggests that we replace the quadratic penalty function with a penalty function that is at least locally convex. It is easy to see that the quartic penalty function $\|x \circ (e - x)\|^2$ is convex near the 0-1 integer points.

Thus, we solve the quartic penalty problem with the quadratic cut described in Sec-

tion 4.1.1, starting from a randomly generated point.

4.5 Random Starting Points for the Quartic Penalty Problems

In this section we will show how to compute a starting point x^k for (TPC). For the quartic penalty function, every 0-1 integer point is a local strict minimum. Hence, every 0-1 integer point has a *domain of attraction*. A domain of attraction of a point x is the set of points starting from which the algorithm will converge to x. Ideally we would like a starting point x^k for (TPC) to fall within the domain of attraction of some integer point in Ω. Consequently, starting from x^k, we will find an integer solution in Ω. However, these domains of attraction are not known *a priori*. Given that, it is reasonable to generate a starting point by sampling a point uniformly in Ω. We argue that if we start our algorithm from a sufficiently large number of such sampling points, we will almost surely find an integer point in Ω; i.e., as the number of sampling points goes to ∞, the probability of finding an integer point in Ω is 1.

To generate a point uniformly distributed in Ω, one would naturally think of the acceptance-rejection method in which Ω is enclosed by a region for which an efficient technique to generate uniform points is known. For example, a hyperellipsoid is such an enclosing region. However, the number of trial points that must be generated to obtain a point in Ω grows exponentially in the dimension of Ω [102, 105]. Alternatively, the *symmetric mixing* algorithm by Smith [105] is an approach based upon a continuous state Markov process with certain regularity conditions on the transition probability function. Under the assumption that the region $\mathcal{S}$ of interest is an open set of full dimension, the sequence of points generated by the symmetric mixing algorithm is shown to be *asymptotically uniformly distributed* on $\mathcal{S}$. A sequence $\{x^\ell\}$ is said to be asymptotically uniformly distributed on $\mathcal{S}$ if x^ℓ is uniformly distributed on $\mathcal{S}$ as $\ell \to \infty$.

In our case, Ω is not full dimensional due to the presence of equality constraints $Lx = b$. Let $p = (N-1)^2$. We partition x as follows.

$$x = \begin{bmatrix} y \\ z \end{bmatrix}, \quad y \in \mathbb{R}^m, \quad z \in \mathbb{R}^p \tag{4.18}$$

after possible rearranging the order of the coordinates in x. We rewrite $Lx = b$ as

$$\begin{bmatrix} L_y & L_z \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = b$$

where $L_y \in \mathbb{R}^{m\times m}$ is nonsingular and $L_z \in \mathbb{R}^{m\times p}$. Thus we have

$$y = L_y^{-1}b - L_y^{-1}L_z z = \tilde{b} - Gz \tag{4.19}$$

with $\tilde{b} \in \mathbb{R}^m$ and $G \in \mathbb{R}^{m\times p}$ defined as

$$\tilde{b} = L_y^{-1}b, \tag{4.20a}$$

$$G = L_y^{-1}L_z. \tag{4.20b}$$

Denoting $Q = \begin{bmatrix} Q_y & Q_u \\ Q_u^T & Q_z \end{bmatrix}$, we rewrite $q(x) \le q(\hat{x}) - \epsilon$ as

$$\tfrac{1}{2}\begin{bmatrix} y^T & z^T \end{bmatrix} \begin{bmatrix} Q_y & Q_u \\ Q_u^T & Q_z \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} + \begin{bmatrix} c_y^T & c_z^T \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} \le q(\hat{x}) - \epsilon,$$

$$\tfrac{1}{2}y^TQ_y y + \tfrac{1}{2}y^TQ_u z + \tfrac{1}{2}z^TQ_u^T y + \tfrac{1}{2}z^TQ_z z + c_y^T y + c_z^T z \le q(\hat{x}) - \epsilon. \tag{4.21}$$

Substituting (4.19) in (4.21) and simplifying, we get

$$\begin{aligned} &\tfrac{1}{2}z^T\left[G^TQ_yG - (Q_u^TG + G^TQ_u) + Q_z\right] z \\ &\quad + \left[Q_u^T\tilde{b} + c_z - G^T(Q_y\tilde{b} + c_y)\right]^T z + \tfrac{1}{2}\tilde{b}^TQ_y\tilde{b} + c_y^T\tilde{b} \le q(\hat{x}) - \epsilon, \\ &\tfrac{1}{2}z^T\tilde{Q}z + \tilde{c}^Tz \le \bar{q}(\hat{x}) \end{aligned}$$

with $\tilde{Q} \in \mathbb{R}^{p\times p}$, $\tilde{c} \in \mathbb{R}^p$ and $\tilde{q} \in \mathbb{R}$ defined as

$$\tilde{Q} = G^TQ_yG - (Q_u^TG + G^TQ_u) + Q_z, \tag{4.22a}$$

$$\tilde{c} = Q_u^T\tilde{b} + c_z - G^T(Q_y\tilde{b} + c_y), \tag{4.22b}$$

$$\bar{q}(\hat{x}) = q(\hat{x}) - \epsilon - \tfrac{1}{2}\tilde{b}^TQ_y\tilde{b} - c_y^T\tilde{b}. \tag{4.22c}$$

Note that

$$\tilde{Q} = \begin{bmatrix} -G^T & I \end{bmatrix} \begin{bmatrix} Q_y & Q_u \\ Q_u^T & Q_z \end{bmatrix} \begin{bmatrix} -G \\ I \end{bmatrix} = \begin{bmatrix} -G^T & I \end{bmatrix} Q \begin{bmatrix} -G \\ I \end{bmatrix}.$$

Since Q is positive definite, $\tilde{Q}$ is positive definite. Furthermore, $x \ge 0$ is equivalent to

$$\begin{aligned} &y \ge 0 \quad \text{or} \quad Gz \le \tilde{b}, \\ &z \ge 0. \end{aligned}$$

Thus we have the following full dimensional set

$$\mathcal{S} = \left\{ z \in \mathbb{R}^p : Gz \leq \tilde{b}, \tfrac{1}{2} z^T \tilde{Q} z + \tilde{c}^T z \leq \bar{q}(\hat{x}), z \geq 0 \right\}. \tag{4.23}$$

Now we apply the symmetric mixing algorithm to generate points in the interior of $\mathcal{S}$, denoted by $\text{int}(\mathcal{S})$. We outline the approach in Algorithm 4.3. For the moment, let us

Algorithm 4.3 (Smith [105], 1984) (*Symmetric Mixing*).

Input: region $\mathcal{S}$, an initial point $z^{k-1} \in \text{int}(\mathcal{S})$, and number of iterations r.

Output: z^k, a point asymptotically uniformly distributed on $\text{int}(\mathcal{S})$.

Let $w^0 = z^{k-1}$;

for $i = 1$ **to** r

Generate a random direction d^i by sampling a point uniformly on the unit hypersphere;

Determine the line segment $\mathcal{L} = \{\alpha \in \mathbb{R} : w^{i-1} + \alpha d^i \in \text{int}(\mathcal{S})\}$;

Generate τ^i from the uniform distribution on $\mathcal{L}$;

Let $w^i = w^{i-1} + \tau^i d^i$;

end (for)

Let $z^k = w^r$;

assume that initially a point $z^0 \in \text{int}(\mathcal{S})$ is available. We will discuss how to obtain an initial interior point of $\mathcal{S}$ in Section 5.1.5. Thus we can recursively generate a sequence of asymptotically independent uniformly distributed points $\{z^k\}$ in $\text{int}(\mathcal{S})$ for $k = 1, 2, \ldots$.

To sample a point uniformly on the unit hypersphere, we generate p independent normally distributed random numbers $v = (v_j)$, $j = 1, 2, \ldots, p$, where v_j is from $\mathcal{N}(0, 1)$, and let

$$d^i = \frac{v}{\|v\|}.$$

It is well-known (e.g., [34, 102]) that the random vector d^i is uniformly distributed on the unit hypersphere.

We determine the line segment $\mathcal{L}$ as follows. Let G_j, $j = 1, 2, \ldots, m$, denote row j

of G, i.e., $G^T = \left[\begin{array}{ccc} G_1^T & \cdots & G_m^T \end{array} \right]$. For constraints $Gz \leq \tilde{b}$, we define

$$\lambda_j = \frac{\tilde{b}_j - G_j z^{i-1}}{G_j d^i}, j = 1, 2, \ldots, m,$$

$$\lambda^+ = \min_{j=1,\ldots,m} \{\lambda_j : \lambda_j > 0\},$$

$$\lambda^- = \max_{j=1,\ldots,m} \{\lambda_j : \lambda_j < 0\}.$$

For constraints $z \geq 0$, we define

$$\mu_j = -\frac{z_j^{i-1}}{d_j^i}, j = 1, 2, \ldots, p,$$

$$\mu^+ = \min_{j=1,\ldots,p} \{\mu_j : \mu_j > 0\},$$

$$\mu^- = \max_{j=1,\ldots,p} \{\mu_j : \mu_j < 0\}.$$

For constraint $\frac{1}{2} z^T \tilde{Q} z + \tilde{c}^T z \leq \bar{q}(\hat{x})$, we define

$$\nu^+ = \frac{-\beta_1 + \sqrt{\beta_1^2 - 4\beta_0\beta_2}}{2\beta_0},$$

$$\nu^- = \frac{-\beta_1 - \sqrt{\beta_1^2 - 4\beta_0\beta_2}}{2\beta_0}$$

where $\beta_0 = \frac{1}{2}(d^i)^T \tilde{Q} d^i$, $\beta_1 = (\tilde{Q} z^{i-1} + \tilde{c})^T d^i$, and $\beta_2 = \frac{1}{2}(z^{i-1})^T \tilde{Q} z^{i-1} + \tilde{c}^T z^{i-1} - \bar{q}(\hat{x})$. It is easy to verify that $\beta_0 > 0$ for positive definite $\tilde{Q}$ and $\beta_2 < 0$ if z^{i-1} is in the interior of $\frac{1}{2} z^T \tilde{Q} z + \tilde{c}^T z \leq \bar{q}(\hat{x})$. Consequently, $\nu^+ > 0$ and $\nu^- < 0$. Let $\alpha^+ = \min\{\lambda^+, \mu^+, \nu^+\}$ and $\alpha^- = \max\{\lambda^-, \mu^-, \nu^-\}$. We have found the line segment

$$\mathcal{L} = \{\alpha \in \mathbb{R} : \alpha^- < \alpha < \alpha^+\}.$$

After we have obtained z^k, we can recover y^k by (4.19). Hence we have a starting point x^k for (TPC).

Chapter 5

Algorithm Implementation and Numerical Results

5.1 Implementation Details

We will now discuss the details of implementation of our algorithm for the QAP. We will first describe in Section 5.1.1 how we solve the relaxation problems. In Section 5.1.2 we will discuss the use of limited-memory BFGS update to approximate the Hessian for speedup. In Section 5.1.3 we will compare solving (RPC) vs. (DPC). Then we will show in Section 5.1.4 that (TPC) can be solved efficiently, one of the key factors for the performance of our algorithm. In Section 5.1.5 we will describe how to find an initial interior point to start the Symmetric Mixing algorithm. How we choose the values of the various parameters in our algorithm will be discussed in Section 5.1.6.

5.1.1 Solving Optimization Problems Using IPOPT

The relaxation problems introduced in Section 4.1.1 can be solved using a general nonlinear optimization algorithm. Generally speaking, due to the presence of the convex quadratic constraint in the relaxation problems with the quadratic cut, any robust implementation of a sequential quadratic programming algorithm or interior-point algorithm can be used to solve those optimization problems. In the implementation of our algorithm for the QAP, we favor an interior-point algorithm because we believe that the way the inequality constraints are handled in a typical nonlinear interior-point algorithm (e.g., [23, 113, 114]) is more conducive to success in finding integer solutions to (RPC) and

(DPC). In particular, we consider the following nonlinear optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^p$ and $h : \mathbb{R}^n \mapsto \mathbb{R}^r$ are smooth functions. In a typical nonlinear interior-point algorithm ([23, 113, 114]), the proposed algorithm approximately solves a sequence of barrier problems of the form

$$\begin{aligned} \underset{x,s}{\text{minimize}} \quad & f(x) - \mu_b \sum_{i=1}^{p} \log(s_i) \\ \text{subject to} \quad & g(x) + s = 0 \\ & h(x) = 0 \end{aligned}$$

for a decreasing sequence of barrier parameters μ_b converging to zero. In our context, the barrier problems are of the form

$$\begin{aligned} \underset{x,s}{\text{minimize}} \quad & f(x) - \mu_b \log(s) - \mu_b \sum_{i=1}^{n} \log(x_i) \\ \text{subject to} \quad & Lx = b \\ & q(x) + s = q(\hat{x}) - \epsilon \end{aligned} \tag{5.1}$$

where $f(x)$ represents the objective function of (RPC) or (DPC). In solving (RPC) or (DPC), the iterates could follow a path back toward $\hat{x}$, the incumbent solution. This is more likely when we have obtained an integer solution by solving (RXP) and are trying to find a better integer solution via (RPC)—(RXP) and (RPC) have the same objective function. For an appropriately chosen initial μ_b, the barrier term $-\mu_b \log(s)$ in (5.1) pushes the iterates away from the quadratic cut at the beginning and the barrier terms $-\mu_b \sum_{i=1}^{n} \log(x_i)$ prevent the iterates from initially getting too close to $\hat{x}$. Thus, a typical nonlinear interior-point algorithm with an appropriate choice of initial μ_b tends to avoid following a path back toward $\hat{x}$, which consequently gives us a better chance of finding an integer solution to (RPC) or (DPC). In this research, we use IPOPT, version 3.2.3, an interior-point code implementing the method of Wächter and Biegler [114], to solve the relaxation problems. For (RPC) or (DPC), we set "mu_strategy", the IPOPT option for the barrier parameter update strategy, to "monotone". For the other relaxation problems,

"mu_strategy" is set to "adaptive". With the "monotone" barrier parameter update strategy, the algorithm starts from some initial value of μ_b and monotonically decreases μ_b as it proceeds whereas with the "adaptive" barrier parameter update strategy, the algorithm dynamically adjusts (increases or decreases) the barrier parameter μ_b during the iterations.

5.1.2 Approximating the Hessian with an L-BFGS Update

The primary reason for using a limited-memory BFGS update is that the Hessian of the QAP objective function is usually very dense. As we can see from Table B.2, the majority of the original QAPLIB instances listed in the table have dense Hessians, e.g., most of them having more than 50% of the nonzero elements in the Hessian. When the Matrix Reduction algorithm is used, all the listed QAPLIB problems except lipa20a–lipa90a have almost completely dense Hessians. Consequently, the optimization algorithm using the exact Hessian becomes slow. The situation becomes even worse as the problem size increases. To speed up solving the relaxation problems, we use the limited-memory BFGS (L-BFGS) update to approximate the Hessian. In our implementation, we set "hessian_approximation", the IPOPT option indicating what Hessian information is used, to "exact" when the exact Hessian is to be used. When the L-BFGS update is to be used, we set "hessian_approximation" to "limited-memory".

