QUADRATIC ASSIGNMENT PROBLEM: LINEARIZATIONS AND POLYNOMIAL TIME SOLVABLE CASES

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Güneş Erdoğan
October, 2006

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Prof. Barbaros Tansel (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Prof. Cevdet Aykanat

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Prof. Erhan Erkut

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Assoc. Prof. Levent Kandiller

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

_____

Assist. Prof. Hande Yaman

Approved for the Institute of Engineering and Science:

_____

Prof. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

QUADRATIC ASSIGNMENT PROBLEM: LINEARIZATIONS AND POLYNOMIAL TIME SOLVABLE CASES

Güneş Erdoğan

Ph.D. in Industrial Engineering

Supervisor: Prof. Barbaros Tansel

October 2006

The Quadratic Assignment Problem (QAP) is one of the hardest combinatorial optimization problems known. Exact solution attempts proposed for instances of size larger than 15 have been generally unsuccessful even though successful implementations have been reported on some test problems from the QAPLIB up to size 36. In this dissertation, we analyze the binary structure of the QAP and present new IP formulations. We focus on "flow-based" formulations, strengthen the formulations with valid inequalities, and report computational experience with a branch-and-cut algorithm. Next, we present new classes of instances of the QAP that can be completely or partially reduced to the Linear Assignment Problem and give procedures to check whether or not an instance is an element of one of these classes. We also identify classes of instances of the Koopmans-Beckmann form of the QAP that are solvable in polynomial time. Lastly, we present a strong lower bound based on Bender's decomposition.

Keywords: Quadratic Assignment Problem, Linearization, Computational Complexity, Polynomial Time Solvability

ÖZET

KARESEL ATAMA PROBLEMİ: DOĞRUSALLAŞTIRMALAR VE POLİNOM ZAMANDA ÇÖZÜLEBİLİR DURUMLAR

Güneş Erdoğan
Endüstri Mühendisliği Bölümü Doktora
Tez Yöneticisi: Prof. Barbaros Tansel
Ekim 2006

Karesel Atama Problemi (KAP) bilinen en zor kombinatoryal eniyileme problemlerinden biridir. QAPLIB'deki boyutu 36'yı bulan bazı test problemlerinde başarılı çözümler elde edilmiş olsa da, tam çözüm yöntemleri boyutu 15'i geçen problemlerde genel olarak başarısız olmuştur. Bu tezde, KAP'ın ikili yapısını inceleyip yeni tamsayı programlar sunmaktayız. "Akış-tabanlı" formülasyonlara odaklanıp, bunları geçerli eşitsizliklerle kuvvetlendirip, dallan-ve-kes algoritması ile edindiğimiz hesapsal tecrübeyi sunmaktayız. Devamla, KAP'ın Doğrusal Atama Problemine tamamen veya kısmen indirgenebilen özel hallerini sunmakta ve verilen bir problemin bu sınıfların bir elemanı olup olmadığını kontrol eden prosedürler vermekteyiz. Ayrıca KAP'ın Koopmans-Beckmann formuülasyonunun polinom zamanda çözülebilir sınıflarını ortaya çıkartmaktayız. Son olarak, Bender ayrışımına dayanan kuvvetli bir alt sınır sunmaktayız.

Anahtar Kelimeler: Karesel Atama Problemi, Doğrusallaştırma, Hesaplama Zorluğu, Polinom Zamanlı Çözülebilirlik

To the light,

ACKNOWLEDGEMENTS

I thank my advisor Prof. Barbaros Tansel for his guidance, expertise, and patience throughout this dissertation research. With his support, this study has been an invaluable learning experience for me. It has really been an honor to work with this consummate professional.

I am indebted to members of my dissertation committee, Prof. Cevdet Aykanat, Prof. Erhan Erkut, Assoc. Prof. Levent Kandiller, and Assist. Prof. Hande Yaman, for showing keen interest in the subject matter and for accepting to read and review this thesis. Their remarks and recommendations have been invaluable.

It may be unusual to thank an institution; nevertheless I feel the need to thank Bilkent University for providing such an excellent environment for self-development. After 4 years of B.S., 2 years of M.S., and 5 years of Ph.D. study, I ask myself: If there was a higher degree of education, would I pursue it at Bilkent University? The answer is yes, without hesitation.

I would like to express my thanks to my colleagues at Tepe Teknoloji Inc., Babur Baturay, Fatih Canpolat, Menderes Fatih Güven, Nejat Serpen, Osman Tufan Doğan, and Veli Biçer for their help and support during the last phase of my graduate study.

I also would like to express my special thanks to my friends Başar Uncu, Bedrettin – Zeynep Duran, Burç Uzman, Burkay Genç, Emre Can Sezer, İlker – Esra Yağlıdere, Kağan Menekşe, Kamer Kaya, Murat Güler, Özgür Kutluözen, Selim Akgül, Serkan Bayraktar, Sibel Alumur, and Tayfun Küçükyılmaz for their help and morale support.

Finally, I would like to express my deepest gratitude to all members of my family, especially my mother Şenol Erdoğan, for their love, understanding, and patience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

The Quadratic Assignment Problem (QAP) was introduced by Koopmans and Beckmann in 1957 as a mathematical model for the location of a set of indivisible economic activities. The decision to be made is a one-to-one assignment of n facilities to n locations, which is exactly the same as the Linear Assignment Problem (LAP) except for the objective function. The term "quadratic" describes the cost function, which is the sum of the products of distances between locations and the amounts of flows between the facilities assigned to the locations.

Generating a feasible solution for the QAP is a trivial task. Let $\boldsymbol{a} = (a(1), a(2), \ldots, a(n))$ be a permutation of the integers $\{1,...,n\}$ with $a(i)$ denoting the index of the location to which facility i is assigned. Any such vector $\boldsymbol{a}$ is a feasible solution to the QAP. Similarly, devising a heuristic for the QAP is not a major task. A greedy k-exchange algorithm that starts with a random assignment is a valid (and surprisingly high quality) heuristic for the QAP. On the contrary, proving computationally the optimality of a given solution is next to impossible for large instances of the QAP. It has been shown that the QAP is NP-Hard in the strong sense (Sahni and Gonzales, 1976).

Despite 49 years of academic effort, from its initial formulation in 1957 to date, it remains as yet one of the hardest combinatorial optimization problems. Even though faster computers, specialized data structures, and algorithmic improvements have led to significant progress in solvable sizes of many NP-Hard problems (e.g. the Traveling Salesman, Vehicle Routing, Set Covering,

Uncapacitated Facility Location, etc.), the QAP has been defiantly resisting all solution attempts beyond the size of $n > 15$ when the cost data is arbitrary. The largest solved instance of the QAP to date is of size 36 (Nyström, 1999; Brixius and Anstreicher, 2001) while the largest solved size of, for example, the Traveling Salesman Problem has close to 25000 cities (Applegate et al., 2001).

For a better understanding of the current computational status of the QAP, we now give a historical sketch of the computational progress. A collection of instances and respective solutions, QAPLIB (Burkard, Karisch, and Rendl, 1997), is available online to benchmark efficiency of solution methods for the QAP. Although many different classes of instances exist in the QAPLIB, the computational improvement for the QAP may best be explained by the progress in solving the notoriously difficult instances of Nugent, Vollmann, and Ruml (1968). These are the most used instances for testing computational efficiency. The original set consists of 8 instances of sizes 5, 6, 7, 8, 12, 15, 20, and 30. Distance matrices for sizes 5 and 7 represent almost rectangular grid graphs. For sizes 6, 8, 12, 15, 20, and 30, the distance matrix represents grids of 2*3, 2*4, 3*4, 3*5, 4*5, and 5*6, respectively. Later, instances of sizes 14, 16, 17, 18, 21, 22, 24, and 25 were added to the original set by Clausen and Perregaard (1997) by deleting certain rows and columns of flow and distance matrices of larger instances. Likewise, Anstreicher et al. (2002) constructed instances of sizes 27 and 28 in the same way.

Nugent, Vollmann, and Ruml (1968) solved instances `nug05`, `nug06`, `nug07`, and `nug08` to optimality using complete enumeration. Burkard and Stratmann (1978) solved `nug12` and Burkard and Derigs (1983) solved `nug15`. Clausen and Perregaard were able to solve instances up to size 20 for the first time in 1994. Their results were published in 1997. Bruengger et al. were the first ones to solve `nug21` and `nug22` in 1996. In the same year, Clausen et al. reportedly solved `nug24`. Marzetta and Brüngger managed to solve `nug25` in

1999. Finally, Anstreicher et al. (2002) were able to solve `nug27`, `nug28`, and `nug30` to optimality in the year 2002. The progress is summarized in Figure 1.

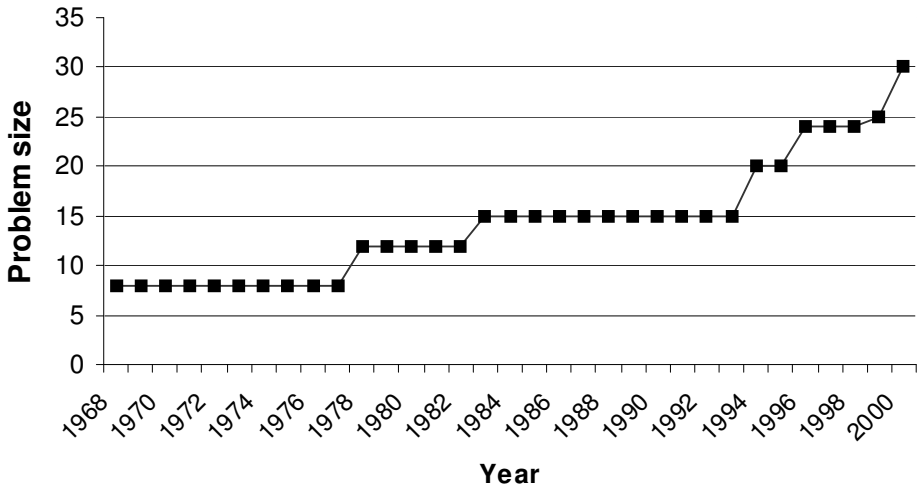

Figure 1: Computational Progress for Instances of Nugent, Vollmann, and Ruml

This set of instances is not fully representative of the overall computational state of the art for the QAP. As of this writing, the largest instances reportedly solved are `ste36a`, `ste36b`, and `ste36c` that are of size 36. These instances were proposed by Steinberg in 1961. Solving `ste36a` required 180 hours on a PIII 800 Mhz PC, while `ste36b` and `ste36c` took approximately 60 days and 200 days of CPU time, respectively. However, instances proposed by Burkard and Offermann in 1977 of size 26 have remained unsolved until recently (March 2004), at which time they were solved by the method of Hahn et al. (2001). There are still instances of size 30 waiting to be solved in the QAPLIB. We emphasize the fact that most successful applications are parallel implementations that rely on high amounts of computing power.

The computational status of the QAP poses a challenge: What new perspectives do we need to solve larger sizes of the QAP without having to rely on the high computing power of parallel processing? In this dissertation we pick up the challenge and devise an exact solution technique for the QAP that can solve large instances in a reasonable amount of computing time. Our search for such techniques has led us to identify instances which can be solved in polynomial time, which we also present.

We start by giving the formal definition of the QAP.

1. 1 Problem Definition

Although a brief description of the problem was given in the beginning of the chapter, we believe that the QAP can best be expressed in terms of compact formulations. The original formulation of the QAP by Koopmans and Beckmann (1957), where the decision variable x_{ij} is defined to be equal to 1 if facility i is assigned to location j and 0 otherwise, follows:

$$\min \sum_{i,j,k,l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl} + \sum_{i,j=1}^{n} c_{ij} x_{ij} \tag{1}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i = 1,...,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j = 1,...,n \tag{3}$$

$$x_{ij} \in \{0,1\}, \forall i, j = 1,...,n \tag{4}$$

where f_{ik} denotes the amount of flow between facilities i and k, d_{jl} denotes the distance between locations j and l, and c_{ij} denotes the cost of locating facility i at

location *j*. The linear cost coefficients may be added to certain quadratic cost coefficients to yield a pure quadratic problem and were ignored in later studies.

Lawler (1963) studied the case of generalized cost coefficients where a four dimensional matrix that defines all the costs is the input data instead of two *n* by *n* coefficient matrices. The following is the formulation for the Lawler QAP:

$$\min \sum_{i,j,k,l=1}^{n} C_{ijkl} x_{ij} x_{kl} \tag{5}$$

s.t.

(2), (3), and (4)

where C_{ijkl} denotes the cost incurred when facility *i* is located at *j* and facility *k* is located at *l* simultaneously.

A third formulation by Edwards (1977), also known as the trace formulation and is useful for certain derivations, is as follows:

$$\min_{X \in \Pi} \; tr(FXD^T X^T) \tag{6}$$

where *F* is the *n* by *n* flow matrix, *D* is the *n* by *n* distance matrix, *X* is an *n* by *n* permutation matrix, $\prod$ represents the set of *n* by *n* permutation matrices, and $tr : R^{n^2} \rightarrow R$ is the trace operator that returns the sum of the diagonal elements of a square matrix.

A fourth formulation known as the Kronecker product formulation (Lawler, 1963) is as follows. Kronecker product of two matrices $A \in R^{mn}$ and $B \in R^{pq}$ is defined by

$$A \otimes B := \begin{bmatrix} a_{11}B & a_{12}B & . & a_{1n}B \\ . & . & . & . \\ . & . & . & . \\ a_{m1}B & a_{m1}B & . & a_{mn}B \end{bmatrix} \tag{7}$$

and the operation $\langle , \rangle$ for two matrices $A \in R^{mn}$ and $B \in R^{mn}$ is defined by

$$\langle A,B \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij} \tag{8}$$

The Kronecker product formulation for the QAP is:

$$min \ \langle C,Y \rangle \tag{9}$$

s.t.

$$Y = X \otimes X \tag{10}$$

$$X \in \Pi \tag{11}$$

where $C = [C_{ijkl}]$ is the four dimensional generalized cost coefficient matrix, and X and $\prod$ are as defined above.