An important issue we need to consider when we use the L-BFGS approximation is the convergence of the optimization algorithms. Generally speaking, for difficult nonlinear optimization problems, using the L-BFGS update may result in convergence problems. That is, it may cause the algorithm to diverge on a problem on which the algorithm would converge if the exact Hessian were used. In our case, since we have a well-conditioned positive definite Hessian, using the L-BFGS update to approximate the Hessian does not exhibit this difficulty.

We have run steps (1) and (2) of our algorithm for the QAP, as shown in Figure 4.1, on a subset of the QAPLIB instances using both the exact Hessian and L-BFGS update. For the pre-conditioning, we have applied the Scaling and Shift algorithm but not the Matrix Reduction algorithm. The results are shown in Table 5.1. Column "pubval" shows the objective values of the published optimal solutions or best known solutions. "solval" represents the objective value of our solutions using both the exact Hessian and

Table 5.1 Exact Hessian vs. L-BFGS Approximation

Problem	pubval	solval		# iter		CPU[1] (sec)	
		exact	lbfgs	exact	lbfgs	exact	lbfgs
bur26c	5426795	5628125	5544761	163	58	73.3	0.64
chr12a	9552	29072	32432	111	21	0.5	0.03
chr15b	7990	34414	45372	133	34	1.6	0.06
chr18a	11098	58398	38658	113	37	2.8	0.09
chr20c	14142	46722	25842	123	49	5.0	0.11
chr22a	6156	10864	11128	127	41	8.6	0.12
chr25a	3796	14258	10966	149	54	20.4	0.19
els19	17212548	32095222	38323406	124	51	5.5	0.12
esc16d	16	28	48	96	20	0.7	0.05
esc32a	130	276	336	151	26	44.3	0.17
had12	1652	1778	1798	98	25	1.0	0.05
had14	2724	2766	2770	132	22	2.9	0.05
had16	3720	3878	4134	145	32	6.2	0.09
had18	5358	5742	5692	136	62	10.7	0.23
had20	6922	7446	7512	124	41	17.0	0.22
kra30a	88900	112250	106050	257	50	224.5	0.50
kra32	88700	116510	112850	279	44	307.0	0.48
lipa20b	27076	29107	27076	110	45	13.3	0.22
lipa30a	13178	13725	13584	180	63	314.3	1.25
nug12	578	734	702	164	22	1.6	0.05
nug15	1150	1488	1464	158	28	4.0	0.06
nug17	1732	2124	1918	120	26	5.7	0.08
nug20	2570	2900	2788	178	44	20.0	0.19
nug22	3596	4008	4330	216	61	42.6	0.31
nug24	3488	4468	4144	254	42	74.0	0.30
nug27	5234	6288	5716	208	64	118.3	0.66
nug30	6124	7330	6862	282	45	413.9	0.67
rou12	235528	260900	279742	104	18	1.0	0.03
rou15	354210	444336	411710	110	27	3.0	0.08
rou20	725522	844792	795974	116	39	15.2	0.20
scr12	31410	43422	39948	149	35	1.2	0.05
scr15	51140	79644	82942	126	35	2.4	0.06
scr20	110030	187576	182044	211	67	15.1	0.20
sko42	15812	19214	17030	190	97	2476.6	7.23
ste36a	9526	14092	13702	209	104	130.3	1.44
tai12a	224416	276546	279476	105	23	1.0	0.05
tai15b	51765268	53530792	53606615	212	25	6.0	0.05
tai17a	491812	574930	544786	108	25	6.0	0.09
tai20b	122455319	146002742	143654171	132	48	16.0	0.23
tai25a	1167256	1302072	1392138	129	36	65.7	0.38
tai30b	637117113	798646283	798289184	179	66	292.7	1.11
tai35a	2422002	2792988	2562302	134	74	654.7	2.52
tai40b	637250948	905391924	821196315	177	101	1720.5	4.38
tho30	149936	180602	177196	187	47	203.3	0.55
tho40	240516	290582	265608	203	72	1600.3	1.88

[1]CPU time is based on an Intel Pentium 4 2.80 GHz processor.

L-BFGS update; "# iter" stands for the number of iterations. Note that the majority of the CPU time shown in Table 5.1 is spent on solving (RXP); the time spent on (QPR) is almost negligible. As is shown, although the objective values of the solutions can be better or worse than those with the exact Hessian, the L-BFGS approximation usually results in extraordinary speedup in solving these optimization problems, especially the instances of larger sizes.

5.1.3 Solving (RPC) vs. (DPC)

In our algorithm for the QAP, we can solve either (RPC) or (DPC) to find a better solution after we have solved (RXP). To compare the performance of our algorithm using both formulations, we have run steps (1) through (4) in Figure 4.1 on a subset of the QAPLIB instances. That is, if we find an integer solution in step (4) we go back to step (3); otherwise we exit. For the pre-conditioning, we have applied both the Matrix Reduction algorithm (except for bur26c whose flow and distance matrices are both asymmetric) and the Scaling and Shift algorithm. The L-BFGS update has been used to approximate the Hessian.

The results are shown in Table 5.2. The column headers "pubval" and "solval" have connotations similar to those used in Table 5.1. In Table 5.2, "# cuts" denotes the number of successive times we obtained an improved integer solution by solving (RPC) or (DPC). The CPU time pertains to solving (RPC) or (DPC) only, i.e., not including that in solving (QPR) and (RXP). As we can see, solving (RPC) yields better solutions than using formulation (DPC) for some problems whereas the contrary is true for other problems, indicating that the appropriate formulation is problem dependent. One strategy is to solve both (RPC) and (DPC) and then choose the better of the two solutions produced by them.

5.1.4 Solving (TPC) Efficiently

The main point in solving (TPC) described in Section 4.1 is that we start from multiple trial points in Ω as defined in (4.2) and determine if one of those points leads to an integer solution. The more trials points we attempt, the greater are the chances of finding an integer solution in Ω. Thus, the ability to solve (TPC) efficiently is a key to the success in finding an improved solution to the QAP.

Table 5.2 Quadratic Cut: Solving (RPC) vs. (DPC)

Problem	pubval	solval		# cuts		CPU[1] (sec)	
		RPC	DPC	RPC	DPC	RPC	DPC
bur26c	5426795	5587938	5569676	0	0	1.2	0.9
chr12a	9552	15398	15136	0	1	0.1	0.3
chr15b	7990	17644	17644	0	0	0.2	0.1
chr18a	11098	27920	27920	0	0	0.3	0.2
chr20c	14142	24442	24442	0	0	0.4	0.4
chr22a	6156	8268	9458	1	0	1.0	0.5
chr25a	3796	9722	8504	1	0	1.6	1.0
els19	17212548	25347796	31160654	1	0	0.9	0.3
esc16d	16	20	24	1	0	0.5	0.3
esc32a	130	324	202	1	0	4.8	5.9
esc64a	116	134	136	0	0	91.7	107.2
had12	1652	1682	1682	0	0	0.1	0.1
had14	2724	2744	2744	0	0	0.1	0.1
had16	3720	3780	3780	0	0	0.2	0.1
had18	5358	5430	5430	0	0	0.2	0.2
had20	6922	7196	7150	1	1	0.7	0.5
kra30a	88900	108700	107040	1	0	3.3	1.3
kra32	88700	104010	106270	1	0	5.1	3.3
lipa20b	27076	27076	27076	1	1	0.5	0.3
lipa30a	13178	13524	13525	0	0	1.7	1.4
lipa50a	62093	62849	63117	0	0	20.0	13.0
lipa80b	7763962	7763962	7763962	0	0	91.1	64.8
nug12	578	662	626	0	0	0.1	0.1
nug15	1150	1312	1336	0	1	0.1	0.2
nug17	1732	1844	1844	0	0	0.2	0.2
nug20	2570	2854	2824	1	0	0.7	0.4
nug25	3744	4152	4324	0	0	0.9	1.0
nug30	6124	6822	6584	0	0	2.0	2.0
rou12	235528	245718	245718	0	0	0.1	0.1
rou15	354210	387978	387978	0	0	0.1	0.1
rou20	725522	774224	774224	0	0	0.3	0.2
scr12	31410	36628	40206	0	1	0.1	0.1
scr15	51140	64450	62642	0	0	0.1	0.1
scr20	110030	160320	164522	1	0	0.7	0.3
sko42	15812	17056	17548	0	0	11.9	7.1
sko64	48498	51260	53446	0	0	83.9	77.8
ste36a	9526	12596	13504	0	0	4.8	3.3
tai12a	224416	236436	236436	0	0	0.1	0.1
tai15b	51765268	52485675	52766595	1	0	0.5	0.1
tai20b	122455319	150009318	149128616	0	2	0.7	0.9
tai30b	637117113	782365838	822906526	1	0	10.6	4.2
tai40b	637250948	845276670	813760201	0	0	10.2	8.6
tai64c	1855928	2235970	2209780	0	0	212.2	127.1
tho30	149936	167852	177828	0	0	1.9	1.5
tho40	240516	277278	269536	0	0	6.3	6.3
wil50	48816	49880	50500	0	0	18.2	20.8

Since the objective function of (TPC) is convex near a 0-1 integer point, a nonlinear optimization algorithm is able to converge rapidly towards an integer point (possibly infeasible). If such an integer point is in Ω, we have obtained an integer solution to (TPC); otherwise, we have determined that the given starting point does not lead to an integer solution and we should try a different starting point. In addition, using the L-BFGS update to approximate the Hessian, as discussed in Section 5.1.2, further speeds up solving (TPC).

To illustrate how efficiently (TPC) can be solved, we have run steps (5) and (6) of our algorithm multiple times on a subset of the QAPLIB instances. That is, for each instance in the set, we started from 50 different randomly generated initial points and solved (TPC). The results are shown in Table 5.3. Column "avg itr" represents the

Table 5.3 Solving Multiple Instances of (TPC) with LBFGS

Problem	avg itr	CPU[1] (sec)	Problem	avg itr	CPU[1] (sec)
bur26c	11.5	0.45	nug15	8.9	0.07
chr12a	10.9	0.06	nug17	8.9	0.11
chr15b	11.1	0.08	nug20	9.0	0.17
chr18a	11.0	0.11	nug25	10.9	0.36
chr20c	11.1	0.14	nug30	10.2	0.86
chr22a	9.9	0.20	rou12	11.8	0.06
chr25a	11.7	0.31	rou15	11.9	0.10
els19	12.3	0.15	rou20	11.6	0.19
esc16d	9.0	0.08	scr12	11.5	0.06
esc32a	9.6	0.93	scr15	11.4	0.10
esc64a	8.2	12.37	scr20	11.3	0.16
had12	9.3	0.05	sko42	9.0	3.15
had14	8.9	0.08	sko64	9.5	14.98
had16	9.3	0.09	ste36a	10.4	1.51
had18	9.4	0.12	tai12a	11.1	0.05
had20	9.4	0.18	tai15b	12.0	0.12
kra30a	9.3	0.81	tai20b	10.0	0.15
kra32	10.6	1.04	tai30b	11.6	0.91
lipa20b	9.8	0.19	tai40b	11.5	2.55
lipa30a	10.0	0.91	tai64c	10.6	12.52
lipa50a	9.1	6.12	tho30	10.6	0.85
lipa80b	9.8	39.00	tho40	9.7	2.31
nug12	9.0	0.06	wil50	9.9	6.04

average number of iterations taken to solve (TPC). The CPU time is the average time spent on solving (TPC). As we can see, (TPC) can indeed be solved quickly. Taking the nontrivial instance lipa80b for example, it takes an average of 39 seconds to solve (TPC).

5.1.5 Initial Interior Point for the Symmetric Mixing Algorithm

To use the Symmetric Mixing algorithm outlined in Section 4.5 for generating asymptotically uniformly distributed points in the interior of $\mathcal{S}$, where $\mathcal{S}$ is defined in (4.23), we initially need a point $z^0 \in \text{int}(\mathcal{S})$. We set $\bar{x}$ to x^0, the optimal solution to (QPR), and extract $\bar{z}$ according to the partition in (4.18), i.e.,

$$\bar{x} = \begin{bmatrix} \bar{y} \\ \bar{z} \end{bmatrix}, \quad \bar{y} \in \mathbb{R}^m, \quad \bar{z} \in \mathbb{R}^p.$$

Suppose that $\text{int}(\mathcal{S})$ is nonempty. It is easy to see that $\bar{z}$ satisfies $\frac{1}{2}\bar{z}^T\tilde{Q}\bar{z} + \tilde{c}^T\bar{z} < \bar{q}(\hat{x})$ with strict inequality. $\tilde{Q}$, $\tilde{c}$ and $\bar{q}(\hat{x})$ are defined in (4.22a)–(4.22c). Also the following holds for $\bar{z}$:

$$\begin{aligned} G\bar{z} &\leq \tilde{b}, \\ \bar{z} &\geq 0, \end{aligned}$$

where G and $\tilde{b}$ are defined in (4.20a)–(4.20b). However, the strict inequalities may not be satisfied. Thus, we start from $\bar{z}$ and solve the following unconstrained optimization problem

$$\underset{z}{\text{minimize}}\, f(z) \triangleq -\log(\bar{q}(\hat{x}) - \tfrac{1}{2}z^T\tilde{Q}z - \tilde{c}^T z) - \sum_{i=1}^{m}\log(\tilde{b}_i + \delta - G_i z) - \sum_{i=1}^{p}\log(z_i + \delta) \tag{5.2}$$

for an interior point in $\mathcal{S}$. δ is chosen such that the logarithmic terms in (5.2) are properly defined. δ should be small enough so that the optimal solution to (5.2) is in the interior of $\mathcal{S}$. In our implementation, δ is set to 10^{-6}.

The Hessian of the objective function of (5.2) is

$$\nabla^2 f(z) = \frac{1}{\theta}\tilde{Q} + \frac{1}{\theta^2}(\tilde{Q}z + \tilde{c})(\tilde{Q}z + \tilde{c})^T + G^T\Gamma_1^{-2}G + \Gamma_2^{-2}$$

where $\theta = \bar{q}(\hat{x}) - \frac{1}{2}z^T\tilde{Q}z - \tilde{c}^T z$, $\Gamma_1 = \text{diag}(\tilde{b} - Gz) + \delta I$, and $\Gamma_2 = \text{diag}(z) + \delta I$. Note that $\tilde{Q}$ is positive definite and for $w \in \mathbb{R}^p$, we have

$$w^T(\tilde{Q}z + \tilde{c})(\tilde{Q}z + \tilde{c})^T w = \left((\tilde{Q}z + \tilde{c})^T w\right)^2 \geq 0.$$

Thus, $f(z)$ is strictly convex and (5.2) has a unique minimum. We choose z^0 in the Symmetric Mixing algorithm to be the minimum of (5.2).