The fifth and final formulation, referred to as the combinatorial formulation, is as follows: Let $\boldsymbol{a} = (a(1), a(2), \ldots, a(n))$ be a permutation of the integers $\{1,...,n\}$ with $a(i)$ denoting the index of the location to which facility i is assigned. Define A to be the set of all such permutations. The combinatorial form of the Koopmans-Beckmann QAP is defined as:

$$\min_{a \in A} \sum_{i,j} f_{ij} d_{a(i)a(j)} \tag{12}$$

while the formulation for the general cost coefficients becomes

$$\min_{a \in A} \sum_{i,j} C_{ia(i)ja(j)} \qquad (13)$$

In the next section, we provide a brief literature review.

1. 2 Literature Review

In this section we briefly go over the studies in the literature that deal with exact solution techniques or identify polynomially solvable cases. For a more complete exposition to the literature on the QAP, we refer the reader to the following surveys:

Pardalos, Rendl, and Wolkowicz (1994) gave an extensive survey about the developments in methods and applications regarding the QAP. They presented various formulations, respective representations of the feasible set of solutions, theoretical and practical applications, discussions about computational complexity issues, and a survey of numerical methods for the QAP. Burkard et al. (1997) presented a survey that focus on the polynomially solvable cases that have been identified. They tried to draw a line between the NP-Hard and polynomially solvable cases of the QAP. They analyzed coefficient matrices with special properties (sum, product, Monge, Anti-Monge, Kalmanson, Toeplitz, and circulant matrices) and gave computational complexity results for many of the resulting cases and posed questions for open cases. Burkard et al. (1998) gave an extensive survey about the QAP. They provided various formulations, polytope analysis of formulations, lower bounding techniques, exact solution methods, instances that can be solved in polynomial time, and the studies about the asymptotic behavior of QAP. Çela (1998) published a book named “The Quadratic Assignment Problem, Theory and Algorithms” covering many topics about the QAP.

One of the main tracks of research on the exact solution techniques for the QAP has been Mixed Integer Programming (MIP) formulations. Since the QAP has originally been stated as a nonlinear optimization problem, the MIP formulations for the QAP are known as *linearizations*. Many linearization attempts have been made the first of which is given by Lawler in 1963. This was also the first formulation involving the pair assignment variables ($y_{ijkl} = x_{ij}x_{kl}$) and exploiting the relation between the pair assignment and the single assignment variables. The formulation involves $n^4 + n^2$ variables and $n^4 + 2n + 1$ constraints, and is valid for the general cost coefficient case. Love and Wong (1976) proposed a mixed integer formulation for the case when one of the matrices is the distance matrix of a grid graph. Their formulation requires n^2 binary variables, $4n^2 + 2n$ continuous variables, and $n^2 + 3n$ constraints. Their formulation aims at exploiting the rectilinear structure embedded into the distance matrix. The largest problem size they could cope with was $n = 8$. Kaufman and Broeckx (1978) proposed a linearization involving $2n^2$ variables and $n^2 + 2n$ constraints. They defined the cost incurred by each assignment variable as a decision variable ($w_{ij} = x_{ij} \sum_{k,l=1}^{n} f_{ik} d_{jl} x_{kl}$). Although the number of variables and constraints of Kaufman and Broeckx is much less than the linearization of Lawler, lower bounds generated by the formulation were too weak to be of use. Balas and Mazzola (1980) proposed an exponential-sized linearization that involved a constraint for every possible permutation matrix. They adapted a constraint generation approach to cope with the huge number of constraints. Their formulation was not usable for instances of size $n \geq 10$. Bazaraa and Sherali (1980) applied a cutting plane algorithm by applying Bender's decomposition on a linearized formulation for the QAP. Although they could not prove the optimality of their solutions, they conjectured that their method yielded high quality suboptimal results at the early stages. Kettani and Oral (1993) presented a linearization for the QAP based on the formulation of

Kaufman and Broeckx, together with a method to decrease the number of binary variables. Their linearization required *nlogn* binary variables, n^2 continuous nonnegative variables, and $2n^2 + 4n$ constraints. They were able to solve instances of size $n \leq 15$. Adams and Johnson (1994) presented a linearization that generalized previous linearizations involving the pair assignment variables of Lawler. They proved that lower bounds generated by the LP relaxation of their formulation are always as strong as the Gilmore-Lawler Bound (GLB), which will be mentioned in detail below. Resende, Ramakrishnan, and Drezner (1995) performed a computational test of the lower bounds generated by the relaxation of the formulation by Adams and Johnson (1994). Failing to solve the resulting LP with commercial solvers, they used an experimental interior point method code, called ADP. Problems of size $n \leq 30$ taken from the QAPLIB were used for the experimentation. They reported that, in 87% of the instances they have tested, the formulation yielded the best lower bound known until then. Ramakrishnan, Resende, and Pardalos (1995) implemented a branch-and-bound algorithm for the formulation by Adams and Johnson (1994), and extensively tested the algorithm using instances in QAPLIB. They were able to solve all instances with size $n \leq 15$. Ramachandran and Pekny (1996) provided a formulation involving the so called three-body interaction variables. The number of variables in the formulation was $O(n^6)$ and the number of constraints was $O(n^5)$. Ball, Kaku, and Vakhutinsky (1998) presented two network based linearizations, the first one with $O(n^3)$ nodes and $O(n^4)$ arcs, and the second one with $O(n)$ nodes and $O(n^2)$ arcs. Both linearizations involved the single assignment variables. They made computational experiments with a branch-and-bound algorithm using the first formulation, and a constraint generation approach for the second formulation. They were not able to solve instances of size $n > 8$. Ramakrishnan et al. (2002) performed an empirical analysis of the three-body formulation of Ramachandran and Pekny, and reported that all instances from QAPLIB that are of size $n \leq 12$ are solved at the root node of the branch-and-bound tree. They were not able to

solve larger size problems because of the exceedingly large number of variables and constraints.

There have been a few attempts to analyze the polyhedral structure of the QAP in order to discover valid inequalities that could lead to IP formulations with tighter relaxations. Unfortunately, the results were unfruitful because of the large number of variables. Jünger and Kaibel (2001) analyzed the formulation by Adams and Johnson (1994) and interpreted the formulation as the problem of finding a minimum weighted *n*-clique. They constructed a projection of the original formulation and proved that finding a minimum weighted *n*-clique in the original problem is equivalent to finding a minimum weighted *n*-1 or *n*-2 clique in the projected problem. Furthermore, they claimed that polyhedral investigations were much easier for the projected problem. In their subsequent work (Jünger and Kaibel, 2001), the authors identified a large class of facet defining inequalities which they refer to as *box inequalities*. Their computational experiments showed that adding the box inequalities tightened the relaxation considerably, but the resulting linear problems were hard to solve.

Another relevant track of research on the QAP has been the search for strong lower bounds to be used in a branch-and-bound setting. The first and most famous lower bound, proposed independently by Gilmore (1962) and Lawler (1963), depended on the idea of solving $n^2 + 1$ LAPs of size n. First n^2 LAPs answer the following question: "What is the minimum objective function value for $x_{ij} \sum_{k,l=1}^{n} f_{ik} d_{jl} x_{kl}$ when $x_{ij} = 1$?". Each answer is recorded in the corresponding parameter l_{ij}. A final LAP is solved to obtain the bound for the QAP, for which the objective function cost coefficients are the l_{ij}'s. The complexity of the lower bounding technique is $O(n^5)$ for the case of the general cost coefficients that can be reduced to $O(n^3)$ for the Koopmans–Beckmann form. The bound is very strong for small sized problems but quickly deteriorates as the instance size

increases. Kaku and Thompson (1986) proposed a branch and bound algorithm that use LAPs to calculate certain lower bounds that are similar to Gilmore-Lawler Bound (GLB). As preprocessing, they solved n^2 LAPs of size $(n-1)*(n-1)$. At each node of the branch-and-bound tree, they solved another LAP whose objective coefficients were determined by the branches until that node and the data available from preprocessing. They were able to solve problems up to size $n = 10$. Finke, Burkard, and Rendl (1987), in their survey, elaborated on lower bounding techniques and presented an eigenvalue based bound. Using the trace formulation by Edwards (1977), they proved that the minimal product of eigenvalues of coefficient matrices constitutes a lower bound for the QAP with symmetric matrices. For obtaining tighter lower bounds, they studied the so called reduction techniques that transfer a part of the quadratic terms to linear terms. They analyzed the constant row and column reductions and diagonal reductions, and proved that diagonal reductions are unnecessary. They devised an optimal reduction scheme to transform the quadratic coefficients to the linear coefficients. Carraressi and Malucelli (1992) proposed a way of reformulating QAP so as to transfer the quadratic cost coefficients to linear cost coefficients that is effectively another form of reduction. At the end of the transfer, they solved the linear part to obtain a lower bound. They applied the transfer algorithm iteratively to get a strong lower bound. Although they were able to produce good quality lower bounds, computational complexity of lower bound generation method was $O(kn^5)$, where k is the number of iterations per transfer sequence. Rendl and Wolkowicz (1992) improved the eigenvalue-based lower bound proposed by Finke, Burkard, and Rendl (1987). Using the reduction scheme proposed before, they used a steepest ascent algorithm to find a reduction that would yield a stronger lower bound. Their approach was computationally expensive. On the average, they had to perform 70 eigenvalue computations to find a stronger lower bound. Consequently, the lower bounding mechanism was still too slow for an effective branch and bound approach. Hadley, Rendl, and Wolkowicz (1992) used an orthogonal relaxation of the QAP to come up with improved eigenvalue based bounds. Building upon the trace formulation by

Edwards (1977), they transformed the feasible set of solution matrices from permutation matrices to orthogonal and doubly stochastic (sum of elements each row and column is 1) matrices. They successfully computed bounds that are almost as strong as those of Rendl and Wolkowicz (1992) and computationally not more demanding than the original eigenvalue based bound. Hahn, Grant and Hall (1998) presented a branch-and-bound algorithm based on the Kronecker product formulation for the QAP. Their algorithm employed a dual procedure to compute lower bounds. The dual procedure performed a series of reductions on cost elements C_{ijkl} to decrease the elements while preserving the optimal solutions and nonnegativity of the elements. Anstreicher and Brixius (1999) announced a new lower bounding technique for the QAP, based on convex quadratic programming. Simply, they reinterpreted the derivation of the projected eigenvalue bound by Hadley, Rendl, and Wolkowicz (1992), and added a previously ignored quadratic term. Next, they used an interior point algorithm to approximate the quadratic term. The quality of the resulting lower bound was high and computational complexity was not very high compared to its quality. This lower bound proved to have the best performance among those listed above, in terms of the trade-off between the strength of the bound and the computation time of the bound.

Yet another relevant track of research has been the identification of classes of instances that can be solved easily, though it has been rather limited compared to the rest of the studies on the QAP. Christofides and Benavent (1989) studied the case when the flow matrix represents a tree, and proved that the QAP was NP-Hard even for this special case. They presented a branch-and-bound algorithm, which uses the Lagrangean relaxation of an integer programming formulation of the tree QAP. To solve the relaxation, they used a dynamic programming algorithm. They were able to solve problems up to size 25, in no more than 350 seconds. Chen (1995) proposed three special cases of the general form of the QAP that can be represented as parametric LAPs. The complexity status of these classes is open, but computational results have been reported by Chen (1995) for

test problems up to size 50. Burkard et al. (1995) provided three polynomial time solvable classes of the Koopmans-Beckmann form where one input matrix is monotone Anti-Monge while the other is either symmetric Toeplitz generated by a benevolent (or a k-benevolent) function, or symmetric with bandwidth one. They show that certain assignments qualify as optimal for these cases. Deineko and Woeginger (1998) provided another polynomially solvable class for the Koopmans-Beckmann form with one matrix being Kalmanson and the other being symmetric decreasing circulant. They proved that identity permutation was the optimal solution for this case. They also stated that permuted Kalmanson matrices could be recognized in $O(n^2)$ time and proved that permuted symmetric decreasing circulant matrices could be recognized in $O(n^2)$ time. Burkard et al. (1997) analyzed in their aforementioned survey coefficient matrices with special properties (sum, product, Monge, Anti-Monge, Kalmanson, Toeplitz, circulant) and gave complexity results for many of the resulting cases.

1. 3 Outline of the Dissertation

In this chapter we gave the definition of the problem and provided a brief literature review. We focused on the studies about the exact solution techniques and the classes of instances of the QAP with special structure. In Chapter 2, we present an in-depth analysis of the existing linearization paradigm in the literature and present new linearizations based on our findings. In Chapter 3, we focus on one of the new linearizations and present sets of valid inequalities. We describe a branch-and-cut algorithm that solves problems up to size $n = 30$ and provide extensive experimental results using data from the QAPLIB. In Chapter 4, we shift our focus to classes of instances of the QAP with special structure and provide new polynomially solvable classes. In Chapter 5, we present a lower bounding method that returns lower bounds that are provably at least as strong as the GLB. Finally, in Chapter 6, we present our conclusions and address directions possible future work.

Chapter 2

LINEARIZATIONS

In this chapter, we analyze the existing Mixed Integer Programming (MIP) formulations for the QAP and construct new MIP formulations for the QAP based on our findings. In Section 1, we state our tools of analysis. In Section 2, we perform the analysis and uncover new ways of linearizing the QAP based on our analysis. In Section 3, we construct the IP models based on the observations stated in Section 2. In Section 4, we present our computational experience with the models presented. In Section 5, we give our concluding remarks.

2. 1 Tools of Analysis

Recall that for any feasible solution to the QAP, the decision variables form a permutation matrix each corresponding to a one-to-one onto assignment of facilities to locations. In Figure 2, a small example for $n = 3$ is depicted, where the entry (i,j) of the matrix denotes the value of the assignment variable x_{ij}. The Assignment Matrix in the example represents the solution where facility 1 is assigned to location 1 ($x_{11} = 1$), facility 2 is assigned to location 3 ($x_{23} = 1$), and facility 3 is assigned to location 2 ($x_{32} = 1$).

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

Figure 2: An Example of Assignment Matrix

Perhaps more important than the Assignment Matrix is the Pairwise Assignment Matrix that represents the values of the quadratic terms $x_{ij}x_{kl}$. Although the assignment variables represent the core decisions, the costs are incurred by pairs of assignment variables. It is not an easy task to represent these n^4 values in two dimensions in a structured way. Hahn et al. (1998) use the scheme depicted in Figure 3 that enables us to better understand the structure of the pairwise interactions of assignment decisions. The rows of the Pairwise Assignment Matrix are labeled with facility pairs (i,k) and the columns are labeled with location pairs (j,l). The entry in row (i,k) and column (j,l) of the Pairwise Assignment Matrix is the value of the quadratic term $x_{ij}x_{kl}$. In Figure 3, the row (and column) labels are 11, 12, 13, 21, 22, 23, 31, 32, and 33 where the leading index is shown as a header in the leftmost column and the topmost row. Observe that whenever the ij-entry is 1 in the assignment matrix (of Figure 2), a copy of the assignment matrix is reproduced in the Pairwise Assignment Matrix (of Figure 3) in the submatrix corresponding to the header indices i and j. Observe also that whenever $x_{ij} = 0$ in the Assignment Matrix, all the entries in the submatrix corresponding to header indices i and j are also zero. We note that this matrix is the result of the Kroenecker product of an assignment / permutation matrix X with itself. Simply put, if we denote the Assignment Matrix as X, then submatrix (i,j) of the Pairwise Assignment Matrix is equal to $x_{ij}X$. The objective function value for a given solution is returned by the sum of the cost coefficients corresponding to the 1's in the Pairwise Assignment Matrix (C_{ijkl} for the entry in row (i, k) and column (j,l)).

	j	1			2			3			Location pairs
i	$k\backslash l$	1	2	3	1	2	3	1	2	3	
1	1	1	0	0	0	0	0	0	0	0	
	2	0	0	1	0	0	0	0	0	0	
	3	0	1	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	1	0	0	
	2	0	0	0	0	0	0	0	0	1	
	3	0	0	0	0	0	0	0	1	0	
3	1	0	0	0	1	0	0	0	0	0	
	2	0	0	0	0	0	1	0	0	0	
	3	0	0	0	0	1	0	0	0	0	

Facility pairs

Figure 3: Pairwise Assignment Matrix and its Submatrices

These two matrices play a crucial role in our forthcoming analysis. In the next section, we will demonstrate that the auxiliary variables of the linearizations available in the literature correspond to simultaneous effects of decisions in two subsets of the Assignment Matrix. As a consequence, the auxiliary variables represent the cost incurred by certain subsets of the Pairwise Assignment Matrix. Each such subset corresponds to a subset sum of the quadratic objective function $\sum_{i,j,k,l=1}^{n} C_{ijkl} x_{ij} x_{kl}$. In brief, we will be using the Assignment Matrix, the Pairwise Assignment Matrix, and the quadratic objective function to analyze the patterns of the models in the literature.

2. 2 Analysis of the Formulations in the Literature

As stated in Chapter 1, linear MIP models for the QAP are customarily referred to as *linearizations*. The QAP is originally stated as a nonlinear problem

while any attempt to describe it by linear inequalities and a linear objective function transforms it to a linear MIP. To be able to linearize the QAP, we need to define auxiliary variables that describe the cost contribution of the quadratic interactions. Thus, the core structure of the linearization process is shaped by the way that the auxiliary variables are defined. Generally, each type of auxiliary variable describes the total cost of the simultaneous effects of decisions in some two subsets of the Assignment Matrix. The models we are about to analyze differ in the level of aggregation of these costs.

We now proceed to demonstrate our foregoing observation on the models in the literature. Even though many different linearization techniques have been proposed for various special cases (Love and Wong, 1976; Christofides and Benavent, 1989), models based on Lawler's pairwise assignment variables have dominated the literature. The author defined the variables $y_{ijkl} = x_{ij}x_{kl}$ so as to represent the simultaneous effect of every pair of the assignment decisions.

Later, many other authors used this variable definition to construct linearizations of the QAP (Frieze and Yadegar, 1983; Resende, Ramakrishnan, and Drezner, 1994; Adams and Johnson, 1994). One of these studies by Adams and Johnson (1994) includes a proof of the fact that their linearization is at least as strong as the well-known GLB. The authors also show that many lower bound generation methods may be perceived as Lagrangean relaxations of their formulation. The formulation by Adams and Johnson (1994), which is known to yield the strongest lower bound among the formulations with $O(n^4)$ variables, follows:

(IP1)

$$\min \sum_{i,j,k,l=1}^{n} C_{ijkl}\, y_{ijkl} \qquad (14)$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,...,n \tag{15}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,...,n \tag{16}$$

$$\sum_{k=1}^{n} y_{ijkl} = x_{ij} \quad \forall i, j, l = 1,...,n \tag{17}$$

$$\sum_{l=1}^{n} y_{ijkl} = x_{ij} \quad \forall i, j, k = 1,...,n \tag{18}$$

$$\sum_{i=1}^{n} y_{ijkl} = x_{kl} \quad \forall j, k, l = 1,...,n \tag{19}$$

$$\sum_{j=1}^{n} y_{ijkl} = x_{kl} \quad \forall i, k, l = 1,...,n \tag{20}$$

$$y_{ijkl} = y_{klij} \quad \forall i, j, k, l = 1,...,n : i < k, j \neq l \tag{21}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j = 1,...,n \tag{22}$$

$$y_{ijkl} \in \{0,1\} \quad \forall i, j, k, l = 1,...,n \tag{23}$$

Lawler's linearization with auxiliary variables y_{ijkl} accounts for the simultaneous effect of the pair of assignment variables x_{ij} and x_{kl}. The pairs of subsets of the Assignment Matrix under consideration are the pairs of assignment variables. Any such pair corresponds to a single cell of the Pairwise Assignment Matrix, as depicted in Figure 4.

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 4: Lawler's Linearization

Kaufman and Broeckx (1978) defined the variables $w_{ij} = x_{ij} \sum_{k,l} C_{ijkl} x_{kl}$ to represent by w_{ij} the contribution of each assignment variable x_{ij} to the overall cost. This linearization is somehow less favored in the literature, due to its weaker lower bound. Observe that, in terms of our tools of analysis, the relevant subsets of the Assignment Matrix under consideration are the pairings of each assignment variable with the overall Assignment Matrix. Hence, the variable w_{ij} stands for the simultaneous effect of the assignment variable x_{ij} with the rest of the Assignment Matrix. Figure 5 illustrates such a pairing corresponding to ij = 23 in 5(a) with the entire Assignment Matrix in 5(b). The resulting submatrix in the Pairwise Assignment Matrix corresponding to header indices 2,3 is marked in 5(c). The cost contribution that results from this interaction is the sum of the cost elements that correspond to the 1's in the darkest colored submatrix of Figure 5(c). The formulation by Kaufman and Broeckx follows:

(IP2)

$$\min \sum_{i,j=1}^{n} w_{ij} \qquad (24)$$

s.t.

$$w_{ij} \geq \sum_{k,l=1}^{n} C_{ijkl} x_{kl} - M(1 - x_{ij}) \quad \forall i, j = 1,...,n \tag{25}$$

$$w_{ij} \geq 0 \quad \forall i, j = 1,...,n \tag{26}$$

and (15), (16), (22)

where M is a sufficiently large constant.

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

(a)

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

(b)

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

(c)

Figure 5: Kaufman and Broeckx's Linearization

An even less favored formulation is that of Balas and Mazzola (1980), for which they define a single auxiliary variable, $w = \sum_{i,j,k,l} C_{ijkl} x_{ij} x_{kl}$, to represent the overall cost.

(IP3)

$$\min w \tag{27}$$

$$w \geq \sum_{\substack{\forall i,j,k,l: \\ x^*_{kl}=1}} C_{ijkl} x^*_{ij} x_{kl} - \sum_{\substack{\forall k,l: \\ x^*_{kl}=0}} M x_{kl} \quad \forall x^* \in QAP^n \tag{28}$$

$$w \geq 0 \tag{29}$$

and (15), (16), (22)

where QAP^n denotes the set of all feasible solutions for a QAP of size n.

Constraint set (28) forces the single auxiliary variable to be greater than or equal to the objective function value yielded by x^*, if $x = x^*$, and has no effect otherwise. In this case, the authors define a single variable that pairs up the Assignment Matrix with itself. Figure 6(a)(b) identify the submatrices corresponding to the auxiliary variable w while the interaction of these submatrices yields the entire Pairwise Assignment Matrix shown in 6(c).

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

(a)

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

(b)

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

(c)

Figure 6: Balas and Mazzola's Linearization

From our observations up to this point, we can see that any possible partitioning of the Pairwise Assignment Matrix into any two subsets would result in a different linearization. An auxiliary variable is introduced by each such

pairing. In order to better model the QAP, we feel the need to identify certain patterns in the foregoing formulations. We can easily observe that there are two types of subsets of the Assignment Matrix that have been used in the literature in the modeling process. One type is a single entry of the Assignment Matrix (i.e. single assignment variables) and the other type is the whole Assignment Matrix. Pairings of type I or type II with type I or type II yield the linearizations of Lawler, of Kaufman and Broeckx, and of Balas and Mazzola. Even though not seen in the literature (except Erdoğan and Tansel, 2005), the columns and the rows of the Assignment Matrix are also eligible to be used for constructing linearizations. The four types of subsets (single entry, column, row, and the whole Assignment Matrix) seem to yield the best physical interpretations for the corresponding cost aggregations. So, we base our analysis on these subsets. The graph in Figure 7 depicts the current situation of the literature. Each subset is denoted as a node and the arcs (solid lines) denote the interactions analyzed in the literature. The arcs depicted by broken lines in the figure are linearizations that have not yet been analyzed in the literature except for the linearizations introduced in a recent work of ours (Erdoğan and Tansel, 2005). We want to emphasize that each arc implies a unidirectional relation. For example, one can define the variables of Kaufman and Broeckx as the interaction of the Assignment Matrix and a single assignment variable ($\overline{w}_{kl} = x_{kl} \sum_{i,j} C_{ijkl} x_{ij}$) and still construct the same formulation. In the next section, we focus on the interactions corresponding to broken lines and present the resulting new formulations for the QAP.

2. 3 New Formulations

To complete the picture in Figure 7, let us start with the interaction between a variable and a column. The corresponding submatrix in the Pairwise Assignment Matrix is a column of the submatrix corresponding to the assignment variable, as

depicted in Figure 8. Following the examples above, we define the variable $y_{ijl} = x_{ij} \sum_{k=1}^{n} C_{ijkl} x_{kl}$. This nonlinear representation shows that the variable y_{ijl} represents the interaction between the assignment variable x_{ij} and the l'th column of the Assignment Matrix. Using this variable definition, the following formulation is constructed:

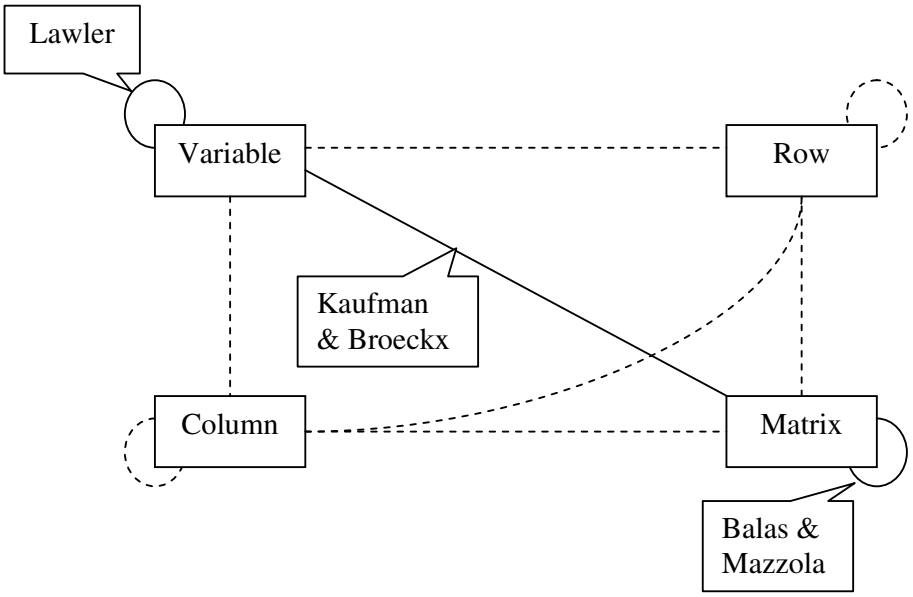

Figure 7: An Overview of the Literature

(IP4)

$$\min \sum_{i,j,l=1}^{n} y_{ijl} \tag{30}$$

$$y_{ijl} \geq \sum_{k=1}^{n} C_{ijkl} x_{kl} - M(1 - x_{ij}) \quad \forall i, j, l = 1,...,n \tag{31}$$

$$y_{ijl} \geq 0 \quad \forall i, j, l = 1,...,n \tag{32}$$

and (15), (16), (22)

where *M* is a large constant. Observe that constraint (31) forces the variable y_{ijl} to be greater than or equal to $\sum_{k=1}^{n} C_{ijkl} x_{kl}$ whenever $x_{ij} = 1$, and has no effect otherwise.