5.1.6 Choice of Parameters

Penalty Parameter μ in (RXP) and (RPC). In (RXP) and (RPC), we choose the penalty parameter $\mu > \frac{1}{2}\overline{\lambda}(Q)$ through computing the largest eigenvalue of Q rather than using the estimate of μ by Proposition 3.6. This is because using the property that the eigenvalues of Q are formed from all possible products of the eigenvalues of the flow matrix F and distance matrix D, computing the largest eigenvalue of Q is not very expensive for the QAP instances we have run, especially those from the QAPLIB. Moreover, the method in Proposition 3.6 yields an unnecessarily large value for μ.

ϵ in the Quadratic Cut. For all the test problems we have run, the elements of the flow matrix F and distance matrix D are all integers, and there is no linear cost term; i.e., C in (1.1) is zero. By formulation (1.1), it is easy to see that any solution better than $\hat{x}$ must have an objective value that is at least one less than that at $\hat{x}$. That is, we could choose ϵ to be one and the quadratic cut will not exclude any solutions that are better than $\hat{x}$. Note that we have scaled the objective function after we apply the Scaling and Shift algorithm. Thus, we set ϵ to $1/\sigma$.

Penalty Parameter ρ in (DPC). As we have shown in Section 4.1.1, we need to choose $\rho > 1$. Based on our numerical experience we have set a default value of 2 for ρ.

Number of Iterations r in the Symmetric Mixing Algorithm. As we have discussed in Section 4.5, a point generated by the Symmetric Mixing algorithm is uniformly distributed on $\mathcal{S}$ when the number of iterations r goes to ∞. However, in a practical implementation of the algorithm, we have to choose a finite value for r. Through experiments, we have found out that a default value of 200 for r yields reasonably good results for the QAP test problems we have run.

5.2 Results on the QAPLIB Instances

We have run our algorithm by the default settings on the QAPLIB [22] instances with the number of locations $N < 100$. More specifically, we have applied the pre-conditioning algorithms, i.e., the Matrix Reduction algorithm and the Scaling and Shift algorithm, before the main iterations as outlined in Figure 4.1. The results are reported in Table 5.4. In the table, "ObjVal" indicates the objective values of the published solutions or the solutions obtained by our algorithm. For the entries under column "Type", "OPT" stands for *optimum* and "BKS" for *best known solution*. We have also computed the *gap*, a commonly used measure of the quality of a solution by an approximate algorithm, defined as follows:

$$\text{gap} = \frac{\hat{\text{Z}} - \text{Z}^*}{\text{Z}^*} \times 100 \tag{5.3}$$

where Z^* is the optimal or best known objective value and $\hat{Z}$ is the objective value of the solution obtained by our algorithm. The CPU time in the format *minutes* : *seconds* is based on an Intel Pentium 4 2.80 GHz processor. Note that we did not run the trivial instance esc16f whose flow matrix is 0—Any feasible solution is an optimal solution to instance esc16f.

Table 5.4 Results on the QAPLIB Instances with $N < 100$

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
bur26a	26	5426670	OPT	5575423	2.74	3.61
bur26b	26	3817852	OPT	3938345	3.16	4.55
bur26c	26	5426795	OPT	5664686	4.38	3.38
bur26d	26	3821225	OPT	4027956	5.41	3.55
bur26e	26	5386879	OPT	5469384	1.53	3.61
bur26f	26	3782044	OPT	3937035	4.10	4.41
bur26g	26	10117172	OPT	10459179	3.38	3.64
bur26h	26	7098658	OPT	7256589	2.22	4.39
chr12a	12	9552	OPT	15398	61.20	0.52
chr12b	12	9742	OPT	16592	70.31	0.39
chr12c	12	11156	OPT	14214	27.41	0.56
chr15a	15	9896	OPT	29806	201.19	0.88
chr15b	15	7990	OPT	17644	120.83	0.67
chr15c	15	9504	OPT	29210	207.34	0.86
chr18a	18	11098	OPT	27920	151.58	1.11
chr18b	18	1534	OPT	2542	65.71	1.59
chr20a	20	2192	OPT	3948	80.11	1.41

Continued on next page

Continued from previous page

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
chr20b	20	2298	OPT	6880	199.39	1.94
chr20c	20	14142	OPT	24442	72.83	1.28
chr22a	22	6156	OPT	8268	34.31	2.88
chr22b	22	6194	OPT	9354	51.02	2.73
chr25a	25	3796	OPT	9236	143.31	2.59
els19	19	17212548	OPT	25347796	47.26	2.13
esc16a	16	68	OPT	94	38.24	1.06
esc16b	16	292	OPT	294	0.68	1.23
esc16c	16	160	OPT	166	3.75	1.51
esc16d	16	16	OPT	30	87.50	1.14
esc16e	16	28	OPT	52	85.71	1.53
esc16f	16	0	OPT	Not run		
esc16g	16	26	OPT	42	61.54	1.16
esc16h	16	996	OPT	1066	7.03	1.31
esc16i	16	14	OPT	24	71.43	1.17
esc16j	16	8	OPT	10	25.00	1.26
esc32a	32	130	BKS	296	127.69	22.45
esc32b	32	168	BKS	300	78.57	18.97
esc32c	32	642	BKS	686	6.85	19.53
esc32d	32	200	BKS	258	29.00	17.83
esc32e	32	2	OPT	6	200.00	23.89
esc32f	32	2	OPT	6	200.00	23.88
esc32g	32	6	OPT	12	100.00	21.16
esc32h	32	438	BKS	478	9.13	12.77
esc64a	64	116	BKS	130	12.07	6:31.78
had12	12	1652	OPT	1682	1.82	0.58
had14	14	2724	OPT	2744	0.73	0.45
had16	16	3720	OPT	3780	1.61	0.72
had18	18	5358	OPT	5430	1.34	0.91
had20	20	6922	OPT	7140	3.15	1.69
kra30a	30	88900	OPT	100780	13.36	6.83
kra30b	30	91420	OPT	106780	16.80	12.08
kra32	32	88700	OPT	104970	18.34	8.52
lipa20a	20	3683	OPT	3795	3.04	1.39
lipa20b	20	27076	OPT	27076	0.00	1.47
lipa30a	30	13178	OPT	13608	3.26	5.42
lipa30b	30	151426	OPT	151426	0.00	4.80
lipa40a	40	31538	OPT	32085	1.73	16.23
lipa40b	40	476581	OPT	476581	0.00	13.61
lipa50a	50	62093	OPT	62849	1.22	47.39
lipa50b	50	1210244	OPT	1210244	0.00	52.84
lipa60a	60	107218	OPT	108256	0.97	1:45.31
lipa60b	60	2520135	OPT	2520135	0.00	2:17.01
lipa70a	70	169755	OPT	170623	0.51	4:33.27
lipa70b	70	4603200	OPT	4603200	0.00	3:04.59
lipa80a	80	253195	OPT	255134	0.77	6:32.61
lipa80b	80	7763962	OPT	7763962	0.00	8:18.31
lipa90a	90	360630	OPT	364664	1.12	11:25.53
lipa90b	90	12490441	OPT	12490441	0.00	8:27.56
nug12	12	578	OPT	626	8.30	0.91

Continued on next page

Continued from previous page

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
nug14	14	1014	OPT	1072	5.72	0.52
nug15	15	1150	OPT	1324	15.13	0.67
nug16a	16	1610	OPT	1690	4.97	0.69
nug16b	16	1240	OPT	1354	9.19	0.89
nug17	17	1732	OPT	1844	6.47	0.75
nug18	18	1930	OPT	2098	8.70	1.09
nug20	20	2570	OPT	2810	9.34	2.42
nug21	21	2438	OPT	2770	13.62	2.52
nug22	22	3596	OPT	3980	10.68	4.23
nug24	24	3488	OPT	3984	14.22	2.39
nug25	25	3744	OPT	4210	12.45	3.80
nug27	27	5234	OPT	6026	15.13	4.00
nug28	28	5166	OPT	5854	13.32	4.38
nug30	30	6124	OPT	6992	14.17	6.06
rou12	12	235528	OPT	245718	4.33	0.41
rou15	15	354210	OPT	387978	9.53	0.61
rou20	20	725522	OPT	774224	6.71	1.16
scr12	12	31410	OPT	38274	21.85	0.61
scr15	15	51140	OPT	66914	30.84	0.61
scr20	20	110030	OPT	159960	45.38	1.34
sko42	42	15812	BKS	17730	12.13	29.88
sko49	49	23386	BKS	25272	8.06	59.47
sko56	56	34458	BKS	38124	10.64	1:32.75
sko64	64	48498	BKS	51690	6.58	4:06.50
sko72	72	66256	BKS	69990	5.64	5:51.66
sko81	81	90998	BKS	95574	5.03	12:12.05
sko90	90	115534	BKS	125012	8.20	14:37.48
ste36a	36	9526	OPT	14212	49.19	13.05
ste36b	36	15852	OPT	37778	138.32	22.81
ste36c	36	8239110	OPT	11546668	40.14	16.08
tai10a	10	135028	OPT	153250	13.49	0.52
tai10b	10	1183760	OPT	1415640	19.59	0.50
tai12a	12	224416	OPT	236436	5.36	0.61
tai12b	12	39464925	OPT	51433727	30.33	0.70
tai15a	15	388214	OPT	421164	8.49	1.09
tai15b	15	51765268	OPT	52485675	1.39	1.48
tai17a	17	491812	OPT	542856	10.38	1.00
tai20a	20	703482	OPT	770440	9.52	1.31
tai20b	20	122455319	OPT	147889773	20.77	3.00
tai25a	25	1167256	OPT	1269666	8.77	2.86
tai25b	25	344355646	OPT	436069858	26.63	9.20
tai30a	30	1818146	BKS	1965172	8.09	7.55
tai30b	30	637117113	BKS	777619629	22.05	13.36
tai35a	35	2422002	BKS	2572476	6.21	9.89
tai35b	35	283315445	BKS	354305731	25.06	32.27
tai40a	40	3139370	BKS	3378478	7.62	19.63
tai40b	40	637250948	BKS	815986827	28.05	34.48
tai50a	50	4938796	BKS	5209870	5.49	53.22
tai50b	50	458821517	BKS	600132348	30.80	2:32.42
tai60a	60	7205962	BKS	7556190	4.86	2:06.83

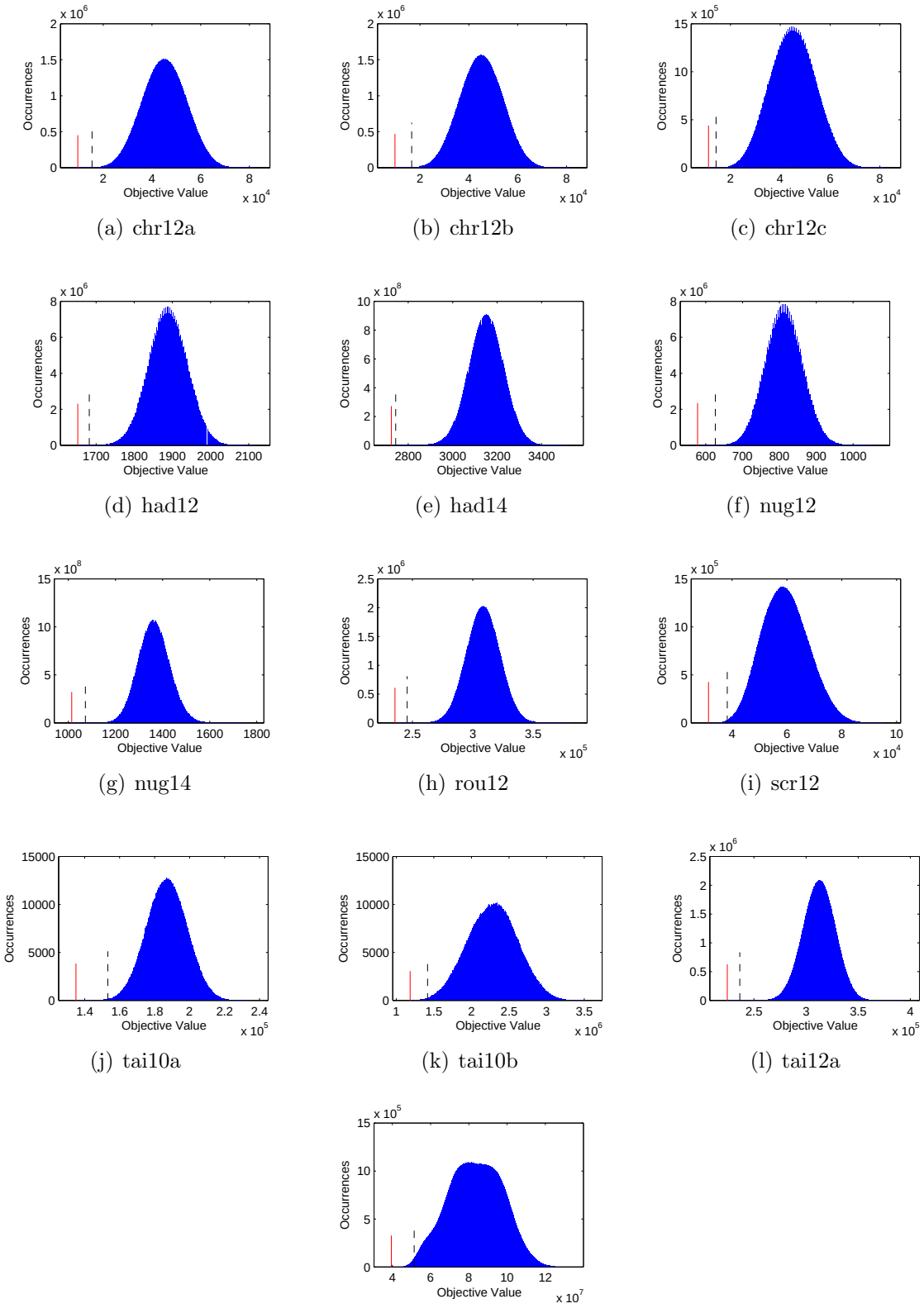
Continued on next page

Continued from previous page

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
tai60b	60	608215054	BKS	726133324	19.39	2:17.06
tai64c	64	1855928	BKS	2041062	9.98	28:46.23
tai80a	80	13515450	BKS	14536796	7.56	6:31.76
tai80b	80	818415043	BKS	952760408	16.42	7:56.01
tho30	30	149936	OPT	177632	18.47	6.39
tho40	40	240516	BKS	280050	16.44	17.69
wil50	50	48816	BKS	50934	4.34	1:12.13

In terms of the quality of the solutions by our algorithm, for almost half of the QAPLIB instances in Table 5.4, the gap as defined in (5.3) is less than 10%. For the 13 QAPLIB instances with $N \leq 14$, we enumerated all the feasible solutions. Then we divided the solutions into 1000 equal intervals between the best and worst objective values and counted the number of occurrences in these intervals. The histograms are plotted in Figure 5.1. It appears that the objective values of the feasible QAP solutions follow a normal distribution. We also plotted our solutions along with the optimal solutions on the histograms. In Figure 5.1, the solid vertical lines indicate the optimal solutions and the dashed vertical lines show the solutions by our algorithm. If we define a good solution to be one that is better than the majority of the feasible solutions, the solutions to all these 13 instances by our algorithm are good solutions. Note that for instances chr12a and chr12b, the gap is 61.2% and 70.31% respectively. This leads us to believe that a measure by gap can give a bias towards the very few feasible solutions (including the optimum) at the long left tail of a normal distribution and hence distort the quality of otherwise good solutions.