		Location *j*		
		1	2	3
Facility *i*	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location *l*		
		1	2	3
Facility *k*	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	*k**l*	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 8: Multicommodity Flow Formulation

The way we construct constraint (31) will serve as an example for the formulations to follow in this section. Whenever one of the subsets is a single assignment variable, the simultaneous effect of the subsets is linearized as soon as the variable is decided. For this case, $y_{ijl} = \sum_{k=1}^{n} C_{ijkl} x_{kl}$ if $x_{ij} = 1$, and $y_{ijl} = 0$ if $x_{ij} = 0$. Hence, we have constructed constraint (31) by enumeration on possible values x_{ij} can take, i.e. to force a lower bound of $\sum_{k=1}^{n} C_{ijkl} x_{kl}$ on y_{ijl} whenever $x_{ij} = 1$, and to have no effect whenever $x_{ij} = 0$. The same constraint construction method may be used if one of the subsets is a column or a row, by enumeration on the possible assignments in the column or row. For example the linearization above can be recast as:

$$y_{ijl} \geq C_{ijkl} x_{ij} - M(1 - x_{kl}) \quad \forall i, j, k, l = 1,...,n \tag{33}$$

Notice that constraint set (33) is constructed by enumeration on the decisions in column *l*, i.e. for each possible decision x_{kl}. It forces a lower bound of $C_{ijkl} x_{ij}$ on y_{ijl} whenever $x_{kl} = 1$, and has no effect whenever $x_{kl} = 0$.

Although this formulation is quite similar to the formulation of Kaufman and Broeckx, this modeling paradigm gives us a structure to exploit for the Koopmans-Beckmann form. For the Koopmans-Beckmann form, the variable definition becomes $y_{ijl} = x_{ij} \sum_{k=1}^{n} f_{ik} d_{jl} x_{kl}$ which can further be simplified to $y'_{ijl} = x_{ij} \sum_{k=1}^{n} f_{ik} x_{kl}$, with the term d_{jl} becoming the objective function coefficient of y'_{ijl}. The variable y'_{ijl} also has a physical meaning: it represents the total material flow from location *j* to location *l* whenever facility *i* is located at *j*. This flow based interpretation suggests adding flow conservation constraints into the formulation:

(IP4')

$$\min \sum_{i,j,l=1}^{n} d_{jl} y'_{ijl} \tag{34}$$

s.t.

$$\sum_{l=1}^{n} y'_{ijl} = x_{ij} \left(\sum_{k=1}^{n} f_{ik} \right), \forall i, j = 1,...,n \tag{35}$$

$$\sum_{j=1}^{n} y'_{ijl} = \sum_{k=1}^{n} f_{ik} x_{kl}, \forall i, l = 1,...,n \tag{36}$$

$$y'_{ijl} \geq \sum_{k=1}^{n} f_{ik} x_{kl} - M(1 - x_{ij}) \quad \forall i, j, l = 1,...,n \tag{37}$$

$$y'_{ijl} \geq 0, \forall i, j, l = 1,...,n \tag{38}$$

and (15), (16), (22)

We now prove the validity of IP4'.

Theorem 1: Let x be a feasible solution to an instance of the QAP defined by matrices F and D with objective value $z_{QAP}(x)$. Then, there exists a unique vector y' such that (x, y') is feasible to IP4' with objective value $z_{IP4'}(x, y') = z_{QAP}(x)$.

Proof: With x being feasible to the QAP, constraints (15), (16), and (22) are satisfied. For each $i \in \{1,\ldots,n\}$, let $a(i)$ be the location index j for which $x_{ij} = 1$. Similarly, for each location j, let $a^{-1}(j)$ be the facility index i for which $x_{ij} = 1$. Since $x_{ij} = 0 \ \forall \ j \neq a(i)$, (35) and (38) imply that $y'_{ijl} = 0 \ \forall \ j \neq a(i)$ and $i \in \{1,\ldots,n\}$. Consequently, the left side of (36) gives $y'_{ia(i)l}$ (because all terms except for $j = a(i)$ are zero), while the right side of (36) gives $f_{ia^{-1}(l)}$ (because all terms except for $l = a^{-1}(j)$ are zero). Hence, y' is uniquely determined by the equations

$$y'_{ia(i)l} = f_{ia^{-1}(l)} \forall i, l \in \{1,...,n\} \tag{39}$$

and

$$y'_{ijl} = 0 \ \forall \ j \neq a(i) \text{ and } i, l \in \{1,\ldots,n\} \tag{40}$$

Solution y' constructed in this way satisfies (38). It also satisfies (36) by construction. The only remaining possibility to be checked is constraint (35). If $j \neq a(i)$, then (35) gives zero on both sides. If $j = a(i)$, the left side of (35) is

$\sum_{l=1}^{n} y'_{ia(i)l}$ while the right side is $\sum_{k=1}^{n} f_{ik}$. Since $y'_{ia(i)l} = f_{ia^{-1}(l)}$ by construction, the left side is $\sum_{l=1}^{n} f_{ia^{-1}(l)}$, which is the same as $\sum_{k=1}^{n} f_{ik}$. This proves the uniqueness and feasibility of (x, y') to IP4' for each feasible x to QAP.

To prove $z_{IP4'}(x,y) = z_{QAP}(x)$, observe that $f_{ik} d_{jl} x_{ij} x_{kl} = 0$, unless $i = a^{-1}(j)$ and $k = a^{-1}(l)$, in which case it is $f_{a^{-1}(j)a^{-1}(l)} d_{jl}$. Since the objective value of IP4' gives $\sum_{j,l=1}^{n} d_{jl} f_{a^{-1}(j)a^{-1}(l)}$, it is the same as $\sum_{i,j,k,l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl}$. □

IP4 and IP4' has $O(n^3)$ variables and $O(n^3)$ constraints. Both formulations are valid for arbitrary distance data. Notice that the proof of Theorem 1 does not involve the constraint set (37). This formulation will be referred as the *Multicommodity Flow Formulation* for the rest of this study. Next, we focus on the interaction between a variable and a row. The corresponding reflection on the Pairwise Assignment Matrix is a row of the submatrix corresponding to the assignment variable, as depicted in Figure 9.

Following the examples above, we define the variable $t_{ijk} = x_{ij} \sum_{l=1}^{n} C_{ijkl} x_{kl}$. This representation shows that the variable t_{ijk} represents the interaction between the assignment variable x_{ij} and k'th row of the Assignment Matrix. Using this variable definition, a formulation conjugate to IP4 may be constructed. Similarly, the extension of this formulation to the Koopmans-Beckmann form results in variables representing the induced distance between two facilities (and another formulation conjugate to IP4'). Although the word "commodity" does not make sense when we speak of distances, since this formulation is the conjugate of the

Multicommodity Flow Formulation presented above, it will be referred to as the *Multicommodity Distance Formulation* in the rest of this study. We omit these formulations for the sake of brevity.

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 9: Multicommodity Distance Formulation

Next we focus on the interactions of columns of the Assignment Matrix. As depicted in Figure 10, the reflection of this interaction is a column of the Pairwise Assignment Matrix.

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 10: Single Commodity Flow Formulation

We define the corresponding variable as $y_{jl} = \sum_{i,k} C_{ijkl} x_{ij} x_{kl}$. The variable y_{jl} represents the interaction between the *j*'th and *l*'th columns of the Assignment Matrix. The resulting formulation follows:

(IP5)

$$\min \sum_{j,l=1}^{n} y_{jl} \tag{41}$$

s.t.

$$y_{jl} \geq \sum_{k=1}^{n} C_{ijkl} x_{kl} - M(1 - x_{ij}) \quad \forall i, j, l = 1,...,n \tag{42}$$

$$y_{jl} \geq 0, \forall j, l = 1,...,n \tag{43}$$

and (15), (16), (22)

Constraint (42) forces the variable y_{jl} to be greater than or equal to $\sum_{k=1}^{n} C_{ijkl} x_{kl}$ whenever $x_{ij} = 1$, and has no effect otherwise. As in the case of (IP4), this modeling paradigm allows us to move towards a more specific formulation for the Koopmans-Beckmann form. For this form, the variable definition becomes $y_{jl} = \sum_{i,k=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl}$ which can be further simplified to $y'_{jl} = \sum_{i,k=1}^{n} f_{ik} x_{ij} x_{kl}$ where the term d_{jl} becomes the objective function coefficient of the variable y'_{jl}. This variable definition also has a physical meaning: it defines (gives) the total material flow from location *j* to location *l*. This gives us the opportunity to add flow conservation constraints to the formulation:

(IP5')

$$\min \sum_{j,l=1}^{n} d_{jl} y'_{jl} \tag{44}$$

s.t.

$$\sum_{l=1}^{n} y'_{jl} = \sum_{i=1}^{n} \left(\sum_{k=1}^{n} f_{ik} \right) x_{ij}, \forall j = 1,...,n \tag{45}$$

$$\sum_{j=1}^{n} y'_{jl} = \sum_{k=1}^{n} \left(\sum_{i=1}^{n} f_{ik} \right) x_{kl}, \forall l = 1,...,n \tag{46}$$

$$y'_{jl} \geq \sum_{k=1}^{n} f_{ik} x_{kl} - M(1 - x_{ij}) \quad \forall i, j, l = 1,...,n \tag{47}$$

$$y'_{jl} \geq 0, \forall j, l = 1,...,n \tag{48}$$

and (15), (16), (22)

Constraint (47) forces the variable y'_{jl} to be greater than or equal to $\sum_{k=1}^{n} f_{ik} x_{kl}$ whenever $x_{ij} = 1$, and has no effect otherwise. We now prove the validity of the formulation.

Theorem 2: Let x be a feasible solution to an instance of the QAP. Then, there exists a unique y' such that (x, y') is feasible to IP5' with objective function value $z_{IP5'}(x, y') = z_{QAP}(x)$.

Proof: With x being feasible to the QAP, constraints (15), (16), and (22) are satisfied. With $a^{-1}(j)$ denoting the facility index i for which $x_{ij} = 1$, (45) and (46) give, respectively, that $\sum_{l=1}^{n} y'_{jl} = \sum_{k=1}^{n} f_{a^{-1}(j)k}$ and $\sum_{j=1}^{n} y'_{jl} = \sum_{i=1}^{n} f_{ia^{-1}(l)}$. Constraint

(47) sets the exact lower bound of each flow as $y'_{il} \geq f_{a^{-1}(j)a^{-1}(l)}$. This upper bound must be satisfied as equality, otherwise constraints (45) and (46) are violated. This proves the uniqueness and feasibility of (x, y') to IP5' for each feasible x to QAP.

To prove $z_{IP5'}(x,y) = z_{QAP}(x)$, observe that $y'_{il} = f_{a^{-1}(j)a^{-1}(l)}$, implying that the objective value of IP5' is $\sum_{j,l=1}^{n} d_{jl} f_{a^{-1}(j)a^{-1}(l)}$ which is the same as $\sum_{i,j,k,l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl}$.□

This formulation will be referred to as the *Single Commodity Flow Formulation* in the rest of this study. IP5 and IP5' has $O(n^2)$ variables and $O(n^3)$ constraints. Note that both formulations are valid for arbitrary distance data. Unlike the Multicommodity Flow Formulation, the flow conservation constraints are not sufficient to define the set of feasible solutions. As stated in the proof, $y'_{il} = f_{a^{-1}(j)a^{-1}(l)}$ for every permutation a, which implies that variables y' take permuted values of the flow matrix. This provides an opportunity to exploit any structure that may be available in the flow matrix.

We next focus on the interactions between the rows of the Assignment Matrix. In this case, the induced subset is a row of the Pairwise Assignment Matrix, as depicted in Figure 11.

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 11: Single Commodity Distance Formulation

Following the scheme above, we define the variable $t_{ik} = \sum_{j,l=1}^{n} C_{ijkl} x_{ij} x_{kl}$. The variable t_{ik} represents the interaction between the i'th and k'th rows of the Assignment Matrix. Using this variable definition, a formulation conjugate to IP5 may be constructed. Similar to the Multicommodity Flow Formulation case, the extension of this formulation to the Koopmans-Beckmann form results in variables representing the induced distance between two facilities (and another formulation conjugate to IP5'). Hence, these formulations will be referred to as the *Single Commodity Distance Formulation* in the rest of this study. We omit this formulation as it is straightforward to derive it.

We now focus on the interactions between the row of the Assignment Matrix and the whole Assignment Matrix. In this case, the induced submatrix in the Pairwise Assignment Matrix is a submatrix consisting of all rows associated with the header index of the chosen row in the Assignment Matrix (depicted in Figure 12).

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 12: Facility-Based Formulation

For this case, we define the variable $w_i = \sum_{j,k,l=1}^{n} C_{ijkl} x_{ij} x_{kl}$. The variable w_i represents the interaction between the i'th row of the Assignment Matrix the whole Assignment Matrix. Using this variable definition, the following formulation may be constructed:

(IP6)

$$\min \sum_{i=1}^{n} w_i \tag{49}$$

$$w_i \geq \sum_{k,l=1}^{n} C_{ijkl} x_{kl} - M(1 - x_{ij}) \quad \forall i, j = 1,...,n \tag{50}$$

$$w_i \geq 0 \quad \forall i = 1,...,n \tag{51}$$

and (15), (16), (22)

Constraint (50) forces the variable w_i to be greater than or equal to $\sum_{k,l=1}^{n} C_{ijkl} x_{kl}$ whenever $x_{ij} = 1$, and has no effect otherwise. The physical interpretation of w_i is the cost incurred by the i'th facility. For this reason, we refer to this formulation as the *Facility-Based* formulation. This formulation resembles that of Kaufman and Broeckx to a great extent. The main difference is in the number of variables. In essence, our w_i is their $\max_j w_{ij}$.

We now focus on the interactions between the columns of the Assignment Matrix and the whole Assignment Matrix. In this case, the induced submatrix in the Pairwise Assignment Matrix is a submatrix consisting of all columns associated with the header index of the chosen column in the Assignment Matrix (Figure 13).

		Location j		
		1	2	3
Facility i	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location l		
		1	2	3
Facility k	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	$k\backslash l$	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 13: Location-Based Formulation

For this case, we define the variable $w'_j = \sum_{i,k,l=1}^{n} C_{ijkl} x_{ij} x_{kl}$. The variable w'_j represents the interaction between the j'th column of the Assignment Matrix and

the whole Assignment Matrix. Using this variable definition, a formulation similar to IP6 may be constructed directly. Since the physical interpretation of $w_j^{'}$ is the cost incurred by the *j*'th location, we will be referring to this formulation as the *Location-Based* formulation.

Our final focus is on the interactions between columns and rows of the Assignment Matrix. In this case, the induced submatrix in the Pairwise Assignment Matrix is a set of parallel rows in the submatrix of the Pairwise Assignment Matrix corresponding to the header index of the chosen column of the Assignment Matrix (Figure 14).

		Location *j*		
		1	2	3
Facility *i*	1	1	0	0
	2	0	0	1
	3	0	1	0

		Location *l*		
		1	2	3
Facility *k*	1	1	0	0
	2	0	0	1
	3	0	1	0

	j	1			2			3		
i	*k**l*	1	2	3	1	2	3	1	2	3
1	1	1	0	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0
	3	0	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1	0	0
	2	0	0	0	0	0	0	0	0	1
	3	0	0	0	0	0	0	0	1	0
3	1	0	0	0	1	0	0	0	0	0
	2	0	0	0	0	0	1	0	0	0
	3	0	0	0	0	1	0	0	0	0

Figure 14: Facility-Location Formulation

For this case, we define the variable $r_{il} = \sum_{j,k=1}^{n} C_{ijkl} x_{ij} x_{kl}$. The variable r_{il} represents the interaction between the *i*'th column of the Assignment Matrix and the *l*'th row of the Assignment Matrix. Using this variable definition, the following formulation is constructed:

(IP7)

$$\min \sum_{i,l=1}^{n} r_{il} \tag{52}$$

$$r_{il} \geq \sum_{k=1}^{n} C_{ijkl} x_{kl} - M(1 - x_{ij}) \quad \forall i, j, l = 1,...,n \tag{53}$$

$$r_{il} \geq 0 \quad \forall i, l = 1,...,n \tag{54}$$

and (15), (16), (22)

Constraint (53) forces the variable r_{il} to be greater than or equal to $\sum_{k=1}^{n} C_{ijkl} x_{kl}$ whenever $x_{ij} = 1$, and has no effect otherwise. In the foregoing formulations, the physical meanings of the auxiliary variables were either related to the facilities, or to the locations but not both. For IP7, the auxiliary variables denote the interaction cost that arises from assignment of a facility to a location. Hence we name this formulation as the *Facility-Location* based formulation.

2. 4 Computational experience

In this section, we present our results on selected instances of the QAP. We have implemented all the formulations presented in this chapter, except that of Balas and Mazzola (1980). The most succesful formulations that use Lawler's and Kaufman and Broeckx's variables are given by Adams and Johnson (1994), and Kettani and Oral (1993), respectively. Hence, we used those formulations in our implementations. To avoid excessively large run times, we have constructed five instances of size 8 (`chr08'`, `had08'`, `nug08'`, `rou08'`, `scr08'`) by removing four rows and columns from the flow and distance

matrices of corresponding instance in QAPLIB. CPLEX 9.1 MIP solver was used for optimizing the resulting integer programs. The runs were conducted on a single PC (3.0 Ghz Dell OPTIPLEX with 2GB RAM). The results are presented in Table 1.

It can be observed that the lower bounds yielded by the formulations that do not involve equalities (Kettani-Oral, Facility-Based, Location-Based, and Facility-Location) are 0. Although the run times for these formulations are short for certain instances, the large number of nodes traversed makes them poor candidates for solving large instances. To the contrary, Adams-Johnson formulation yields the strongest lower bounds without exception, but the number of variables ($O(n^4)$) and constraints ($O(n^3)$) it involves is too high to be used for larger instances. Notably, Multicommodity Flow and Single Commodity Flow formulations yield a relatively strong lower bound and consequently traverse a relatively small number of nodes.

Table 1: Computational Results

Data File	*Formulation*	*Optimum Solution Value*	*LP Relaxation Value*	*Number of nodes*	*CPU Time (sec)*
chr08′	Adams-Johnson	7638.00	7638.00	0	0.28
chr08′	Kettani-Oral	7638.00	0.00	357	0.22
chr08′	Multicommodity Flow	7638.00	6942.62	0	0.08
chr08′	Multicommodity Distance	7638.00	0.00	1106	19.12
chr08′	Single Commodity Flow	7638.00	6572.85	5	0.19
chr08′	Single Commodity Distance	7638.00	0.00	696	3.70
chr08′	Facility-Based formulation	7638.00	0.00	1474	0.52
chr08′	Location-Based formulation	7638.00	0.00	428	0.22
chr08′	Facility-Location formulation	7638.00	0.00	2722	1.56
had08′	Adams-Johnson	556.00	556.00	0	1.55
had08′	Kettani-Oral	556.00	0.00	23820	11.91
had08′	Multicommodity Flow	556.00	368.12	2504	29.34
had08′	Multicommodity Distance	556.00	334.22	326	6.03
had08′	Single Commodity Flow	556.00	419.86	10842	35.19
had08′	Single Commodity Distance	556.00	459.84	2474	8.36

had08'	Facility-Based formulation	556.00	0.00	46742	19.26
had08'	Location-Based formulation	556.00	0.00	48380	17.80
had08'	Facility-Location formulation	556.00	0.00	1309413	1382.93
nug08'	Adams-Johnson	214.00	203.50	12	10.41
nug08'	Kettani-Oral	214.00	0.00	16633	7.23
nug08'	Multicommodity Flow	214.00	154.00	532	5.30
nug08'	Multicommodity Distance	214.00	0.00	4996	43.75
nug08'	Single Commodity Flow	214.00	154.00	1993	4.83
nug08'	Single Commodity Distance	214.00	48.00	9109	19.81
nug08'	Facility-Based formulation	214.00	0.00	21811	8.67
nug08'	Location-Based formulation	214.00	0.00	20953	7.74
nug08'	Facility-Location formulation	214.00	0.00	47921	50.55
rou08'	Adams-Johnson	126894.00	126264.57	6	4.52
rou08'	Kettani-Oral	126894.00	0.00	27552	13.69
rou08'	Multicommodity Flow	126894.00	37247.50	1156	25.84
rou08'	Multicommodity Distance	126894.00	58937.48	1044	21.88
rou08'	Single Commodity Flow	126894.00	95117.51	3859	14.81
rou08'	Single Commodity Distance	126894.00	97766.92	3232	13.23
rou08'	Facility-Based formulation	126894.00	0.00	38147	19.61
rou08'	Location-Based formulation	126894.00	0.00	39370	16.49
rou08'	Facility-Location formulation	126894.00	0.00	>600000	>3600.00
scr08'	Adams-Johnson	21586.00	20974.00	4	5.74
scr08'	Kettani-Oral	21586.00	0.00	2391	1.02
scr08'	Multicommodity Flow	21586.00	18676.00	142	1.92
scr08'	Multicommodity Distance	21586.00	0.00	3786	33.86
scr08'	Single Commodity Flow	21586.00	18676.00	419	1.99
scr08'	Single Commodity Distance	21586.00	2424.00	7293	21.94
scr08'	Facility-Based formulation	21586.00	0.00	4420	1.78
scr08'	Location-Based formulation	21586.00	0.00	2467	1.06
scr08'	Facility-Location formulation	21586.00	0.00	125481	43.11

2. 5 Concluding Remarks

In this chapter, we have analyzed the existing literature with respect to the way the auxiliary variables are defined and presented seven new linearizations for the QAP based on the viewpoint we proposed. All seven new formulations involve big *M*'s, which are known to weaken the LP relaxation. Since our focus is on modeling rather than solving, possible values of the big *M*'s are not computed. Big *M*'s were not unexpected, since the formulations involve continuous variables that are not restricted to the interval [0,1]. A final view of

the linearizations, organized with respect to the variable definitions, is given in Figure 15. Existing formulations have been emphasized using boldface characters and shading.

The formulations that stand out are Multicommodity and Single Commodity Flow Formulations (and conjugate distance formulations), which involve flow conservation equations. Specifically the auxiliary variables of the Single Commodity Flow Formulation (and conjugate distance formulation) take permuted values of the entries of the flow (and distance) matrix. This fact presents us with the opportunity to exploit any available special structure in the input matrices that may be in line with the auxiliary variables. In the next chapter, we focus on the Single Commodity Flow and Distance Formulations, present valid inequalities, and give a branch-and-cut algorithm.

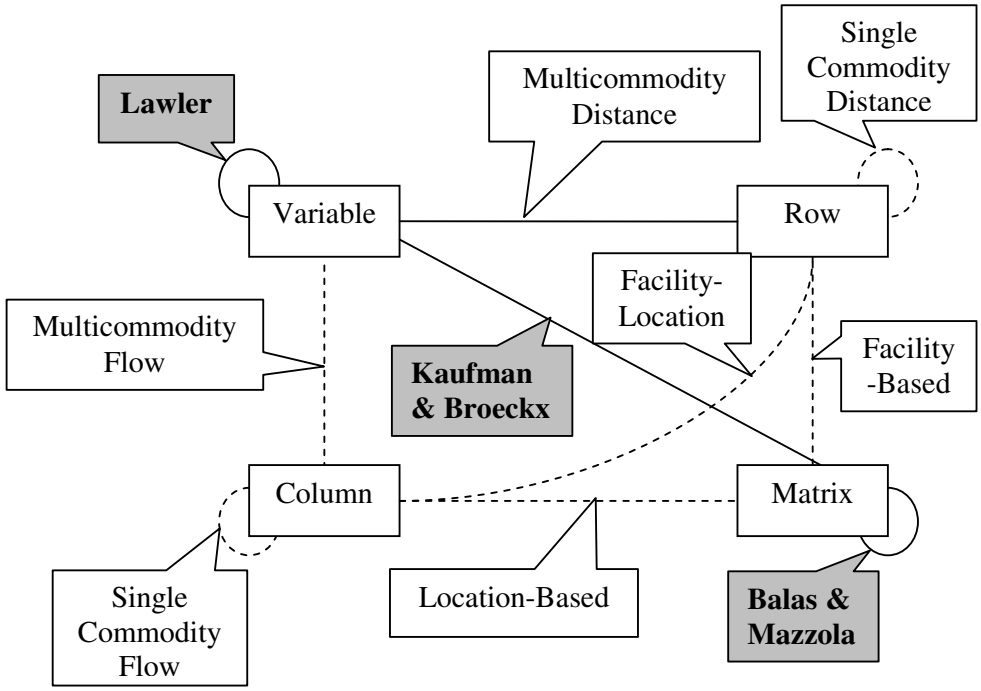

Figure 15: A Final View of the Literature

Additional Insights

- Although the linearization of Adams and Johnson yields the best lower bounds, the formulation is not only huge but also highly degenerate. The LP relaxation of the formulation for nug12 requires 453.80 CPU seconds to be solved by the dual simplex solver of CPLEX 9.1 on a 2.0 GHZ PC with 2 GB RAM.
- The Facility-Location Formulation may be used as an example of the pathological cases where a Mixed Integer Programming model fails entirely. The run was stopped after an hour at which time more than 600000 nodes were traversed and the memory requirement was about 120 MB.
- Generic cuts are usually not very effective for the linearizations of QAP. However, for Multicommodity Flow Formulation the flow cover inequalities generated by CPLEX were quite useful. The Single Commodity Flow Formulation was selected for experiments with larger instances since it is smaller and the valid inequalities proposed in Chapter 3 are experimentally observed to be stronger.
- The linearizations given in this chapter may be used for checking if a solution with a given objective function value exists. For example, let us name the formulation obtained by adding $\sum_{i,j,l=1}^{n} d_{jl} y'_{ijl} \leq z'$ to IP4' as IP4''. If IP4'' has a feasible solution, then the solution is also feasible for the QAP. If this feasibility problem can be solved efficiently, a bisection on possible values of z' can lead to an efficient solution method. The flow structure of IP4' (and IP4'') seems suitable for such an experiment, at first glance. However, initial experimental results were not encouraging, so the idea is later abondoned.

Chapter 3

FLOW BASED FORMULATIONS

In operations research, the common approach is to model the problem on hand in a way that is as independent from the problem instance as possible. Modelers often tend to dump the data into the objective function and focus on solving a well-defined, static constraint matrix. As stated in the introduction chapter, this paradigm has been unfruitful beyond certain sizes for the QAP, due to the large number of variables required. In our opinion, to be able to solve the QAP exactly, one needs to incorporate the data into the model, and exploit a pattern that may exist in the data. The Single Commodity Flow and Distance Formulations we have presented in the previous chapter seem to be well-suited for this task, as the auxiliary variables in these formulations take permuted values of the entries of the flow and distance matrices, respectively. This provides an opportunity to exploit special properties of the flow or distance matrices that may be consistent with definitions of auxiliary variables.