As for the solution times, in Table 5.5 we compare the CPU time of our algorithm with that of the exact algorithms for the recently solved large QAPLIB instances that we showed in Table 1.3. The column header "Exact" is for the exact algorithm that optimally solved these instances and "Proposed" for our algorithm. "Gap" measures the solutions by our algorithm. As we can see, the solution times of our algorithm is virtually negligible compared to those of the exact algorithms. Moreover, the gap is not very large for the instances except ste36a–ste36c.



(a) chr12a (b) chr12b (c) chr12c

(d) had12 (e) had14 (f) nug12

(g) nug14 (h) rou12 (i) scr12

(j) tai10a (k) tai10b (l) tai12a

(m) tai12b

Figure 5.1 Performance of the Algorithm on the QAPLIB Instances with $N \leq 14$

Table 5.5 Comparison on Recently Solved Large QAPLIB Instances

Problem	CPU Time		Gap %
	Exact (days)	Proposed (secs)	
kra30a	99	6.83	13.36
kra30b	1527	12.08	16.80
kra32	5536	8.52	18.34
nug27	113	4.00	15.13
nug28	722	4.38	13.32
nug30	3999	6.06	14.17
ste36a	18	13.05	49.19
ste36b	60	22.81	138.32
ste36c	200	16.08	40.14
tai25a	394	2.86	8.77
tho30	8997	6.39	18.47

5.3 Results on Other Published QAP Test Problems

We have also run our algorithm on the other three sets of QAP test problems described in Section 1.4. The pre-conditioning algorithms have been applied before the main iterations of our algorithm for these problems. The results are shown in Tables 5.6–5.8, which have the same column headers and entries in column "Type" as in Table 5.4. The CPU time is also based on an Intel Pentium 4 2.80 GHz processor.

Instances by Drezner, Hahn and Taillard [30]. The results on these instances (with $N < 100$) are summarized in Table 5.6. By the measure of gap, it seems that our algorithm does not perform well on this set of problems. This is probably due to the rugged landscape of the feasible regions of these instances as described in Section 1.4. Since our continuous optimization algorithm relies on the information of the feasible region in search for solutions, a steep landscape of the feasible region may well affect the performance. In spite of that, the solution times for these instances are relatively small.

Table 5.6 Results on the Instances by Drezner et al. with $N < 100$

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
dre15	15	306	OPT	774	152.94	0.95
dre18	18	332	OPT	886	166.87	1.11
dre21	21	356	OPT	1166	227.53	2.44
dre24	24	396	OPT	1146	189.39	3.13
dre28	28	476	OPT	1150	141.60	4.47
dre30	30	508	OPT	1306	157.09	5.72
dre42	42	764	OPT	2544	232.98	27.06
dre56	56	1086	OPT	3084	183.98	1:43.36
dre72	72	1452	OPT	2988	105.79	6:47.04
dre90	90	1838	OPT	6220	238.41	12:40.42
tai27e01	27	2558	OPT	3176	24.16	3.83
tai27e02	27	2850	OPT	4410	54.74	4.59
tai27e03	27	3258	OPT	5504	68.94	6.75
tai27e04	27	2822	OPT	5332	88.94	6.91
tai27e05	27	3074	OPT	3546	15.35	4.25
tai27e06	27	2814	OPT	7458	165.03	4.58
tai27e07	27	3428	OPT	4966	44.87	4.45
tai27e08	27	2430	OPT	4388	80.58	4.87
tai27e09	27	2902	OPT	3506	20.81	6.69
tai27e10	27	2994	OPT	3532	17.97	7.41
tai27e11	27	2906	OPT	4502	54.92	4.34
tai27e12	27	3070	OPT	4726	53.94	6.50
tai27e13	27	2966	OPT	3462	16.72	6.20
tai27e14	27	3568	OPT	7422	108.02	4.61
tai27e15	27	2628	OPT	4474	70.24	4.28
tai27e16	27	3124	OPT	3962	26.82	6.98
tai27e17	27	3840	OPT	5228	36.15	6.69
tai27e18	27	2758	OPT	7618	176.21	7.78
tai27e19	27	2514	OPT	3496	39.06	4.23
tai27e20	27	2638	OPT	4170	58.07	5.19
tai45e01	45	6412	OPT	21432	234.25	37.72
tai45e02	45	5734	BKS	18246	218.21	35.37
tai45e03	45	7438	BKS	16666	124.07	39.75
tai45e04	45	6698	BKS	13684	104.30	44.28
tai45e05	45	7274	BKS	25714	253.51	1:24.41
tai45e06	45	6612	BKS	10374	56.90	30.81
tai45e07	45	7526	BKS	16254	115.97	1:44.62
tai45e08	45	6554	BKS	12076	84.25	59.95
tai45e09	45	6648	BKS	20568	209.39	42.19
tai45e10	45	8286	BKS	31740	283.06	1:25.80
tai45e11	45	6510	BKS	15378	136.22	56.58
tai45e12	45	7510	BKS	17550	133.69	40.72
tai45e13	45	6120	BKS	21738	255.20	59.02
tai45e14	45	6854	BKS	13704	99.94	49.67
tai45e15	45	7394	BKS	11808	59.70	56.94
tai45e16	45	6520	BKS	17194	163.71	49.22
tai45e17	45	8806	BKS	13980	58.76	59.92
tai45e18	45	6906	BKS	13786	99.62	43.87
tai45e19	45	7170	BKS	22336	211.52	39.64

Continued on next page

Continued from previous page

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
tai45e20	45	6510	BKS	14990	130.26	54.03
tai75e01	75	14488	OPT	24916	71.98	6:48.53
tai75e02	75	14444	BKS	30526	111.34	7:20.37
tai75e03	75	14154	BKS	43146	204.83	11:55.23
tai75e04	75	13694	BKS	47736	248.59	10:48.25
tai75e05	75	12884	BKS	27546	113.80	6:42.32
tai75e06	75	12554	BKS	23706	88.83	7:25.70
tai75e07	75	13782	BKS	31468	128.33	6:29.44
tai75e08	75	13948	BKS	30974	122.07	6:47.31
tai75e09	75	12650	BKS	21412	69.26	6:26.47
tai75e10	75	14192	BKS	30436	114.46	10:23.98
tai75e11	75	15250	BKS	32098	110.48	10:28.92
tai75e12	75	12760	BKS	31228	144.73	8:28.57
tai75e13	75	13024	BKS	35388	171.71	7:53.45
tai75e14	75	12604	BKS	23666	87.77	6:14.20
tai75e15	75	14294	BKS	27450	92.04	9:41.09
tai75e16	75	14204	BKS	25176	77.25	8:17.50
tai75e17	75	13210	BKS	30466	130.63	7:01.82
tai75e18	75	13500	BKS	29550	118.89	6:32.97
tai75e19	75	12060	BKS	23470	94.61	6:19.22
tai75e20	75	15260	BKS	30976	102.99	11:01.59

Instances by Palubeckis [91]. The results on these instances (with $N < 100$) are listed in Table 5.7. By the measure of gap, our algorithm performs well on this set of problems. Moreover, the solution times are relatively small.

Table 5.7 Results on the Instances by Palubeckis with $N < 100$

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
Inst20	20	81536	OPT	83036	1.84	1.49
Inst30	30	271092	OPT	278900	2.88	7.08
Inst40	40	837900	OPT	848464	1.26	21.64
Inst50	50	1840356	OPT	1857684	0.94	54.64
Inst60	60	2967464	OPT	3034136	2.25	1:58.63
Inst70	70	5815290	OPT	5856766	0.71	4:10.94
Inst80	80	6597966	OPT	6656386	0.89	8:20.28

Instances by Stützle and Fernandes [108]. The results on these instances (with $N < 100$) are contained in Table 5.8. Due to their varied characteristics, these instances are more difficult than the QAPLIB instances. There are about a quarter of the instances

in the table for which the gap of the solutions by our algorithm is less than 10%. For several instances, we have seen unusually large values ($> 1000\%$) of the gap of the solutions. Nevertheless, the solution times are relative small.

Table 5.8 Results on the Instances by Stützle and Fernandes with $N < 100$

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
GridRandom.974820449	50	550969	BKS	570631	3.57	57.58
GridRandom.974820450	50	400389	BKS	419420	4.75	55.13
GridRandom.974820451	50	357607	BKS	372707	4.22	55.28
GridRandom.974820452	50	246191	BKS	259915	5.57	58.95
GridRandom.974820453	50	156957	BKS	173474	10.52	51.28
GridRandom.974820454	50	93887	BKS	112891	20.24	55.14
GridRandom.974820455	50	64481	BKS	81120	25.80	1:56.72
GridRandom.974820456	50	35469	BKS	46175	30.18	53.75
GridRandom.974820457	50	11378	BKS	15802	38.88	56.38
GridRandom.974820458	50	339024	BKS	353328	4.22	1:06.47
GridRandom.974820459	50	253761	BKS	270335	6.53	53.70
GridRandom.974820460	50	220469	BKS	240132	8.92	56.48
GridRandom.974820461	50	152116	BKS	166068	9.17	1:01.66
GridRandom.974820462	50	97538	BKS	105731	8.40	53.03
GridRandom.974820463	50	52986	BKS	63133	19.15	53.08
GridRandom.974820464	50	42454	BKS	52821	24.42	59.61
GridRandom.974820465	50	15373	BKS	22518	46.48	51.30
GridRandom.974820466	50	4718	BKS	8228	74.40	54.45
GridRandom.974820467	50	285840	BKS	307068	7.43	57.14
GridRandom.974820468	50	207863	BKS	221392	6.51	1:02.77
GridRandom.974820469	50	182387	BKS	199667	9.47	1:08.41
GridRandom.974820470	50	121797	BKS	139181	14.27	58.11
GridRandom.974820471	50	79749	BKS	96057	20.45	1:00.56
GridRandom.974820472	50	39927	BKS	48481	21.42	51.72
GridRandom.974820473	50	25008	BKS	33328	33.27	58.45
GridRandom.974820474	50	16887	BKS	25758	52.53	52.19
GridRandom.974820475	50	4579	BKS	9557	108.71	52.78
GridRandom.974820476	50	151403	BKS	168663	11.40	59.75
GridRandom.974820477	50	105515	BKS	125565	19.00	50.09
GridRandom.974820478	50	86290	BKS	98282	13.90	1:00.48
GridRandom.974820479	50	62820	BKS	78251	24.56	58.36
GridRandom.974820480	50	36278	BKS	46863	29.18	53.22
GridRandom.974820481	50	23683	BKS	34381	45.17	58.16
GridRandom.974820482	50	12728	BKS	21038	65.29	1:00.50
GridRandom.974820483	50	5336	BKS	8576	60.72	57.50
GridRandom.974820484	50	2914	BKS	4800	64.72	53.86
GridStructured.974825925	50	4243	BKS	6603	55.62	1:13.94
GridStructured.974825926	50	25986	BKS	41700	60.47	58.69
GridStructured.974825927	50	91887	BKS	115187	25.36	1:15.84
GridStructured.974825928	50	303320	BKS	329910	8.77	1:08.89
GridStructured.974825929	50	1770	BKS	2302	30.06	1:13.97
GridStructured.974825930	50	16335	BKS	33001	102.03	57.92
GridStructured.974825931	50	53538	BKS	83525	56.01	1:11.88

Continued on next page

Continued from previous page

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
GridStructured.974825932	50	161448	BKS	173852	7.68	56.78
GridStructured.974825933	50	3620	BKS	6427	77.54	1:10.36
GridStructured.974825934	50	9152	BKS	16050	75.37	1:03.11
GridStructured.974825936	50	59994	BKS	80919	34.88	1:13.95
GridStructured.974825937	50	149938	BKS	167170	11.49	59.77
GridStructured.974825938	50	1914	BKS	3137	63.90	1:54.67
GridStructured.974825939	50	4747	BKS	8545	80.01	1:09.63
GridStructured.974825940	50	29106	BKS	38705	32.98	1:04.50
GridStructured.974825941	50	69820	BKS	84801	21.46	1:01.03
GridStructuredPlus.974826006	50	3493	BKS	4835	38.42	1:21.23
GridStructuredPlus.974826008	50	24313	BKS	43435	78.65	1:04.45
GridStructuredPlus.974826009	50	118283	BKS	144073	21.80	1:21.02
GridStructuredPlus.974826010	50	251804	BKS	278376	10.55	1:03.00
GridStructuredPlus.974826011	50	2312	BKS	3360	45.33	1:24.86
GridStructuredPlus.974826012	50	12939	BKS	23851	84.33	1:17.31
GridStructuredPlus.974826013	50	77515	BKS	89607	15.60	1:26.50
GridStructuredPlus.974826014	50	175559	BKS	197787	12.66	55.25
GridStructuredPlus.974826015	50	3551	BKS	5124	44.30	1:10.53
GridStructuredPlus.974826016	50	11276	BKS	21432	90.07	1:02.23
GridStructuredPlus.974826017	50	46340	BKS	67421	45.49	1:09.31
GridStructuredPlus.974826018	50	121688	BKS	147915	21.55	1:11.19
GridStructuredPlus.974826019	50	952	BKS	1886	98.11	2:01.30
GridStructuredPlus.974826020	50	5645	BKS	9396	66.45	1:09.48
GridStructuredPlus.974826021	50	29568	BKS	45305	53.22	1:00.36
GridStructuredPlus.974826022	50	65000	BKS	75289	15.83	1:00.84
RandomRandom.974820375	50	15946893	BKS	16537777	3.71	1:01.22
RandomRandom.974820376	50	10711742	BKS	11289450	5.39	54.80
RandomRandom.974820377	50	10146457	BKS	10557496	4.05	51.03
RandomRandom.974820378	50	7411328	BKS	8079555	9.02	1:26.55
RandomRandom.974820379	50	4899211	BKS	5402286	10.27	1:17.94
RandomRandom.974820380	50	3429288	BKS	3796952	10.72	1:10.36
RandomRandom.974820382	50	2402176	BKS	2741599	14.13	1:09.08
RandomRandom.974820383	50	856989	BKS	1099838	28.34	1:01.11
RandomRandom.974820384	50	213156	BKS	333450	56.43	1:04.02
RandomRandom.974820385	50	8663908	BKS	8892523	2.64	51.39
RandomRandom.974820386	50	6539530	BKS	6761932	3.40	51.36
RandomRandom.974820387	50	5951176	BKS	6293057	5.74	54.89
RandomRandom.974820388	50	4528015	BKS	5073935	12.06	2:15.39
RandomRandom.974820389	50	2380849	BKS	2620562	10.07	56.08
RandomRandom.974820390	50	1823708	BKS	2112342	15.83	54.41
RandomRandom.974820391	50	761773	BKS	902353	18.45	48.58
RandomRandom.974820392	50	365155	BKS	534099	46.27	1:31.48
RandomRandom.974820393	50	154444	BKS	306380	98.38	1:07.48
RandomRandom.974820394	50	8647476	BKS	9245593	6.92	57.98
RandomRandom.974820395	50	4173308	BKS	4487094	7.52	50.92
RandomRandom.974820396	50	4674438	BKS	5207040	11.39	1:29.47
RandomRandom.974820397	50	3923467	BKS	4418304	12.61	1:13.17
RandomRandom.974820398	50	2856016	BKS	3409958	19.40	2:05.42
RandomRandom.974820399	50	1294453	BKS	1738705	34.32	53.19
RandomRandom.974820400	50	546764	BKS	773856	41.53	56.69