Before going into details, we find it useful to give a brief introduction to valid inequalities and branch-and-cut. In solving an integer (or mixed integer) programming problem, it is desirable to have the description of the convex hull of feasible integral solutions to be able to obtain the solution by solving an associated linear program whose feasible set coincides with this convex hull. The exact description of the convex hull by means of linear inequalities is not possible in general using a polynomial number of variables and constraints. In practice, a formulation that closely approximates the convex hull (without

leaving out any feasible integer solutions) by a larger polyhedral set and uses a manageable number of variables and constraints is considered to be a good formulation. The linear programming relaxation of the integer program gives a feasible set that contains the convex hull. This outer approximation can be improved by adding *valid inequalities* that slice off portions of the feasible region of the LP relaxation without cutting off any portion of the convex hull (of the feasible integer solutions). Adding all possible valid inequalities at once may inflate the problem size beyond the computational reach, so it is desirable to work with a subset of valid inequalities and add a new one whenever a valid inequality is identified that is violated by the current outer approximation. The generic algorithm consisting of the well known branch-and-bound method together with the detection and addition of violated valid inequalities is called branch-and-cut. We refer the interested reader to Wolsey (1998) for a complete exposition.

In this chapter, we elaborate on the Single Commodity Distance Formulation of the QAP with variables representing induced distances between facilities, and present sets of valid inequalities. For completeness, we restate the Single Commodity Flow Formulation and also give the Single Commodity Distance Formulation that was previously omitted. In Section 2.1, we present sets of valid inequalities. In Section 2.2, we present the results of our computational experiments together with the details of the implementation. In Section 2.3, we give concluding remarks.

The *Single Commodity Flow Formulation* that was introduced in Chapter 2 is given below:

(IP5')

$$\min \sum_{j,l=1}^{n} d_{jl} y'_{jl} \qquad (44)$$

s.t.

$$\sum_{l=1}^{n} y'_{jl} = \sum_{i=1}^{n}\left(\sum_{k=1}^{n} f_{ik}\right)x_{ij}, \forall j = 1,...,n \tag{45}$$

$$\sum_{j=1}^{n} y'_{jl} = \sum_{k=1}^{n}\left(\sum_{i=1}^{n} f_{ik}\right)x_{kl}, \forall l = 1,...,n \tag{46}$$

$$y'_{jl} \geq \sum_{k=1}^{n} f_{ik} x_{kl} - M_i(1 - x_{ij}) \quad \forall i, j, l = 1,...,n \tag{47}$$

$$y'_{jl} \geq 0, \forall j, l = 1,...,n \tag{48}$$

and (15), (16), (22)

Note that the minimum possible value for the parameter *M* is computed as dependent on the facility *i*. Let $M_i = \max_{\substack{k,m\in[1,...,n]\\ k\neq m}} f_{ik} - f_{im}$. The conjugate *Single Commodity Distance Formulation* is as follows:

(IP8)

$$\min \sum_{i,k=1}^{n} f_{ik} t_{ik} \tag{55}$$

s.t.

$$\sum_{k=1}^{n} t_{ik} = \sum_{j=1}^{n}\left(\sum_{l=1}^{n} d_{jl}\right)x_{ij}, \forall i = 1,...,n \tag{56}$$

$$\sum_{i=1}^{n} t_{ik} = \sum_{l=1}^{n}\left(\sum_{j=1}^{n} d_{jl}\right)x_{kl}, \forall l = 1,...,n \tag{57}$$

$$t_{ik} \geq \sum_{l=1}^{n} d_{jl} x_{kl} - M_j(1 - x_{ij}) \quad \forall i, j, k = 1,...,n \tag{58}$$

$$t_{ik} \geq 0, \forall i, k = 1,...,n \tag{59}$$

and (15), (16), (22)

Note that the minimum possible value for M in IP8 is computed as dependent on the location j, where $M_j = \max_{\substack{l,m\in[1,\ldots,n] \\ l\neq m}} d_{jl} - d_{jm}$.

3.1 Valid Inequalities

3.1.1 Triangle inequalities:

We now restrict attention to distance matrices D that are symmetric and triangulated. That is, we assume $d_{ab} = d_{ba}$ $\forall a,b = 1,\ldots,n$ and that $d_{ab} \leq d_{ac} + d_{cb}, \forall a,b,c = 1,\ldots,n$. As stated in the previous chapter, the induced distances will be a permutation of the original distances. Consequently, the triangle inequalities are still valid when expressed in terms of the t variables representing the induced distances.

Theorem 3: For an instance of the QAP whose distance matrix obeys the triangle inequalities, the inequalities $t_{ik} \leq t_{im} + t_{mk}, \forall i,k,m : i \neq k \neq m$ are valid for instances of the QAP with symmetric and triangulated distance matrices.

Proof: Let (x,t) be any feasible solution and consider an arbitrary triplet of distinct facility indices i, k, m. Let a, b, c be the location indices for which $x_{ia} = x_{kb} = x_{mc} = 1$. Then $t_{ik} = d_{ab}$, $t_{im} = d_{ac}$, and $t_{mk} = d_{cb}$. Since the original distance matrix satisfies the triangle inequalities, we have $d_{ab} \leq d_{ac} + d_{cb}$, which implies that $t_{ik} \leq t_{im} + t_{mk}$. □

Another interpretation of the triangle inequality is $\max_{\substack{a,b=1,\ldots,n \\ a\neq b}} (d_{ab} - d_{ac} - d_{cb}) \leq 0, \forall c = 1..n$. Let

$T_c = \max_{\substack{a,b=1,\ldots,n \\ a \neq b}} (d_{ab} - d_{ac} - d_{cb}), \forall c = 1,\ldots,n$. A more general form of triangle inequalities is defined using T_c as follows.

Theorem 4: The inequalities $t_{ik} - t_{im} - t_{mk} \leq \sum_{c=1}^{n} T_c x_{mc}, \forall i,k,m : i \neq k \neq m$ are valid for instances of the QAP with arbitrary distance matrices.

Proof: Let (x,t) be any feasible solution, and consider an arbitrary triplet of distinct facility indices i, k, m. Let a, b, c be the location indices, for which $x_{ia} = x_{kb} = x_{mc} = 1$. Then $t_{ik} = d_{ab}$, $t_{im} = d_{ac}$, and $t_{mk} = d_{cb}$. Since the original distance matrix satisfies the inequality $d_{ab} - d_{ac} - d_{cb} \leq T_c$ (by definition of T_c), we conclude that $t_{ik} - t_{im} - t_{mk} \leq \sum_{c=1}^{n} T_c x_{mc}$. □

Note that violated triangular inequalities can be identified in $O(n^3)$ time by simply checking all the inequalities. Violation of the generalized form of triangle inequalities can be identified by computing and storing $\sum_{c=1}^{n} T_c x_{mc}$ for each m in $O(n^2)$ time and then checking all the inequalities in a time bound of $O(n^3)$ and a space requirement of $O(n)$.

3.1.2 Upper Bound Inequalities:

Let $I = \{1,\ldots,n\}$ and $I_j = I \setminus \{j\}$ $\forall$ $j \in I$. For $p \in \{1,\ldots,n\text{-}1\}$, define $TD_{jp} = \max_{\substack{L \subset I_j \\ |L| = p}} \sum_{l \in L} d_{jl}$. With this definition, TD_{jp} gives the sum of the largest p distances from location j.

Theorem 5: The inequalities $\sum_{k\in J} t_{ik} \le \sum_{j=1}^{n} TD_{j|J|} x_{ij}, \forall i, J$ are valid for IP8, where $J \subseteq I_i$.

Proof: Assume the contrary. Then, there exists a feasible solution (x^*, t^*), for which there exists a facility i and a set $J \subseteq I_i$, such that $\sum_{k\in J} t^*_{ik} > \sum_{j=1}^{n} TD_{j|J|} x^*_{ij}$. Let $x^*_{ij'} = 1$, and let L be the set of locations assigned to the set of facilities J. Note that $|J| = |L|$. Then $\sum_{k\in J} t^*_{ik} = \sum_{l\in L} d_{j'l} \le TD_{j'|J|}$, contradicting the assumed violation. □

Violated upper bound inequalities can be identified by sorting and storing t^*_{ik} values for all i and k ($O(n^2 logn)$), computing and storing $\sum_{j=1}^{n} TD_{j|J|} x^*_{ij}$ for all i and $|J|$ ($O(n^3)$), and comparing the sum of minimum $|J|$ t^*_{ik} values with $\sum_{j=1}^{n} TD_{j|J|} x^*_{ij}$ for all i and $|J|$ ($O(n^2)$), at a total cost of $O(n^3)$ time and $O(n^2)$ space.

3.1.3 Constructed Inequalities:

The valid inequalities we have given so far have exploited certain properties of the distance matrix. We now switch attention to arbitrary distance (and flow) matrices. Suppose that we want to construct a valid inequality of the form

$$t_{ik} \le \sum_{j=1}^{n} \alpha_j x_{ij} + \sum_{l=1}^{n} \beta_l x_{kl} \qquad (60)$$

where α_j, β_l are constants to be determined.

Theorem 6: If α_j and β_l obey the constraints $\alpha_j + \beta_l \geq d_{jl}, \forall j,l : j \neq l$, then the constraint set $t_{ik} \leq \sum_{j=1}^{n} \alpha_j x_{ij} + \sum_{l=1}^{n} \beta_l x_{kl}, \forall i,k$ is valid for IP8.

Proof: Assume the contrary. Then, α_j and β_l obey the constraints $\alpha_j + \beta_l \geq d_{jl}, \forall j,l : j \neq l$, but there exists a solution (x^*,t^*) of IP8 such that $t^*_{ik} > \sum_{j=1}^{n} \alpha_j x^*_{ij} + \sum_{l=1}^{n} \beta_l x^*_{kl}, \forall i,k$ for some *i* and *k*, where $i \neq k$. Let $x^*_{ij} = 1$ and $x^*_{kl} = 1$, where $j \neq l$. Then $t^*_{ik} = d_{jl} > \sum_{j=1}^{n} \alpha_j x^*_{ij} + \sum_{l=1}^{n} \beta_l x^*_{kl} = \alpha_j + \beta_l \geq d_{jl}$, which is a contradiction. □

For a given fractional solution (x^*,t^*) to IP8, a most violated valid inequality can be computed by solving the following linear program (LP).

(LP1)

$$z^*_{LP1} = \max t^*_{ik} - \sum_{j=1}^{n} \alpha_j x^*_{ij} - \sum_{l=1}^{n} \beta_l x^*_{kl} \tag{61}$$

$$\text{s.t.} \quad \alpha_j + \beta_l \geq d_{jl}, \forall j,l : j \neq l \tag{62}$$

All violated inequalities of the form (60) can be identified by solving LP1 $\binom{n}{2}$ times, with corresponding objective function coefficients. A violated inequality is found if $z^*_{LP1} > 0$. LP1 consists of $2n$ variables and $2\binom{n}{2}$ constraints, and provides an upper bound on the distance between two facilities. A total of $\binom{n}{2}$ instances must be solved for complete identification. The idea may be generalized to impose bounds on the sum of distances between *p*

facilities, where $\binom{n}{p}$ instances of similar linear programs with pn variables and $p!\binom{n}{p}$ constraints must be solved. In our implementation, we have resorted to a heuristic way of identification to avoid solving exponentially many linear programs. Details of the heuristic are given below.

For $p = 2$, we solve a single instance of LP1 for each facility i, with $-x^*_{ij}$ as the cost coefficient for α_j, and $-(1- x^*_{ij})/(n-1)$ as the cost coefficient for β_j. Cost coefficients of β describe an imaginary facility which is partially assigned to every possible location in a way that does not contradict facility i. The optimum solution (α^*,β^*) of this particular instance, in a sense, gives the best linearization for the current assignment of locations to facility i. We apply this valid inequality to facility pairs (i,k) $\forall k \neq i$. Valid inequalities constructed in this way require solving n instances of LP1 (instead of $\binom{n}{2}$). Violated valid inequalities of this type can be identified by computing and storing $\sum_{j=1}^{n} \alpha_j x^*_{ij}$ for each i and $\sum_{l=1}^{n} \beta_l x^*_{kl}$ for $i \neq k$ ($O(n^2)$), and comparing $t_{ik}{}^*$ with $\sum_{j=1}^{n} \alpha_j x^*_{ij} + \sum_{l=1}^{n} \beta_l x^*_{kl}$ ($O(n^3)$), resulting in a time bound of $O(n^3)$ and a space requirement of $O(n)$.

Note that, to decrease the computational burden of solving linear programs, one may assume equality of variables with the same subscript (i.e. $\alpha_j = \beta_j$ $\forall k$), which decreases the number of variables by $1/p$ and the number of constraints by $1/(p!)$. The resulting valid inequalities constructed by solving these reduced linear programs are slightly weaker. For $p = 3$, we solve a single linear program with this assumption, and with all objective function coefficients being equal to $-1/n$. This way, we compute a single set of coefficients that give the best possible linearization for the case when assignment variables are equally divided.

Inequalities constructed in this way can be identified by computing and storing $\sum_{j=1}^{n} \alpha_j x_{ij}^*$ for all i ($O(n^2)$), and comparing the sum of distances between every three facility (i,k,m) (i.e. $t_{ik}^* + t_{ki}^* + t_{im}^* + t_{mi}^* + t_{km}^* + t_{mk}^*$) with $\sum_{j=1}^{n} \alpha_j x_{aj}^* + \sum_{j=1}^{n} \alpha_j x_{bj}^* + \sum_{j=1}^{n} \alpha_j x_{cj}^*$ ($O(n^3)$), resulting in a time bound of $O(n^3)$ and a space requirement of $O(n)$. For $p > 3$, construction and identification processes become computationally prohibitive. Hence, we have disregarded valid inequalities corresponding to $p > 3$.

The valid inequalities presented in this section impose upper bounds on linear combinations of the decision variables. Variants of valid inequalities, in which the same combinations are bounded below, can be similarly constructed. As a final note, we note that, in terms of improving the optimum LP relaxation value, the most effective one among the proposed valid inequalities we have presented is the one that uses the triangle inequalities.

3.2 Computational Results:

We have implemented a depth-first branch-and-cut algorithm using IP8 and the valid inequalities presented in Section 2.1. A flow diagram of the algorithm is given in Figure 16.

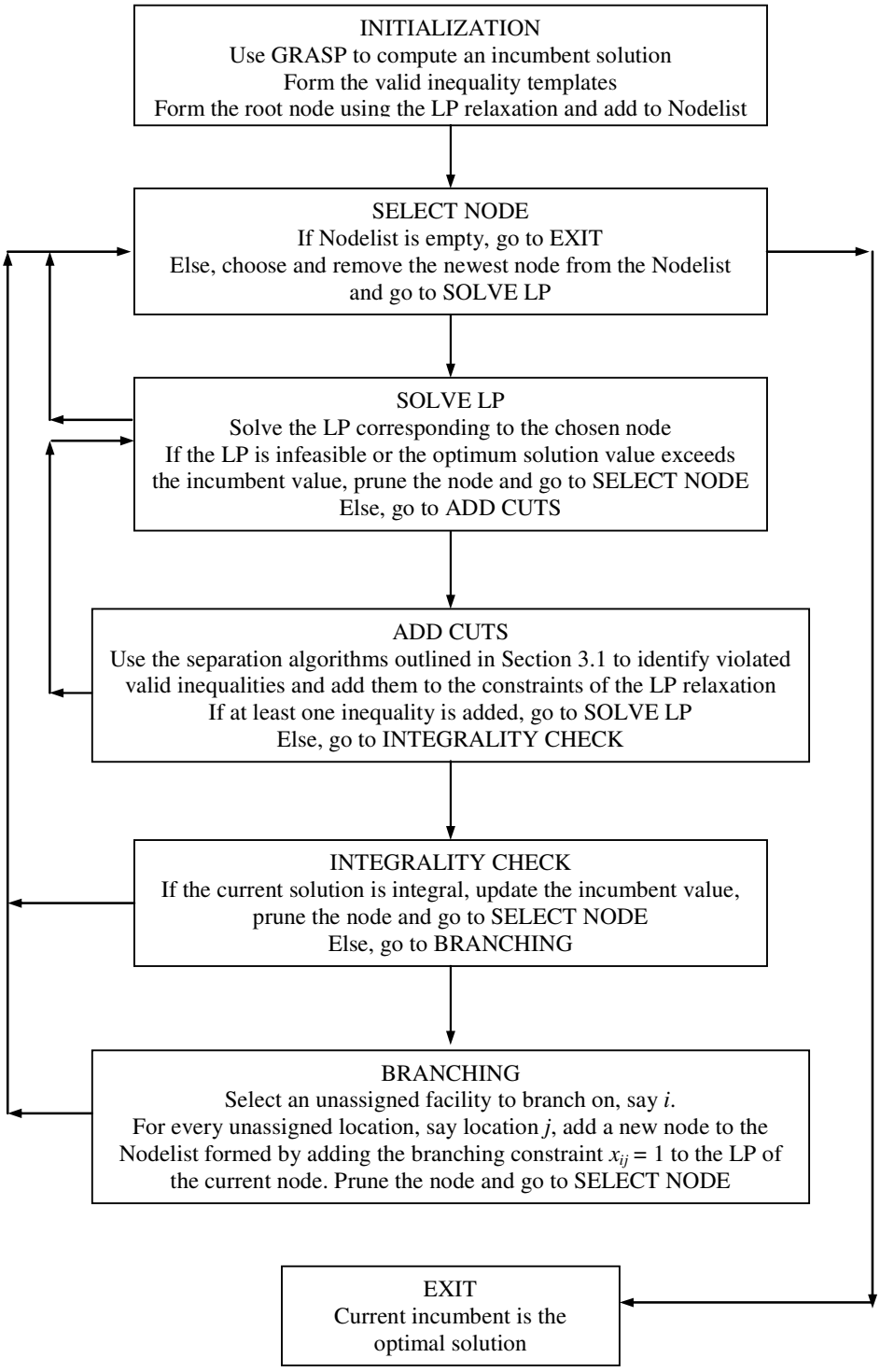

Figure 16: Flow Diagram for the Proposed Branch-and-Cut Algorithm

For instances with symmetric distance matrices, we have made use of the facts $d_{jl} = d_{lj}\ \forall j,l$ and $d_{jj} = 0\ \forall j$ that imply $t_{ik} = t_{ki}\ \forall i,k$, and $t_{ii} = 0\ \forall i$, to decrease the number of t variables by more than one half. Recall that t_{ij} represents the induced distance between facilities i and j and the nonlinear representation of t_{ij} in terms of the assignment variables is $t_{ik} = \sum_{j,l=1}^{n} d_{jl} x_{ij} x_{kl}$. If we were able to completely linearize the objective function, the equality above would hold for all i, k for the solution of the LP relaxation of IP8. When that is not the case, we can compute the violation of t_{ik}^* as $v_{ik}^* = t_{ik}^* - \sum_{j,l=1}^{n} d_{jl} x_{ij}^* x_{kl}^*$. Notice that v_{ik} approximates the error of linearization, and that the error becomes more severe as $|v_{ik}|$ increases. Similarly, the error of the distance between facilities i and k becomes more important as the amount of flow between the facilities deviates from the average flow. We compute the value $a_{ik} = |v_{ik}|*(|f_{ik} - \text{avg}(F)| + \text{avg}(F))$ for all i, k, where avg(F) denotes the average flow, as an indicator of the importance of linearization.

All distances from and to facility i are completely linearized, as soon as facility i is assigned to some location. Consequently, we select as our branching rule an unassigned facility i with the largest sum of $\sum_{k=1}^{n} a_{ik}$. We use *row branching* where child problems are formed by assigning an unassigned facility to all unassigned locations (Anstreicher et al., 2002).

For computational testing of our algorithm, we used problem instances available from the QAPLIB. We have attempted to solve all sets of problems with the exception of the two sets of data supplied by Eschermann and Wunderlich (1990), and Li and Pardalos (1992). The former set involves a high level of symmetry in both flow and distance matrices, which renders branching

efforts fruitless, and is unsuitable for computational testing. The latter set consists of asymmetric instances with known optimal solutions. This set of problems is not used in the literature for computational testing, hence we leave it out due to lack of a basis of comparison with other methods. In addition to the instances from the QAPLIB, we have created 5 new instances of sizes 22, 24, 26, 28, and 30, referred to as `erd22`, `erd24`, `erd26`, `erd28`, `erd30`, respectively, in exactly the same way as the instances `hadxx` are created.

For instances representing symmetric grid graphs, namely `nugxx` and `scrxx`, we have implemented a symmetry test proposed by Mautor and Roucairol (1994) that aims to reduce the number of subproblems at each node of the branch-and-cut tree by identifying sets of symmetric locations and branching on a single element from each set. Even though the symmetry test of Mautor and Roucairol is not generally valid for all distance matrices, it is known to be valid for grid graphs with Manhattan metric. This class includes instances `nugxx` and `scrxx`. The test proved to be very useful, effectively decreasing the CPU time to one quarter of the original or less.

We have used GRASP (Li and Resende, 1994) with 10000 restarts as the startup heuristic. In 89% of the instances, GRASP found the optimum solution. For the cases when the initial incumbent was not optimal, the gap between the optimal solution and the incumbent was at most 2.3%. The application of GRASP did not take more than a few minutes for any of the instances, so we report only the CPU times for the branch-and-cut algorithm. CPLEX 8.11 LP solver was used for optimizing the resulting linear programs. The runs were conducted on a single PC (1133 Mhz Dell PowerEdge with 256MB RAM). Memory requirement was not more than 15MB for even the largest instances. The constraint (58) was removed from the formulation and added to the valid inequality pool, to benefit from a smaller ($O(n)$) static constraint set. As

empirical proof of the strength of the valid inequalities we have proposed we give, in Table 2, the number of cuts added at the root node and the effect on the optimum relaxation value. The separated values in columns 2,3, and 4 are the objective values corresponding, respectively, to the initial relaxation, final relaxation, and the optimal solution of IP8.

Table 2: Effect of Valid Inequalities on Lower Bound

Data File	*Number of Cuts Added*	*Initial Relaxation Value*	*Final Relaxation Value*	*Optimum Solution Value*	*Initial Gap*	*Final Gap*	*Data Type*
bur26a	737	5321819.20	5333986.45	5426670.00	1.93%	1.71%	General
bur26b	784	3725027.10	3748891.39	3817852.00	2.43%	1.81%	General
bur26c	741	5320676.00	5333544.83	5426795.00	1.96%	1.72%	General
bur26d	790	3725693.80	3747069.19	3821225.00	2.50%	1.94%	General
bur26e	687	5310894.40	5318727.83	5386879.00	1.41%	1.27%	General
bur26f	774	3713578.20	3732885.17	3782044.00	1.81%	1.30%	General
bur26g	677	9991175.90	10010200.17	10117172.00	1.25%	1.06%	General
bur26h	793	6994747.10	7030401.88	7098658.00	1.46%	0.96%	General
chr12a	53	8448.70	8840.10	9552.00	11.55%	7.45%	General
chr12b	57	7298.40	8966.50	9742.00	25.08%	7.96%	General
chr12c	46	9784.40	9909.80	11156.00	12.29%	11.17%	General
chr15a	85	7607.70	8084.00	9896.00	23.12%	18.31%	General
chr15b	93	5063.20	6127.20	7990.00	36.63%	23.31%	General
chr15c	70	8823.00	9129.70	9504.00	7.17%	3.94%	General
chr18a	104	9014.60	9611.70	11098.00	18.77%	13.39%	General
chr18b	0	1534.00	1534.00	1534.00	0.00%	0.00%	General
chr20a	216	2156.00	2156.00	2192.00	1.64%	1.64%	General
chr20b	134	2236.90	2241.70	2298.00	2.66%	2.45%	General
chr20c	159	9134.40	12029.50	14142.00	35.41%	14.94%	General
chr22a	177	5952.90	6021.70	6156.00	3.30%	2.18%	General
chr22b	159	6018.70	6066.50	6194.00	2.83%	2.06%	General
chr25a	174	3136.70	3328.10	3796.00	17.37%	12.33%	General
els19	234	6090771.50	15822114.20	17212548.00	64.61%	8.08%	General
erd22	369	7780.30	8556.90	8608.00	9.62%	0.59%	Partial Grid
erd24	429	9486.60	10511.80	10596.00	10.47%	0.79%	Partial Grid
erd26	439	10859.00	12102.00	12222.00	11.15%	0.98%	Partial Grid
erd28	486	13661.70	15185.80	15334.00	10.91%	0.97%	Partial Grid

erd30	554	17188.20	19070.00	19238.00	10.65%	0.87%	Partial Grid
had12	104	1568.00	1640.30	1652.00	5.08%	0.71%	Partial Grid
had14	137	2536.40	2710.50	2724.00	6.89%	0.50%	Partial Grid
had16	193	3392.70	3690.70	3720.00	8.80%	0.79%	Partial Grid
had18	230	4853.60	5287.00	5358.00	9.41%	1.33%	Partial Grid
had20	284	6290.20	6852.80	6922.00	9.13%	1.00%	Partial Grid
nug12	133	457.50	548.70	578.00	20.85%	5.07%	Grid
nug14	166	815.80	966.90	1014.00	19.55%	4.64%	Grid
nug15	168	910.10	1099.40	1150.00	20.86%	4.40%	Grid
nug16a	192	1257.00	1534.90	1610.00	21.93%	4.66%	Grid
nug16b	164	914.00	1195.00	1240.00	26.29%	3.63%	Grid
nug17	222	1293.70	1627.70	1732.00	25.31%	6.02%	Grid
nug18	241	1436.60	1808.20	1930.00	25.56%	6.31%	Grid
nug20	268	1851.70	2416.70	2570.00	27.95%	5.96%	Grid
nug21	346	1673.50	2270.90	2438.00	31.36%	6.85%	Grid
nug22	319	2256.80	3395.20	3596.00	37.24%	5.58%	Grid
nug24	418	2255.50	3272.90	3488.00	35.34%	6.17%	Grid
nug25	443	2504.00	3498.30	3744.00	33.12%	6.56%	Grid
nug27	447	3326.30	4896.34	5236.00	36.47%	6.49%	Grid
nug28	604	3168.10	4815.96	5166.00	38.67%	6.78%	Grid
nug30	615	3745.20	5693.97	6124.00	38.84%	7.02%	Grid
rou12	121	161123.80	211372.80	235528.00	31.59%	10.26%	General
rou15	211	222087.20	307352.50	354210.00	37.30%	13.23%	General
scr12	114	26152.00	30793.60	31410.00	16.74%	1.96%	Grid
scr15	221	39898.20	50222.50	51140.00	21.98%	1.79%	Grid
scr20	329	75420.00	98995.80	110030.00	31.46%	10.03%	Grid
tai12a	123	158216.70	207774.30	224416.00	29.50%	7.42%	General
tai12b	105	11426178.00	36789487.70	39464925.00	71.05%	6.78%	General
tai15a	205	244239.70	340869.50	388214.00	37.09%	12.20%	General
tai15b	187	50094649.20	51485572.30	51765268.00	3.23%	0.54%	General

In some cases, especially when the optimality gap of the initial relaxation is large, the valid inequalities are very effective. For example, for els19 the gap is reduced from 64.61% to 8.08%, and for tai12b, the gap is reduced from 71.05% to 6.78%. In cases when the optimality gap is smaller, varying effects of 0.1 to 20 percent is observed. When the distance matrix represents a grid or a partial grid (namely for the instances erdxx, hadxx, nugxx, and scrxx) the

effect is stronger. We note that no more than a few hundred inequalities are required even for the largest instances, after removing redundant inequalities. Details of the branch-and-cut applied to the same problems are given in Table 3.