Continued on next page

Continued from previous page

Problem	N	Published		Our Solution		
		ObjVal	Type	ObjVal	Gap %	CPU Time
RandomRandom.974820401	50	417346	BKS	601975	44.24	1:04.67
RandomRandom.974820402	50	72019	BKS	220232	205.80	1:49.67
RandomRandom.974820403	50	4704435	BKS	5281055	12.26	52.72
RandomRandom.974820404	50	3265590	BKS	3653103	11.87	1:00.84
RandomRandom.974820405	50	2658867	BKS	2928053	10.12	50.42
RandomRandom.974820406	50	2034005	BKS	2518458	23.82	1:00.36
RandomRandom.974820407	50	1198629	BKS	1462483	22.01	56.77
RandomRandom.974820408	50	510924	BKS	669760	31.09	56.33
RandomRandom.974820409	50	421734	BKS	797261	89.04	1:03.55
RandomRandom.974820410	50	124931	BKS	234408	87.63	53.86
RandomRandom.974820411	50	35129	BKS	72184	105.48	1:07.39
RandomStructured.974823931	50	16107	BKS	129475	703.84	1:13.22
RandomStructured.974823932	50	405846	BKS	1157676	185.25	57.33
RandomStructured.974823933	50	2917225	BKS	4216994	44.55	59.84
RandomStructured.974823934	50	8550661	BKS	9387923	9.79	1:07.31
RandomStructured.974823935	50	5427	BKS	39103	620.53	1:22.63
RandomStructured.974823936	50	186096	BKS	528067	183.76	1:15.02
RandomStructured.974823937	50	2288762	BKS	3018895	31.90	1:21.28
RandomStructured.974823938	50	4437380	BKS	4877027	9.91	1:06.27
RandomStructured.974823939	50	4088	BKS	50110	1125.78	1:21.59
RandomStructured.974823940	50	76534	BKS	312674	308.54	1:02.20
RandomStructured.974823941	50	1528696	BKS	2380591	55.73	1:49.03
RandomStructured.974823942	50	4715920	BKS	5143016	9.06	1:36.06
RandomStructured.974823943	50	5824	BKS	87875	1408.84	1:16.17
RandomStructured.974823944	50	49366	BKS	208407	322.17	1:05.61
RandomStructured.974823945	50	1077626	BKS	1650709	53.18	1:47.78
RandomStructured.974823946	50	2119306	BKS	2514942	18.67	1:33.27
RandomStructuredPlus.974824391	50	7646	BKS	159450	1985.40	1:15.73
RandomStructuredPlus.974824392	50	228623	BKS	481569	110.64	56.38
RandomStructuredPlus.974824393	50	2004066	BKS	2418744	20.69	2:48.66
RandomStructuredPlus.974824394	50	7718049	BKS	8705132	12.79	1:29.70
RandomStructuredPlus.974824395	50	4153	BKS	51694	1144.74	1:06.84
RandomStructuredPlus.974824396	50	224556	BKS	581878	159.12	1:03.30
RandomStructuredPlus.974824397	50	1594195	BKS	2210934	38.69	1:04.53
RandomStructuredPlus.974824398	50	4709065	BKS	4984611	5.85	1:01.28
RandomStructuredPlus.974824399	50	4513	BKS	54409	1105.61	1:22.77
RandomStructuredPlus.974824400	50	100341	BKS	368518	267.27	57.77
RandomStructuredPlus.974824401	50	1486604	BKS	1848590	24.35	1:00.47
RandomStructuredPlus.974824402	50	3448081	BKS	3959981	14.85	1:25.48
RandomStructuredPlus.974824403	50	1607	BKS	17823	1009.09	1:26.59
RandomStructuredPlus.974824404	50	50958	BKS	222046	335.74	1:08.22
RandomStructuredPlus.974824405	50	1124591	BKS	1362367	21.14	56.75
RandomStructuredPlus.974824406	50	1801276	BKS	2142610	18.95	1:30.89

5.4 Summary of Numerical Results

In Sections 5.2 and 5.3, we have shown the numerical results of our algorithm for the QAP on a variety of QAP test problems. For many problems, our algorithm has performed reasonably well and produced good solutions in a small amount of time. As our algorithm is designed to quickly find good solutions through exploiting the problem structures via continuous optimization techniques, there are classes of problems that are inherently difficult for our algorithm.

In our discussion of the results on the QAPLIB instances in Section 5.2, we showed that the commonly used performance measure gap can be biased towards the very few feasible solutions at the extreme of the tail of a normal distribution. Based upon the exhaustively enumerated results, we have demonstrated that our algorithm has produced good solutions for the QAPLIB instances of a size $N \leq 14$. However, the gap measures for several of these instances are fairly large. Hence, we believe that gap is not an appropriate measure to evaluate the performance of an algorithm that seeks good solutions to the QAP. This suggests that a more appropriate measure of performance should be considered. We will discuss possible further research into such a performance measure in Section 6.2.

Chapter 6

Conclusions, Further Research and Extensions

6.1 Conclusions

In this research we have developed an algorithm for the QAP based upon several continuous optimization techniques. Although the method of relaxation with a penalty function has been proposed earlier in the context of QAP or in the related study of nonlinear 0-1 programming, the pre-conditioning of the Hessian of the objective function in our algorithm is a novel application of such a technique in this context. Using the convex transformation to devise a scheme for an initial point is also unique. Furthermore, the quadratic cut is the first attempt to use a higher order cut in the context of quadratic 0-1 programming with the QAP as a special case. Finally, employing random sampling in the interior of the feasible region for different starting points is an original application of random sampling in the context of the QAP.

We have implemented our proposed algorithm for the QAP and run it on a variety of QAP test instances. The numerical results have shown that our algorithm can potentially produce good solutions in a small amount of time for certain classes of QAP instances.

6.2 Further Research and Extensions

As Figure 5.1 shows, the objective values of the feasible solutions to the QAP appear to follow a normal distribution. Hence, we suggest further research to formalize this result. If this conjecture is true, we can sample among the QAP solutions and draw inferences on the underlying normal distribution. Then based upon the estimates of the mean and variance, denoted by μ and σ^2, of the normal distribution, we can use a performance

measure as follows. In particular, for a QAP solution whose objective value is $\hat{q}$, we have

$$\mathrm{P}\left(Z < \frac{\hat{q} - \mu}{\sigma}\right) = \alpha$$

where Z denotes a standard normal variate. That is, we use α as a measure of the quality of the QAP solution.

Extension of our algorithm for the QAP to quadratic 0-1 programming is a straightforward task. In general, the strong equivalence does not hold for a quadratic 0-1 programming problem; therefore, solving the relaxation using the quadratic penalty function is not a guarantee for an integer solution. However, almost all other techniques in our algorithm can be readily applied in the context of quadratic 0-1 programming. When an integer solution cannot be obtained by solving the relaxation with the quadratic penalty function, there needs to be some other scheme to obtain an integer point to start the relaxation problems with the quadratic cut.

After we extend our algorithm to quadratic 0-1 programming, we can then think of further extending it to general nonlinear 0-1 programming. There are two possibilities:

1. Some reduction techniques such as those described in Section 2.1 can be applied to reduce a general nonlinear 0-1 programming problem to an equivalent quadratic 0-1 programming problem. Then the extension of our algorithm to quadratic 0-1 programming is applied to solve the resulting quadratic 0-1 programming problem.

2. Utilizing the idea of a sequential quadratic programming algorithm for continuous nonlinear programming, we can approximate a nonlinear 0-1 programming problem by a quadratic 0-1 programming problem. Then the extension of our algorithm to quadratic 0-1 programming is applied to solve the resulting quadratic 0-1 programming approximation. With this option, we need to address such issues that arise when the solution to the quadratic 0-1 programming approximation is infeasible to the original nonlinear 0-1 programming problem or the quadratic 0-1 programming approximation is an infeasible problem.

Appendix A

Proofs from Literature

A.1 Total Unimodularity of the Assignment Matrix

Proposition A.1 (Salkin and Mathur [104], 1989) *The assignment matrix L in (1.5) is totally unimodular.*

Proof. Consider the following assignment matrix:

$$L = \begin{pmatrix} 1 & \cdots & 1 & 1 & & & & & & & & & & & \\ & & & & 1 & \cdots & 1 & 1 & & & & & & & \\ & & & & & & & & \cdot & \cdot & \cdot & \cdot & & & \\ & & & & & & & & & & & & 1 & \cdots & 1 & 1 \\ 1 & & & & 1 & & & & \cdot & & & & 1 & & & \\ & \ddots & & & & \ddots & & & & \cdot & & & & \ddots & & \\ & & 1 & & & & 1 & & & \cdot & & & & & 1 & \end{pmatrix}.$$

Note that we have deleted the last assignment constraint. Hence matrix L has a full row rank. Since each element of L is either 0 or 1, the determinant of every 1×1 submatrix of L is 0 or 1. Now by induction, suppose the determinant of every submatrix of size $(k-1) \times (k-1)$, denoted by $\det L^{k-1}$, is 0, 1, or -1 and let L^k be any $k \times k$ submatrix of L, where $k \geq 2$. There are three cases we need to consider: i) If any column of L^k is all zeros, then $\det L^k = 0$. ii) If any column of L^k has only a single 1, expand the determinant of L^k by the cofactors of that column. We have $\det L^k = \pm \det L^{k-1}$. Hence, $\det L^k = 0$, 1, or -1. iii) If every column of L^k has two 1's, then one 1 in each column must be from the facility constraint and the other from the location constraint. In this case the sum of the rows in L^k corresponding to the facility constraints is equal to the sum of the rows corresponding to the location constraints and thus $\det L^k = 0$. ∎

A.2 Proof of Proposition 3.6

Proof. Let $S = (s_{ij}) = Q - 2\mu I$. First we show that

$$\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} s_{ij}x_i^2 = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_i^2 + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_j^2. \tag{A.1}$$

Note $\sum_{j=n+1}^{n}(\cdot) = 0$. Hence

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_i^2 = \sum_{i=1}^{n}\sum_{j=i+1}^{n} s_{ij}x_i^2.$$

In the last term of (A.1), we can reverse the order of summation, giving

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_j^2 = \sum_{j=2}^{n}\sum_{i=1}^{j-1} s_{ij}x_j^2.$$

Since S is symmetric, we have

$$\sum_{j=2}^{n}\sum_{i=1}^{j-1} s_{ij}x_j^2 = \sum_{i=1}^{n}\sum_{j=1}^{i-1} s_{ij}x_i^2.$$

Therefore, the right-hand side of (A.1) is

$$\begin{aligned}&\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_i^2 + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_j^2\\ &\qquad = \sum_{i=1}^{n}\sum_{j=i+1}^{n} s_{ij}x_i^2 + \sum_{i=1}^{n}\sum_{j=1}^{i-1} s_{ij}x_i^2 = \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} s_{ij}x_i^2.\end{aligned}$$

Now for any $x \neq 0$, we have

$$\begin{aligned}x^T Sx &= \sum_{i=1}^{n} s_{ii}x_i^2 + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}x_i x_j\\ &= \sum_{i=1}^{n}\Big(s_{ii} + \sum_{\substack{j=1\\ j\neq i}}^{n} s_{ij}\Big)x_i^2 - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} s_{ij}\big(x_i - x_j\big)^2\\ &= \sum_{i=1}^{n}\Big(-2\mu + \sum_{j=1}^{n} q_{ij}\Big)x_i^2 - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij}\big(x_i - x_j\big)^2\\ &\leq \sum_{i=1}^{n}\Big(-2\mu + \sum_{j=1}^{n} q_{ij}\Big)x_i^2 < 0\end{aligned}$$

which completes the proof. ∎

Appendix B

Test Problems and Numerical Results

B.1 QAPLIB Instances

Table B.1 lists the complete set of the QAPLIB instances along with their sizes N and the objective values of the optimal or best known solutions. In column "objval", the asterisked entries are the objective values of the best known solutions and the other entries are the objective values of the optimal solutions.

Table B.1 A Complete List of the QAPLIB Instances

Problem	N	objval	Problem	N	objval
bur26a	26	5426670	bur26b	26	3817852
bur26c	26	5426795	bur26d	26	3821225
bur26e	26	5386879	bur26f	26	3782044
bur26g	26	10117172	bur26h	26	7098658
chr12a	12	9552	chr12b	12	9742
chr12c	12	11156	chr15a	15	9896
chr15b	15	7990	chr15c	15	9504
chr18a	18	11098	chr18b	18	1534
chr20a	20	2192	chr20b	20	2298
chr20c	20	14142	chr22a	22	6156
chr22b	22	6194	chr25a	25	3796
els19	19	17212548	esc16a	16	68
esc16b	16	292	esc16c	16	160
esc16d	16	16	esc16e	16	28
esc16f	16	0	esc16g	16	26
esc16h	16	996	esc16i	16	14
esc16j	16	8	esc32a	32	130 *
esc32b	32	168 *	esc32c	32	642 *
esc32d	32	200 *	esc32e	32	2
esc32f	32	2	esc32g	32	6
esc32h	32	438 *	esc64a	64	116 *
esc128	128	64 *	had12	12	1652
had14	14	2724	had16	16	3720
had18	18	5358	had20	20	6922
kra30a	30	88900	kra30b	30	91420
kra32	32	88700	lipa20a	20	3683
lipa20b	20	27076	lipa30a	30	13178
lipa30b	30	151426	lipa40a	40	31538
lipa40b	40	476581	lipa50a	50	62093
lipa50b	50	1210244	lipa60a	60	107218

Continued on next page

Continued from previous page

Problem	N	objval	Problem	N	objval
lipa60b	60	2520135	lipa70a	70	169755
lipa70b	70	4603200	lipa80a	80	253195
lipa80b	80	7763962	lipa90a	90	360630
lipa90b	90	12490441	nug12	12	578
nug14	14	1014	nug15	15	1150
nug16a	16	1610	nug16b	16	1240
nug17	17	1732	nug18	18	1930
nug20	20	2570	nug21	21	2438
nug22	22	3596	nug24	24	3488
nug25	25	3744	nug27	27	5234
nug28	28	5166	nug30	30	6124
rou12	12	235528	rou15	15	354210
rou20	20	725522	scr12	12	31410
scr15	15	51140	scr20	20	110030
sko42	42	15812 *	sko49	49	23386 *
sko56	56	34458 *	sko64	64	48498 *
sko72	72	66256 *	sko81	81	90998 *
sko90	90	115534 *	sko100a	100	152002 *
sko100b	100	153890 *	sko100c	100	147862 *
sko100d	100	149576 *	sko100e	100	149150 *
sko100f	100	149036 *	ste36a	36	9526
ste36b	36	15852	ste36c	36	8239110
tai10a	10	135028	tai10b	10	1183760
tai12a	12	224416	tai12b	12	39464925
tai15a	15	388214	tai15b	15	51765268
tai17a	17	491812	tai20a	20	703482
tai20b	20	122455319	tai25a	25	1167256
tai25b	25	344355646	tai30a	30	1818146 *
tai30b	30	637117113 *	tai35a	35	2422002 *
tai35b	35	283315445 *	tai40a	40	3139370 *
tai40b	40	637250948 *	tai50a	50	4938796 *
tai50b	50	458821517 *	tai60a	60	7205962 *
tai60b	60	608215054 *	tai64c	64	1855928 *
tai80a	80	13515450 *	tai80b	80	818415043 *
tai100a	100	21054656 *	tai100b	100	1185996137 *
tai150b	150	498896643 *	tai256c	256	44759294 *
tho30	30	149936	tho40	40	240516 *
tho150	150	8133398 *	wil50	50	48816 *
wil100	100	273038 *			

B.2 Results of Matrix Reduction Algorithm

In Table B.2 the columns under "Original" are for the Hessian of the original objective function of the QAP and the columns under "MatRed" for the modified Hessian produced by the Matrix Reduction algorithm. The column header "sp_eigvals" stands for the

spread of eigenvalues. Since the Matrix Reduction algorithm can cause fill-in and produce denser matrices, we show the percentage of nonzero elements in the original Hessian and modified Hessian. Note that since the Matrix Reduction algorithm requires that both the flow matrix and distance matrix be symmetric, we have not run instances bur26a–bur26h—These instances are asymmetric problems in that both the flow matrix and distance matrix are asymmetric, so we cannot symmetrize the flow matrix or distance matrix as discussed in Section 1.2.2 when only one of them but not both are asymmetric.

Table B.2 Matrix Reduction Algorithm on QAPLIB Instances with $N \leq 100$

Problem	Original		MatRed	
	sp_eigvals	nonzeros %	sp_eigvals	nonzeros %
chr12a	286770.4	13.8	69022.2	100
chr12b	313910.8	13.8	66952.8	100
chr12c	235817.4	13.8	58282.1	100
chr15a	333640.1	11.6	76936.0	100
chr15b	365987.6	11.6	86874.9	100
chr15c	317854.4	11.6	75107.4	100
chr18a	406621.2	9.9	97606.7	100
chr18b	23151.1	9.9	5122.5	93.8
chr20a	57711.9	9.0	11699.4	100
chr20b	50026.1	9.0	10993.9	100
chr20c	634422.9	9.0	144979.7	100
chr22a	136795.6	8.3	27171.1	100
chr22b	128071.2	8.3	25848.5	100
chr25a	124052.6	7.4	23727.0	100
els19	634928254.8	29.4	136819917.2	100
esc16a	518.5	20.4	51.4	100
esc16b	992.8	49.4	46.9	100
esc16c	879.0	27.4	138.7	100
esc16d	241.5	11.3	48.0	100
esc16e	374.9	11.3	55.4	100
esc16f	0.0	0.0	0.0	0.0
esc16g	405.8	11.3	61.5	100
esc16h	6227.8	61.8	611.9	100
esc16i	319.0	8.1	58.8	100
esc16j	214.8	6.4	36.6	100
esc32a	1880.6	11.7	292.3	100
esc32b	1568.0	17.1	160.0	100
esc32c	4982.7	20.8	412.3	100
esc32d	2088.2	14.3	193.5	100
esc32e	1016.5	1.0	163.8	100
esc32f	1016.5	1.0	163.8	100
esc32g	778.6	1.4	119.7	100
esc32h	3148.9	22.4	263.0	100
esc64a	3462.4	2.8	377.9	100
had12	5450.0	84.0	316.1	100

Continued on next page

Continued from previous page

Problem	Original		MatRed	
	sp_eigvals	nonzeros %	sp_eigvals	nonzeros %
had14	9423.2	86.2	622.8	100
had16	12655.9	87.9	799.4	100
had18	17799.1	89.2	994.0	99.4
had20	23272.6	90.2	1301.1	100
kra30a	477544.8	35.4	44941.3	100
kra30b	485917.1	35.4	44832.7	100
kra32	519479.2	31.2	48832.8	100
lipa20a	9018.2	90.2	198.8	73.5
lipa20b	89257.7	85.0	4532.3	100
lipa30a	30856.6	93.4	446.8	84.2
lipa30b	483435.9	90.7	20401.5	100
lipa40a	73329.8	95.1	676.2	83.6
lipa40b	1478400.0	92.5	51014.8	100
lipa50a	141054.3	96.0	908.4	82.2
lipa50b	3613869.7	94.1	91994.0	100
lipa60a	241197.3	96.7	1397.0	81.8
lipa60b	7501093.4	95.4	177231.7	100
lipa70a	378559.1	97.2	1726.5	82.8
lipa70b	13699403.1	95.6	279857.9	100
lipa80a	563256.6	97.5	2051.3	80.1
lipa80b	22845506.1	96.3	428487.5	100
lipa90a	796872.0	97.8	2562.2	80.8
lipa90b	36541373.7	96.8	633114.5	100
nug12	2477.2	57.3	307.5	97.2
nug14	4278.1	64.4	496.2	100
nug15	4858.7	62.2	593.0	99.1
nug16a	6374.5	68.1	694.9	100
nug16b	5380.1	61.5	518.7	100
nug17	7010.6	65.8	766.0	100
nug18	7658.3	65.9	777.1	100
nug20	9962.2	67.0	900.8	95.0
nug21	11063.3	59.2	1401.8	99.1
nug22	17168.7	60.3	2301.0	100
nug24	14601.7	61.6	1450.5	100
nug25	14882.3	61.4	1295.3	100
nug27	22240.4	61.6	2446.7	99.5
nug28	20491.8	61.7	2017.2	99.5
nug30	24075.4	62.9	2158.2	96.9
rou12	812749.7	82.8	92167.2	100
rou15	1245417.3	86.3	132120.5	100
rou20	2353227.0	88.8	186990.6	100
scr12	353163.4	35.6	55458.9	97.2
scr15	504987.3	34.8	88811.8	100
scr20	1074731.2	29.4	181114.9	95.0
sko42	54688.2	66.7	4239.9	99.1
sko49	78533.6	66.2	5567.0	100
sko56	116132.1	66.5	7897.6	99.5
sko64	156756.3	66.6	9643.4	100
sko72	215761.9	67.8	12683.8	99.8
sko81	289918.6	68.5	15588.4	100

Continued on next page

Continued from previous page

Problem	Original		MatRed	
	sp_eigvals	nonzeros %	sp_eigvals	nonzeros %
sko90	369755.5	67.7	19116.9	99.7
sko100a	475791.6	67.9	22697.2	100
sko100b	482676.7	67.6	23041.4	100
sko100c	464872.7	66.8	23605.9	100
sko100d	468341.5	66.7	22373.5	100
sko100e	470164.5	66.6	23773.7	100
sko100f	465137.9	66.9	22454.1	100
ste36a	226235.1	25.8	45835.2	98.8
ste36b	933769.9	25.8	330459.9	99.4
ste36c	187350938.0	25.8	39477423.8	99.4
tai10a	526767.2	77.4	67962.9	100
tai10b	13531710.5	63.0	1938952.0	100
tai12a	862064.1	81.5	88684.3	100
tai12b	372653118.4	67.9	102113287.0	100
tai15a	1283257.7	84.6	120370.6	100
tai15b	3670595334.0	67.2	3656764.0	100
tai17a	1566009.8	85.4	148563.3	100
tai20a	2247575.1	87.4	184594.0	100
tai20b	1743764415.1	75.0	561909312.8	100
tai25a	3523086.3	90.3	242668.8	100
tai25b	3797033258.7	76.8	1190483331.1	100
tai30a	5345957.1	91.5	298627.7	100
tai30b	5382582768.2	76.3	1647781512.1	100
tai35a	7044932.1	92.0	349655.2	100
tai35b	1938296811.9	65.7	550496468.9	100
tai40a	9044489.3	92.9	423452.9	100
tai40b	3945470210.1	69.1	709679729.5	100
tai50a	13945390.5	94.4	613883.8	100
tai50b	2836669109.0	65.8	679096990.7	100
tai60a	19842023.1	95.0	672106.7	100
tai60b	3587653910.5	66.2	664239331.9	100
tai64c	36686416.0	4.1	7819714.8	100
tai80a	35386508.4	95.6	953084.5	100
tai80b	4149925240.0	67.2	568166623.6	100
tai100a	53794793.9	96.1	1188864.7	100
tai100b	6275787104.3	67.9	809642681.6	100
tho30	666341.6	46.6	86862.8	99.1
tho40	1046789.0	38.0	111206.4	100
wil50	157996.8	86.2	8137.4	98.7
wil100	794292.0	88.3	22451.5	100

B.3 Results of Scaling and Shift Algorithm

In Table B.3 the columns under "Original" are the results for the first setting, i.e., applying the algorithm to the Hessian of the original objective function, and the columns under "MatRed" the results for the second setting, i.e., applying the algorithm to the

modified Hessian produced by the Matrix Reduction algorithm. σ is the scaling factor as described in Section 4.3.3. $\underline{\lambda}$ and $\overline{\lambda}$ are the smallest and largest eigenvalues, respectively, of the resulting Hessian produced by the Scaling and Shift algorithm. Note that $\underline{\lambda}$ is equal to γ discussed in Section 4.3.3. Column "nz %" shows the percentage of nonzeros in the resulting Hessian. We have not run instances bur26a–bur26h in the second setting for the same reason as mentioned in Appendix B.2.

Table B.3 Scaling and Shift Algorithm on QAPLIB Instances with $N \leq 100$

Problem	Original				MatRed			
	σ	$\underline{\lambda}$	$\overline{\lambda}$	nz %	σ	$\underline{\lambda}$	$\overline{\lambda}$	nz %
bur26a	1388.3	277.7	27766.8	85.8	–	–	–	–
bur26b	1168.7	233.7	23373.4	85.8	–	–	–	–
bur26c	1259.3	251.9	25185.7	83.9	–	–	–	–
bur26d	1060.0	212.0	21200.7	83.9	–	–	–	–
bur26e	1377.4	275.5	27548.5	78.3	–	–	–	–
bur26f	1159.5	231.9	23189.6	78.3	–	–	–	–
bur26g	1919.1	383.8	38382.2	89.1	–	–	–	–
bur26h	1615.5	323.1	32309.3	89.1	–	–	–	–
chr12a	120.3	24.1	2406.9	14.5	59.0	11.8	1180.8	100
chr12b	125.9	25.2	2518.3	14.5	58.2	11.6	1163.0	100
chr12c	109.1	21.8	2182.7	14.5	54.3	10.9	1085.1	100
chr15a	129.8	26.0	2596.2	12.1	62.3	12.5	1246.7	100
chr15b	136.0	27.2	2719.1	12.1	66.2	13.2	1324.8	100
chr15c	126.7	25.3	2534.0	12.1	61.6	12.3	1231.8	100
chr18a	143.3	28.7	2866.1	10.2	70.2	14.0	1404.2	100
chr18b	34.2	6.8	683.9	10.2	16.1	3.2	321.7	93.8
chr20a	54.0	10.8	1079.8	9.3	24.3	4.9	486.2	100
chr20b	50.3	10.1	1005.3	9.3	23.6	4.7	471.3	100
chr20c	179.0	35.8	3580.0	9.3	85.6	17.1	1711.4	100
chr22a	83.1	16.6	1662.4	8.5	37.0	7.4	740.9	100
chr22b	80.4	16.1	1608.5	8.5	36.1	7.2	722.6	100
chr25a	79.2	15.8	1583.1	7.5	34.6	6.9	692.3	100
els19	5662.8	1132.6	113255.6	29.7	2628.7	525.7	52574.1	100
esc16a	5.1	1.0	102.3	20.8	1.0	1.0	52.4	100
esc16b	7.1	1.4	141.6	49.8	1.0	1.0	47.9	100
esc16c	6.7	1.3	133.3	27.8	1.4	1.0	100.0	100
esc16d	2.4	1.0	100.0	11.7	1.0	1.0	49.0	100
esc16e	3.8	1.0	100.0	11.7	1.0	1.0	56.4	100
esc16f	1.0	1.0	1.0	0.4	1.0	1.0	1.0	0.4
esc16g	4.1	1.0	100.0	11.7	1.0	1.0	62.5	100
esc16h	17.7	3.5	354.7	62.2	5.6	1.1	111.2	100
esc16i	3.2	1.0	100.0	8.4	1.0	1.0	59.8	100
esc16j	2.2	1.0	100.0	6.8	1.0	1.0	37.6	100
esc32a	9.7	1.9	194.9	11.8	3.0	1.0	100.0	100
esc32b	8.9	1.8	178.0	17.2	1.6	1.0	100.0	100
esc32c	15.9	3.2	317.3	20.9	4.2	1.0	100.0	100
esc32d	10.3	2.1	205.4	14.4	2.0	1.0	100.0	100
esc32e	7.2	1.4	143.3	1.0	1.7	1.0	100.0	100

Continued on next page

Continued from previous page

Problem	Original				MatRed			
	σ	$\underline{\lambda}$	$\overline{\lambda}$	nz %	σ	$\underline{\lambda}$	$\overline{\lambda}$	nz %
esc32f	7.2	1.4	143.3	1.0	1.7	1.0	100.0	100
esc32g	6.3	1.3	125.4	1.5	1.2	1.0	100.0	100
esc32h	12.6	2.5	252.2	22.5	2.7	1.0	100.0	100
esc64a	13.2	2.6	264.5	2.9	3.8	1.0	100.0	100
had12	16.6	3.3	331.8	84.7	3.2	1.0	100.0	100
had14	21.8	4.4	436.3	86.7	5.6	1.1	112.2	100
had16	25.3	5.1	505.6	88.3	6.4	1.3	127.1	100
had18	30.0	6.0	599.6	89.5	7.1	1.4	141.7	99.4
had20	34.3	6.9	685.7	90.5	8.1	1.6	162.1	100
kra30a	155.3	31.1	3106.0	35.6	47.6	9.5	952.8	100
kra30b	156.7	31.3	3133.1	35.6	47.6	9.5	951.7	100
kra32	162.0	32.4	3239.5	31.3	49.7	9.9	993.2	100
lipa20a	21.3	4.3	426.8	90.5	2.0	1.0	100.0	73.5
lipa20b	67.1	13.4	1342.8	85.3	15.1	3.0	302.6	100
lipa30a	39.5	7.9	789.5	93.6	4.5	1.0	100.0	84.2
lipa30b	156.3	31.3	3125.1	90.8	32.1	6.4	642.0	100
lipa40a	60.9	12.2	1217.1	95.1	5.8	1.2	116.9	83.6
lipa40b	273.3	54.7	5465.0	92.6	50.8	10.2	1015.2	100
lipa50a	84.4	16.9	1688.1	96.1	6.8	1.4	135.5	82.2
lipa50b	427.2	85.4	8544.4	94.1	68.2	13.6	1363.3	100
lipa60a	110.4	22.1	2207.4	96.7	8.4	1.7	168.0	81.8
lipa60b	615.5	123.1	12310.0	95.5	94.6	18.9	1892.2	100
lipa70a	138.3	27.7	2765.4	97.2	9.3	1.9	186.8	82.8
lipa70b	831.8	166.4	16636.0	95.6	118.9	23.8	2377.8	100
lipa80a	168.7	33.7	3373.3	97.5	10.2	2.0	203.6	80.1
lipa80b	1074.2	214.8	21483.1	96.3	147.1	29.4	2942.2	100
lipa90a	200.6	40.1	4012.3	97.8	11.4	2.3	227.5	80.8
lipa90b	1358.5	271.7	27170.0	96.8	178.8	35.8	3576.3	100
nug12	11.2	2.2	223.7	58.0	3.1	1.0	100.0	97.2
nug14	14.7	2.9	294.0	64.9	5.0	1.0	100.1	100
nug15	15.7	3.1	313.3	62.7	5.5	1.1	109.5	99.1
nug16a	17.9	3.6	358.9	68.5	5.9	1.2	118.5	100
nug16b	16.5	3.3	329.7	61.9	5.1	1.0	102.4	100
nug17	18.8	3.8	376.3	66.1	6.2	1.2	124.4	100
nug18	19.7	3.9	393.3	66.2	6.3	1.3	125.3	100
nug20	22.4	4.5	448.6	67.2	6.7	1.3	134.9	95.0
nug21	23.6	4.7	472.8	59.4	8.4	1.7	168.3	99.1
nug22	29.4	5.9	588.9	60.6	10.8	2.2	215.6	100
nug24	27.2	5.4	543.1	61.7	8.6	1.7	171.2	100
nug25	27.4	5.5	548.3	61.6	8.1	1.6	161.8	100
nug27	33.5	6.7	670.3	61.7	11.1	2.2	222.3	99.5
nug28	32.2	6.4	643.4	61.9	10.1	2.0	201.9	99.5
nug30	34.9	7.0	697.4	63.1	10.4	2.1	208.8	96.9
rou12	202.6	40.5	4052.1	83.4	68.2	13.6	1364.5	100
rou15	250.8	50.2	5016.0	86.7	81.7	16.3	1633.7	100
rou20	344.7	68.9	6894.9	89.1	97.2	19.4	1943.6	100
scr12	133.6	26.7	2671.1	36.3	52.9	10.6	1058.5	97.2
scr15	159.7	31.9	3194.0	35.3	67.0	13.4	1339.5	100
scr20	233.0	46.6	4659.6	29.7	95.6	19.1	1912.8	95.0
sko42	52.6	10.5	1051.1	66.8	14.6	2.9	292.7	99.1

Continued on next page

Continued from previous page

Problem	Original				MatRed			
	σ	$\underline{\lambda}$	$\overline{\lambda}$	nz %	σ	$\underline{\lambda}$	$\overline{\lambda}$	nz %
sko49	63.0	12.6	1259.6	66.2	16.8	3.4	335.4	100
sko56	76.6	15.3	1531.7	66.5	20.0	4.0	399.4	99.5
sko64	89.0	17.8	1779.5	66.6	22.1	4.4	441.4	100
sko72	104.4	20.9	2087.8	67.8	25.3	5.1	506.2	99.8
sko81	121.0	24.2	2420.1	68.5	28.1	5.6	561.2	100
sko90	136.7	27.3	2733.1	67.7	31.1	6.2	621.4	99.7
sko100a	155.0	31.0	3100.3	67.9	33.9	6.8	677.1	100
sko100b	156.1	31.2	3122.7	67.6	34.1	6.8	682.3	100
sko100c	153.2	30.6	3064.5	66.8	34.5	6.9	690.6	100
sko100d	153.8	30.8	3075.9	66.7	33.6	6.7	672.3	100
sko100e	154.1	30.8	3081.9	66.7	34.7	6.9	693.0	100
sko100f	153.3	30.7	3065.4	66.9	33.7	6.7	673.5	100
ste36a	106.9	21.4	2137.9	25.9	48.1	9.6	962.3	98.8
ste36b	217.2	43.4	4343.3	25.9	129.2	25.8	2583.8	99.4
ste36c	3076.1	615.2	61521.3	25.9	1412.0	282.4	28240.5	99.4
tai10a	163.1	32.6	3262.2	78.4	58.6	11.7	1171.7	100
tai10b	826.7	165.3	16533.8	64.0	312.9	62.6	6258.7	100
tai12a	208.7	41.7	4173.2	82.2	66.9	13.4	1338.5	100
tai12b	4338.3	867.7	86766.0	68.6	2271.0	454.2	45419.1	100
tai15a	254.6	50.9	5091.6	85.1	78.0	15.6	1559.4	100
tai15b	13615.6	2723.1	272311.3	67.6	429.8	85.9	8595.0	100
tai17a	281.2	56.2	5624.6	85.7	86.6	17.3	1732.4	100
tai20a	336.9	67.4	6738.4	87.7	96.6	19.3	1931.1	100
tai20b	9384.5	1876.9	187690.1	75.3	5327.2	1065.4	106544.4	100
tai25a	421.8	84.4	8436.4	90.5	110.7	22.1	2214.1	100
tai25b	13848.1	2769.6	276961.6	77.0	7754.1	1550.8	155081.2	100
tai30a	519.6	103.9	10392.3	91.6	122.8	24.6	2456.2	100
tai30b	16487.8	3297.6	329756.0	76.4	9122.6	1824.5	182451.4	100
tai35a	596.5	119.3	11929.9	92.1	132.9	26.6	2657.8	100
tai35b	9894.1	1978.8	197882.6	65.7	5272.8	1054.6	105456.8	100
tai40a	675.9	135.2	13517.3	92.9	146.2	29.2	2924.8	100
tai40b	14116.2	2823.2	282323.3	69.2	5986.9	1197.4	119737.1	100
tai50a	839.2	167.8	16784.7	94.4	176.1	35.2	3521.6	100
tai50b	11969.4	2393.9	239387.6	65.8	5856.4	1171.3	117128.7	100
tai60a	1001.1	200.2	20021.2	95.0	184.2	36.8	3684.8	100
tai60b	13460.9	2692.2	269217.1	66.2	5792.0	1158.4	115840.3	100
tai64c	1361.2	272.2	27223.9	4.1	628.4	125.7	12568.8	100
tai80a	1336.9	267.4	26737.2	95.7	219.4	43.9	4388.0	100
tai80b	14477.3	2895.5	289546.0	67.3	5356.8	1071.4	107136.0	100
tai100a	1648.3	329.7	32966.1	96.1	245.0	49.0	4900.8	100
tai100b	17803.3	3560.7	356066.8	67.9	6394.6	1278.9	127892.2	100
tho30	183.4	36.7	3669.0	46.7	66.2	13.2	1324.7	99.1
tho40	229.9	46.0	4598.6	38.1	74.9	15.0	1498.9	100
wil50	89.3	17.9	1786.6	86.2	20.3	4.1	405.5	98.7
wil100	200.3	40.1	4005.8	88.3	33.7	6.7	673.5	100

Bibliography

[1] W.P. Adams and T. Johnson. Improved Linear Programming Based Lower Bounds for the Quadratic Assignment Problem. In P.M. Pardalos and H. Wolkowicz, eds., *Quadratic Assignment and Related Problems*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16, pp. 43–77. AMS, Providence, RI, 1994.

[2] W.P. Adams and H.D. Sherali. A Tight Linearization and an Algorithm for Zero-One Quadratic Programming Problems. *Management Science*, **32**(10), 1274–1290, 1986.

[3] E. Angel and V. Zissimopoulos. On the Landscape Ruggedness of the Quadratic Assignment Problem. *Theoretical Computer Science*, **263**, 159–172, 2001.

[4] K.M. Anstreicher. Recent Advances in the Solution of Quadratic Assignment Problems. *Mathematical Programming*, **97**, 27–42, 2003.

[5] K.M. Anstreicher and N.W. Brixius. A New Bound for the Quadratic Assignment Problem Based on Convex Quadratic Programming. *Mathematical Programming*, **89**(3), 341–357, 2001.

[6] K.M. Anstreicher, N.W. Brixius, J.-P. Goux, and J. Linderoth. Solving Large Quadratic Assignment Problems on Computational Grids. *Mathematical Programming*, **91**(3), 563–588, 2002.

[7] E. Balas. An Additive Algorithm for Solving Linear Programs with Zero-One Variables. *Operations Research*, **13**(4), 517–546, 1965.

[8] E. Balas and J.B. Mazzola. Nonlinear 0-1 Programming: I. Linearization Techniques. *Mathematical Programming*, **30**, 1–21, 1984.

[9] E. Balas and J.B. Mazzola. Nonlinear 0-1 Programming: II. Dominance Relations and Algorithms. *Mathematical Programming*, **30**, 22–45, 1984.

[10] M.S. Bazaraa and O. Kirca. Branch and Bound Based Heuristics for Solving the Quadratic Assignment Problem. *Naval Research Logistics Quarterly*, **30**, 287–304, 1983.

[11] M.S. Bazaraa and H.D. Sherali. Benders' Partitioning Scheme Applied to a New Formulation of the Quadratic Assignment Problem. *Naval Research Logistics Quarterly*, **27**, 29–41, 1980.

[12] M.S. Bazaraa and H.D. Sherali. On the Use of Exact and Heuristic Cutting Plane Methods for the Quadratic Assignment Problem. *Journal of the Operational Research Society*, **33**(11), 991–1003, 1982.

[13] S.H. Bokhari. *Assignment Problems in Parallel and Distributed Computing*. Kluwer Academic Publisher, Boston, 1987.

[14] M. Borchardt. An Exact Penalty Approach for Solving a Class of Minimization Problems with Boolean Variables. *Optimization*, **19**(6), 829–838, 1988.

[15] E. Boros and P.L. Hammer. Pseudo-Boolean Optimization. *Discrete Applied Mathematics*, **123**, 155–225, 2002.

[16] N.W. Brixius and K.M. Anstreicher. Solving Quadratic Assignment Problems Using Convex Quadratic Programming Relaxations. *Optimization Methods and Software*, **16**, 49–68, 2001.

[17] R.E. Burkard. Locations with Spatial Interactions: The Quadratic Assignment Problem. In P.B. Mirchandani and R.L. Francis, eds., *Discrete Location Theory*, pp. 387–437. Wiley, New York, 1991.

[18] R.E. Burkard and E. Çela. Quadratic and Three-Dimensional Assignments: An Annotated Bibliography. In M. Dell'Amico, F. Maffioli, and S. Martello, eds., *Annotated Bibliographies in Combinatorial Optimization*, pp. 373–391. Wiley, Chichester, 1997.

[19] R.E. Burkard, E. Çela, P.M. Pardalos, and L.S. Pitsoulis. The Quadratic Assignment Problem. In D.-Z. Du and P.M. Pardalos, eds., *Handbook of Combinatorial Optimization*, vol. 3, pp. 241–337. Kluwer Academic Publishers, Dordrecht, 1998.

[20] R.E. Burkard and U. Fincke. The Asymptotic Probabilistic Behavior of the Quadratic Sum Assignment Problem. *Zeitschrift für Operations Research*, **27**, 73–81, 1983.

[21] R.E. Burkard and U. Fincke. Probabilistic Asymptotic Properties of Some Combinatorial Optimization Problems. *Discrete Applied Mathematics*, **12**, 21–29, 1985.

[22] R.E. Burkard, S.E. Karisch, and F. Rendl. QAPLIB—A Quadratic Assignment Problem Library. *Journal of Global Optimization*, **10**(4), 391–403, 1997.

[23] R.H. Byrd, M.E. Hribar, and J. Nocedal. An Interior Point Algorithm for Large-Scale Nonlinear Programming. *SIAM Journal on Optimization*, **9**(4), 877–900, 1999.

[24] P. Carraresi and F. Malucelli. A New Lower Bound for the Quadratic Assignment Problem. *Operations Research*, **40**(1), S22–S27, 1992.

[25] P. Carraresi and F. Malucelli. A Reformulation Scheme and New Lower Bounds for the QAP. In P.M. Pardalos and H. Wolkowicz, eds., *Quadratic Assignment and Related Problems*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16, pp. 147–160. AMS, Providence, RI, 1994.

[26] M.W. Carter. The Indefinite Zero-One Quadratic Problem. *Discrete Applied Mathematics*, **7**, 23–44, 1984.

[27] E. Çela. *The Quadratic Assignment Problem: Theory and Algorithms.* Kluwer Academic Publishers, Dordrecht, 1998.

[28] E.G. Davydov and I.Kh. Sigal. Application of the Penalty Function Method in Integer Programming Problems. *Engineering Cybernetics*, **10**, 21–24, 1972.

[29] A. De Maio and C. Roveda. On "An Extension of Lawler and Bell's Method of Discrete Optimization" by Mao and Wallingford. *Management Science*, **15**(10), 1969.

[30] Z. Drezner, P.M. Hahn, and É.D. Taillard. Recent Advances for the Quadratic Assignment Problem with Special Emphasis on Instances that are Difficult for Meta-Heuristic Methods. *Annals of Operations Research*, **139**(1), 65–94, 2005.

[31] S. Elloumi, A. Faye, and E. Soutif. Decomposition and Linearization for 0-1 Quadratic Programming. *Annals of Operations Research*, **99**, 79–93, 2000.

[32] A.N. Elshafei. Hospital Layout as a Quadratic Assignment Problem. *Operational Research Quarterly*, **28**, 167–179, 1977.

[33] G. Finke, R.E. Burkard, and F. Rendl. Quadratic Assignment Problems. *Annals of Discrete Mathematics*, **31**, 61–82, 1987.

[34] G.S. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*, pp. 234–235. Springer-Verlag, New York, 1996.

[35] R. Fortet. L'algèbre de Boole et ses Applications en Recherche Opérationnelle. *Cahiers du Centre d'Études de Recherche Opérationnelle*, **1**, 5–36, 1959.

[36] M. Frank and P. Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, **3**, 95–110, 1956.

[37] J.B.G. Frenk, M. van Houweninge, and A.H.G. Rinnooy Kan. Asymptotic Properties of the Quadratic Assignment Problem. *Mathematics of Operations Research*, **10**, 100–116, 1985.

[38] A.M. Frieze and J. Yadegar. On the Quadratic Assignment Problem. *Discrete Applied Mathematics*, **5**, 89–98, 1983.

[39] J.W. Gavett and N.V. Plyter. The Optimal Assignment of Facilities to Locations by Branch and Bound. *Operations Research*, **14**, 210–232, 1966.

[40] A.M. Geoffrion and G.W. Graves. Scheduling Parallel Production Lines with Changeover Costs: Practical Applications of a Quadratic Assignment/LP Approach. *Operations Research*, **24**(4), 595–610, 1976.

[41] F. Giannessi and F. Niccolucci. Connections between Nonlinear and Integer Programming Problems. In *Symposia Mathematica*, vol. XIX, pp. 161–176. Academic Press, London, 1976.

[42] F. Giannessi and F. Tardella. Connections between Nonlinear Programming and Discrete Optimization. In D.-Z. Du and P.M. Pardalos, eds., *Handbook of Combinatorial Optimization*, vol. 1, pp. 149–188. Kluwer Academic Publishers, Dordrecht, 1998.

[43] P.E. Gill and W. Murray. Newton-Type Methods for Unconstrained and Linearly Constrained Optimization. *Mathematical Programming*, **7**, 311–350, 1974.

[44] P.C. Gilmore. Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem. *SIAM Journal on Applied Mathematics*, **10**, 305–313, 1962.

[45] F. Glover. Improved Linear Integer Programming Formulations of Nonlinear Integer Problems. *Management Science*, **22**(4), 455–460, 1975.

[46] F. Glover and E. Woolsey. Further Reduction of Zero-One Polynomial Programming Problems to Zero-One Linear Programming Problems. *Operations Research*, **21**(1), 156–161, 1973.

[47] F. Glover and E. Woolsey. Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program. *Operations Research*, **22**(1), 180–182, 1974.

[48] Alexander Graham. *Kronecker Products and Matrix Calculus: with Applications*. Ellis Horwood, Chichester, 1981.

[49] D. Granot and F. Granot. Generalized Covering Relaxation in 0-1 Programming. *Operations Research*, **28**(6), 1442–1450, 1980.

[50] F. Granot and P.L. Hammer. On the Role of Generalized Covering Problems. *Cahiers du Centre d'Études de Recherche Opérationnelle*, **17**, 277–289, 1975.

[51] M. Grötschel. Discrete Mathematics in Manufacturing. In Robert E. O'Malley, ed., *Proceedings of the Second International Conference on Industrial and Applied Mathematics*, pp. 119–145. SIAM, 1992.

[52] S.W. Hadley, F. Rendl, and H. Wolkowicz. Bounds for the Quadratic Assignment Problems Using Continuous Optimization Techniques. In *Integer Programming and Combinatorial Optimization*, pp. 237–248. University of Waterloo Press, Waterloo, Ontario, Canada, 1990.

[53] S.W. Hadley, F. Rendl, and H. Wolkowicz. A New Lower Bound via Projection for the Quadratic Assignment Problem. *Mathematics of Operations Research*, **17**, 727–739, 1992.

[54] P. Hahn and T. Grant. Lower Bounds for the Quadratic Assignment Problem Based upon a Dual Formulation. *Operations Research*, **46**(6), 912–922, 1998.

[55] P. Hahn, T. Grant, and N. Hall. Solution of the Quadratic Assignment Problem Using the Hungarian Method. *European Journal of Operational Research*. (to appear).

[56] P.L. Hammer, I. Rosenberg, and S. Rudeanu. On the Determination of the Minima of Pseudo-Boolean Functions (in Romanian). *Studii si Cercetari Matematice*, **14**, 359–364, 1963.

[57] P.L. Hammer and A.A. Rubin. Some Remarks on Quadratic Programming with 0-1 Variables. *Revue Française d'Informatique et de Recherche Opérationnelle*, **3**, 67–79, 1970.

[58] P.L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer-Verlag, Berlin, 1968.

[59] P. Hansen. Methods of Nonlinear 0-1 Programming. *Annals of Discrete Mathematics*, **5**, 53–70, 1979.

[60] P. Hansen, B. Jaumard, and V. Mathon. Constrained Nonlinear 0-1 Programming. *ORSA Journal on Computing*, **5**(2), 87–119, 1993.

[61] D.R. Heffley. Assigning Runners to a Relay Team. In S.P. Ladany and R.E. Machol, eds., *Optimal Strategies in Sports*, pp. 169–171. North-Holland, Amsterdam, 1977.

[62] W.M. Hirsch and A.J. Hoffman. Extreme Varieties, Concave Functions and the Fixed Charge Problem. *Communications on Pure and Applied Mathematics*, **14**, 355–369, 1961.

[63] L.J. Hubert. *Assignment Methods in Combinatorial Data Analysis*. Marcel Dekker, Inc., New York, 1987.

[64] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, pp. 92–93. SIAM, Philadelphia, PA, 1996.

[65] V. Kaibel. *Polyhedral Combinatorics of the Quadratic Assignment Problem*. Ph.D. thesis, Universität zu Köln, Germany, 1997.

[66] B. Kalantari and J.B. Rosen. Penalty for Zero-One Integer Equivalent Problem. *Mathematical Programming*, **24**, 229–232, 1982.

[67] B. Kalantari and J.B. Rosen. Penalty Formulation for Zero-One Nonlinear Programming. *Discrete Applied Mathematics*, **16**, 179–182, 1987.

[68] S.E. Karisch. *Nonlinear Approaches for the Quadratic Assignment and Graph Partition Problems*. Ph.D. thesis, Technical University Graz, Austria, 1995.

[69] L. Kaufmann and F. Broeckx. An Algorithm for the Quadratic Assignment Problem Using Benders' Decomposition. *European Journal of Operational Research*, **2**, 204–211, 1978.

[70] T.C. Koopmans and M. Beckmann. Assignment Problems and the Location of Economic Activities. *Econometrica*, **25**(1), 53–76, 1957.

[71] J. Krarup and P.M. Pruzan. Computer-Aided Layout Design. *Mathematical Programming Study*, **9**, 75–94, 1978.

[72] A.M. Land. A Problem of Assignment with Interrelated Costs. *Operations Research Quarterly*, **14**, 185–198, 1963.

[73] D.J. Laughhunn. Quadratic Binary Programming with Application to Capital-Budgeting Problems. *Operations Research*, **18**(3), 454–461, 1970.

[74] E.L. Lawler. The Quadratic Assignment Problem. *Management Science*, **9**, 586–599, 1963.

[75] E.L. Lawler and M.D. Bell. A Method for Solving Discrete Optimization Problems. *Operations Research*, **14**(6), 1098–1112, 1966.

[76] Han-Lin Li. An Approximate Method for Local Optima for Nonlinear Mixed Integer Programming Problems. *Computers and Operations Research*, **19**(5), 435–444, 1992.

[77] Y. Li, P.M. Pardalos, K.G. Ramakrishnan, and M. Resende. Lower Bounds for the Quadratic Assignment Problem. *Annals of Operations Research*, **50**, 387–410, 1994.

[78] J.C.T. Mao and B.A. Wallingford. An Extension of Lawler and Bell's Method of Discrete Optimization with Examples from Capital Budgeting. *Management Science*, **15**(2), B51–B60, 1968.

[79] T. Mautor and C. Roucairol. An Exact Algorithm for the Solution of Quadratic Assignment Problems. *Discrete Applied Mathematics*, **55**, 281–293, 1992.

[80] R.D. McBride and J.S. Yormark. An Implicit Enumeration Algorithm for Quadratic Integer Programming. *Management Science*, **26**(3), 282–296, 1980.

[81] E.J. McCormick. *Human Factors Engineering.* McGraw-Hill, New York, 1970.

[82] G.G.L Meyer. Accelerated Frank-Wolfe Algorithms. *SIAM Journal on Control*, **12**, 655–663, 1974.

[83] P. Michelon and N. Maculan. Lagrangian Decomposition for Integer Nolinear Programming with Linear Constraints. *Mathematical Programming*, **52**, 303–313, 1991.

[84] P. Michelon and N. Maculan. Lagrangian Methods for 0-1 Quadratic Problems. *Discrete Applied Mathematics*, **42**, 257–269, 1993.

[85] P.B. Mirchandani and T. Obata. Locational Decisions with Interactions between Facilities: the Quadratic Assignment Problem a Review. Working Paper Ps-79-1, Rensselaer Polytechnic Institute, Troy, New York, May 1979.

[86] L. Mirsky. The Spread of a Matrix. *Mathematika*, **3**, 127–130, 1956.

[87] Kien-Ming Ng. *A Continuation Approach for Solving Nonlinear Optimization Problems with Discrete Variables*. Ph.D. thesis, Stanford University, June 2002.

[88] C.E. Nugent, T.E. Vollman, and J. Ruml. An Experimental Comparison of Techniques for the Assignment of Facilities to Locations. *Operations Research*, **16**, 150–173, 1968.

[89] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*, pp. 421–423. Academic Press, New York, 1970.

[90] M.W. Padberg and M.P. Rijal. *Location, Scheduling, Design and Integer Programming*. Kluwer Academic Publishers, Boston, 1996.

[91] G. Palubeckis. An Algorithm for Construction of Test Cases for the Quadratic Assignment Problem. *Informatica*, **11**(3), 281–296, 2000.

[92] P.M. Pardalos, F. Rendl, and H. Wolkowicz. The Quadratic Assignment Problem: A Survey and Recent Developments. In P.M. Pardalos and H. Wolkowicz, eds., *Quadratic Assignment and Related Problems*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16, pp. 1–42. AMS, Providence, RI, 1994.

[93] P.M. Pardalos and G.P. Rodgers. Computational Aspects of a Branch-and-Bound Algorithm for Quadratic Zero-One Programming. *Computing*, **45**, 131–144, 1990.

[94] P.M. Pardalos and J.B. Rosen. *Constrained Global Optimization: Algorithms and Applications*, *Lecture Notes in Computer Science*, vol. 268, pp. 26–27. Springer-Verlag, Berlin, Germany, 1987.

[95] S. Poljak, F. Rendl, and H. Wolkowicz. A Recipe for Semidefinite Relaxation for $(0,1)$-Quadratic Programming. *Journal of Global Optimization*, **7**(1), 51–73, 1995.

[96] M.A. Pollatschek, N. Gershoni, and Y.T. Radday. Optimization of the Typewriter Keyboard by Computer Simulation. *Angewandte Informatik*, **10**, 438–439, 1976.

[97] M. Raghavachari. On Connections between Zero-One Integer Programming and Concave Programming under Linear Constraints. *Operations Research*, **17**, 680–684, 1969.

[98] F. Rendl and H. Wolkowicz. Applications of Parametric Programming and Eigenvalue Maximization to the Quadratic Assignment Problem. *Mathematical Programming*, **53**(1), 63–78, 1992.

[99] W.T. Rhee. A Note on Asymptotic Properties of the Quadratic Assignment Problem. *Operations Research Letters*, **7**, 197–200, 1988.

[100] W.T. Rhee. Stochastic Analysis of the Quadratic Assignment Problems. *Mathematics of Operations Research*, **16**, 223–239, 1991.

[101] I.G. Rosenberg. Reduction of Bivalent Maximization to the Quadratic Case. *Cahiers du Centre d'Études de Recherche Opérationnelle*, **17**, 71–74, 1975.

[102] Reuven Y. Rubinstein. *Monte Carlo Optimization, Simulation and Sensitivity of Queueing Networks*. Wiley, New York, 1986.

[103] S. Sahni and T. Gonzalez. P-Complete Approximation Problems. *Journal of the Association for Computing Machinery*, **23**(3), 555–565, 1976.

[104] H.M. Salkin and K. Mathur. *Foundations of Integer Programming*, pp. 74–76. North-Holland, New York, 1989.

[105] Robert L. Smith. Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed over Bounded Regions. *Operations Research*, **32**(6), 1296–1308, 1984.

[106] L. Steinberg. The Backboard Wiring Problem: A Placement Algorithm. *SIAM Review*, **3**(1), 37–50, 1961.

[107] Gilbert Strang. *Linear Algebra and Its Applications*, 3rd ed., pp. 334–337, 364. Saunders College Publishing, Fort Worth, 1988.

[108] T. Stützle and S. Fernandes. New Benchmark Instances for the QAP and the Experimental Analysis of Algorithms. In J. Gottlieb and G.R. Raidl, eds., *Evolutionary Computation in Combinatorial Optimization, 4th European Conference, EvoCOP 2004*, *Lecture Notes in Computer Science*, vol. 3004, pp. 199–209. Springer-Verlag, Berlin, Germany, 2004.

[109] W. Szpankowski. Combinatorial Optimization Problems for Which Almost Every Algorithm Is Asymptotically Optimal! *Optimization*, **33**, 359–367, 1995.

[110] E.D. Taillard. Robust Tabu Search for the Quadratic Assingnment Problem. *Parallel Computing*, **17**, 443–455, 1991.

[111] E.D. Taillard. Comparison of Iterative Searches for the Quadratic Assingnment Problem. *Location Science*, **3**, 87–105, 1995.

[112] H. Tuy. Concave Programming under Linear Constraints. *Soviet Mathematics*, **5**, 1437–1440, 1964.

[113] R.J. Vanderbei and D.F. Shanno. An Interior-Point Algorithm for Nonconvex Nonlinear Programming. *Computational Optimization and Applications*, **13**, 231–252, 1999.

[114] Andreas Wächter and Lorenz T. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, **106**(1), 25–57, 2006.

[115] L.J. Watters. Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems. *Operations Research*, **15**, 1171–1174, 1967.

[116] D.J. White. A Convex Form of the Quadratic Assignment Problem. *European Journal of Operational Research*, **65**, 407–416, 1993.

[117] Q. Zhao. *Semidefinite Programming for Assignment and Partitioning Problems*. Ph.D. thesis, University of Waterloo, Ontario, Canada, 1996.

[118] Q. Zhao, S.E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite Relaxations for the Quadratic Assignment Problem. *Journal of Combinatorial Optimization*, **2**, 71–109, 1998.

[119] W.X. Zhu. Penalty Parameter for Linearly Constrained 0-1 Quadratic Programming. *Journal of Optimization Theory and Applications*, **116**(1), 229–239, 2003.