Table 3: Problems Solved to Optimality by Branch-and-Cut

Data file	*Problem Size*	*Number of Nodes*	*CPU Time (sec)*	*Initial Incumbent Value*	*Optimum solution value*	*Data Type*
bur26e	26	59761	81731.28	5386879.00	5386879.00	General
bur26f	26	10439	25154.93	3782044.00	3782044.00	General
bur26g	26	118782	98666.64	10117172.00	10117172.00	General
bur26h	26	4618	10709.26	7098658.00	7098658.00	General
chr12a	12	34	0.96	9552.00	9552.00	General
chr12b	12	12	0.35	9742.00	9742.00	General
chr12c	12	89	2.25	11156.00	11156.00	General
chr15a	15	321	15.98	9896.00	9896.00	General
chr15b	15	165	9.19	7990.00	7990.00	General
chr15c	15	15	1.57	9504.00	9504.00	General
chr18a	18	196	18.91	11098.00	11098.00	General
chr18b	18	0	0.01	1534.00	1534.00	General
chr20a	20	1152	264.32	2192.00	2192.00	General
chr20b	20	1198	259.53	2352.00	2298.00	General
chr20c	20	469	96.77	14142.00	14142.00	General
chr22a	22	2141	394.57	6266.00	6156.00	General
chr22b	22	3340	617.37	6314.00	6194.00	General
chr25a	25	9885	2941.85	4250.00	3796.00	General
els19	19	161	66.29	17212548.00	17212548.00	General
erd22	22	393	199.75	8608.00	8608.00	Partial Grid
erd24	24	3634	1946.51	10596.00	10596.00	Partial Grid
erd26	26	19759	14978.78	12222.00	12222.00	Partial Grid
erd28	28	74923	83504.84	15334.00	15334.00	Partial Grid
erd30	30	90444	127158.06	19238.00	19238.00	Partial Grid
had12	12	12	0.85	1652.00	1652.00	Partial Grid
had14	14	27	1.83	2724.00	2724.00	Partial Grid
had16	16	76	10.06	3720.00	3720.00	Partial Grid
had18	18	717	110.44	5358.00	5358.00	Partial Grid
had20	20	743	156.32	6922.00	6922.00	Partial Grid
nug12	12	43	3.17	578.00	578.00	Grid

nug14	14	747	49.82	1014.00	1014.00	Grid
nug15	15	315	27.45	1150.00	1150.00	Grid
nug16a	16	1856	185.45	1610.00	1610.00	Grid
nug16b	16	334	43.20	1240.00	1240.00	Grid
nug17	17	7652	1020.04	1732.00	1732.00	Grid
nug18	18	20353	3551.20	1930.00	1930.00	Grid
nug20	20	50862	13859.07	2570.00	2570.00	Grid
nug21	21	18537	6019.89	2438.00	2438.00	Grid
nug22	22	10370	3314.30	3596.00	3596.00	Grid
nug24	24	322443	208288.91	3488.00	3488.00	Grid
rou12	12	392	46.98	235528.00	235528.00	General
rou15	15	23469	7248.02	354210.00	354210.00	General
scr12	12	22	1.38	31410.00	31410.00	Grid
scr15	15	23	5.95	51140.00	51140.00	Grid
scr20	20	5161	1467.80	110030.00	110030.00	Grid
tai12a	12	132	15.74	224416.00	224416.00	General
tai12b	12	157	6.82	39464925.00	39464925.00	General
tai15a	15	56406	16490.41	388214.00	388214.00	General
tai15b	15	402	116.65	51765268.00	51765268.00	General

We solved all instances of chrxx, including chr25a, in less than one hour of CPU time. For this set of instances only, we have used IP5' instead of IP8, since these instances have a special flow matrix that represents a tree, which can be exploited. All instances of hadxx, for which the distance matrix represents a partial grid network (a subgraph of a grid), are solved within 3 minutes of CPU time. The instance els19, which consists of real world data and exhibits a distance pattern that is quite close to being planar, is solved in about one minute of CPU time. The instances nugxx, especially nug20 and nug24 proved to be harder. For example, nug20 took about 4 hours of CPU time and nug24 took approximately 58 hours of CPU time. Notably, computing time requirement decreased for instances nug20, nug21, and nug22 as the size of the problem increased. This is mainly because of the decreasing level of symmetry in 4*5, 3*7, and 2*11 grids. The instances scrxx were solved relatively easily due to the erratic structure of the flow matrix, which is sparse and helps us to branch efficiently. The instances rouxx and taixx, being randomly generated and

exhibiting no discernible patterns, are the hardest of all. We solved rou15 in close to 2 hours of CPU time and tai15a in about 4.5 hours of CPU time, while sizes of n > 15 exceed computational reach for these two classes of instances.

The instances erdxx were solved quite easily with respect to their size. The largest of them, erd30, took about one and a half days to complete. The computational success mainly depends on the structure of the distance and flow matrices; the former is the shortest distance matrix of a partial grid, and the latter is uniformly drawn from the interval [1,*n*]. This structure, in turn, yields a strong lower bound and results in a small branch-and-cut tree.

There are also instances, not reported in Table 3, that are attempted but not solved to optimality. For example, branch-and-cut trees of the instances bur26a-d and nug25-30 could not be entirely fathomed. For these problems, we have imposed a depth limit of 3 and fathomed the reduced trees to collect further data about the strength of the lower bound at the lower nodes of the tree. Instances bur26a-d were not solved despite the fact that instances bur26e-h, which have the same distance matrix but different flow matrices, were solved relatively easily. This suggests that our branching rule performs better for instances bur26e-h, since the branching rule is the only part that depends on the flow matrix. Instances nug25-30 do not yield good lower bounds even in the lower branches due to the high level of symmetry in the distance matrix. These instances simply require more computing power. The data for suboptimally solved problems is summarized in Table 4.

Table 4: Suboptimally Solved Problems

Data File	*Gap Depth: 0*	*Depth: 1*	*Depth: 2*	*Depth: 3*	*CPU Time (sec)*
bur26a	1.71%	1.37%	1.18%	0.91%	75711.46
bur26b	1.81%	1.49%	1.18%	0.76%	74572.85
bur26c	1.72%	1.47%	1.26%	1.02%	132849.90
bur26d	1.94%	1.61%	1.08%	0.75%	59443.98
nug25	6.56%	6.22%	5.56%	4.20%	16267.40
nug27	6.49%	5.85%	4.68%	3.90%	43729.22
nug28	6.78%	6.37%	5.50%	4.42%	66432.41
nug30	7.02%	6.82%	6.15%	5.04%	130163.14

Even though the instances nugxx are considered to be a benchmark, drastic improvements in computing hardware and inherent differences between sequential and parallel implementations increase the difficulty of comparison. For example, nug20 required 811440.0 CPU seconds of a state of the art computer in 1994, when it was solved for the first time by Clausen and Perregaard (1997). Our method required 13859.07 CPU seconds for the same instance on a PC, but it should be noted that the computing technology has doubled the speed of computation a few times in the past decade. To date, the most successful study in terms of dealing with the instances nugxx has been that of Anstreicher et al. (2002), which is a parallel branch-and-bound implementation that uses the bound presented in the study by Brixius and Anstreicher (2001). Since our implementation is sequential, we compare our results with the results presented in the latter paper. The authors report solution times for the instances nug16b, nug18, nug20, nug21, nug22, and nug24 as CPU minutes of a HP9000 C3000 workstation whereas our solution times are for a single PC (1133 Mhz Dell PowerEdge with 256MB RAM). The CPU's under consideration are different in terms of architecture and speed. For an accurate comparison, we have used the results of a benchmarking study of Guest (2005) to scale the run times. Relative to the benchmark computer, our system is cited to have 19% CPU performance for floating point operations, whereas

HP9000 C3000 is cited to have 17% performance. Hence, we have multiplied our CPU times by 19/17 to have a scaled comparison.

Table 5: Comparison of scaled CPU times (in minutes) for instances `nugxx`

Data File	*Erdogan and Tansel*	*Brixius and Anstreicher*
`nug16b`	0.80	0.90
`nug18`	66.15	69.20
`nug20`	258.15	145.80
`nug21`	112.13	212.30
`nug22`	61.74	134.30
`nug24`	3879.89	5829.90

Table 5 gives the comparison of the two studies in terms of CPU time requirements. While Brixius and Anstreicher can solve `nug20` with greater efficiency, our method performs better for the larger instances `nug21`, `nug22`, and `nug24`. Note that the CPU time requirement of our method is no more than 65% of the method of Brixius and Anstreicher for these instances. Our personal communication with Anstreicher (2005) revealed that, such benchmarking studies generally produce approximations within a 30% error margin. Even allowing a 30% error margin for the results of the benchmarking study, Table 5 gives us reason to claim that our method can compete with the state of the art methods in the literature.

To have a better understanding of the performance of the branch-and-cut algorithm, we have disabled the GRASP heuristic for a few instances that seem to be representative of their corresponding problem sets, and analyzed the change in CPU time and number of nodes traversed. The results are given in Table 6.

Table 6: Run Times with Different Initial Upper Bound Values

Data file	*ub: infinity*		*ub:GRASP*	
	Number of Nodes	*CPU Time (sec.)*	*Number of Nodes*	*CPU Time (sec.)*
chr12a	67	1.69	34	0.96
had12	45	1.77	12	0.85
nug12	53	3.39	43	3.17
rou12	1066	99.13	392	46.98
scr12	82	3.34	22	1.38
chr15a	381	20.61	321	15.98
had16	166	19.45	76	10.06
nug15	647	51.41	315	27.45
rou15	35803	10618.13	23469	7248.02
scr15	184	16.59	23	5.95
chr20a	7197	1253.72	1152	264.32
had20	12877	2660.57	743	156.32
nug20	86399	21874.33	50862	13859.07
scr20	8163	2202.93	5161	1467.80

As expected, CPU times are longer when the initial upper bound value is set to infinity as compared to the case where the initial upper bound is supplied by the GRASP heuristic. However, characteristics of each instance dictate the order and quality of the integral solutions found by the branch-and-cut algorithm. Hence, the results are somewhat erratic, even among the members of the same instance set. For example, while chr15a suffers a 29% increase in CPU time and 19% increase in the number of nodes traversed, CPU time required by the instance chr20a increases more than 4 times and the number of nodes traversed increases more than 6 times when the initial upper bound is set to infinity. In contrast, scr20 performs much better than scr15, resulting in 50% increase in CPU time versus a 179% increase. We emphasize the fact that the algorithm we have presented is designed to prove optimality rather than to find high quality solutions and depends heavily on the quality of the initial solution. In fact, the quality of the initial solutions supplied by the GRASP heuristic encouraged us to use the depth-first node selection rule. A more robust branch-and-cut algorithm

may be implemented by switching the node selection rule to breadth-first or best-first.

To better understand the cases in which our algorithm performs best, we have computed the standard complexity measure for the QAP, namely flow dominance and distance dominance (Vollmann and Buffa, 1966) of the instances we have attempted to solve. We have also included some metrics (indices) that we have devised. Namely, we have computed the ratio of number of feasible solutions for which the valid inequalities are binding to the total number of feasible solutions. The results are given in Table 7.

Table 7:Indices Computed for the Instances from QAPLIB

Data file	*Flow Dominance*	*Distance Dominance*	*Distance Lower Bound Index*	*Distance Upper Bound Index*	*Triangle Sum Lower Bound Index*	*Triangle Sum Upper Bound Index*	*Triangle Diff. Lower Bound Index*	*Triangle Diff. Upper Bound Index*
bur26a	274.744	15.074	0.197	0.614	0.232	0.074	0.087	0.093
bur26b	274.744	15.901	0.262	0.614	0.302	0.068	0.259	0.093
bur26c	228.227	15.074	0.245	0.614	0.232	0.074	0.087	0.093
bur26d	228.227	15.901	0.226	0.614	0.302	0.068	0.259	0.093
bur26e	253.807	15.074	0.191	0.629	0.232	0.074	0.087	0.093
bur26f	253.807	15.901	0.235	0.726	0.302	0.068	0.259	0.093
bur26g	279.687	15.074	0.215	0.614	0.232	0.074	0.087	0.093
bur26h	279.687	15.901	0.257	0.614	0.302	0.068	0.259	0.093
chr12a	63.206	307.980	0.182	0.833	0.564	0.155	0.071	0.018
chr12b	63.206	307.980	0.485	0.833	0.605	0.055	0.092	0.018
chr12c	63.206	307.980	0.250	0.833	0.545	0.064	0.042	0.018
chr15a	69.735	326.951	0.262	0.867	0.635	0.053	0.051	0.011
chr15b	69.735	326.951	0.433	0.867	0.651	0.095	0.073	0.012
chr15c	69.735	326.951	0.143	0.867	0.629	0.037	0.027	0.012
chr18a	63.098	350.595	0.248	0.889	0.692	0.088	0.046	0.007
chr18b	56.863	356.319	0.176	0.889	0.686	0.032	0.019	0.007
chr20a	59.385	345.940	0.197	0.900	0.723	0.040	0.038	0.006
chr20b	59.385	345.940	0.124	0.900	0.716	0.036	0.015	0.006
chr20c	65.630	345.940	0.447	0.900	0.736	0.335	0.059	0.006
chr22a	66.887	420.620	0.273	0.909	0.747	0.057	0.045	0.005

chr22b	66.887	420.620	0.199	0.909	0.742	0.034	0.024	0.005
chr25a	57.925	423.928	0.267	0.920	0.776	0.098	0.041	0.006
els19	530.281	52.030	0.164	0.129	0.022	0.029	0.007	0.014
erd22	45.955	64.209	0.541	0.216	0.038	0.362	0.009	0.267
erd24	46.502	63.699	0.554	0.217	0.033	0.313	0.017	0.270
erd26	45.756	61.711	0.502	0.209	0.030	0.291	0.010	0.253
erd28	44.751	62.042	0.545	0.196	0.021	0.290	0.015	0.254
erd30	44.053	63.933	0.547	0.182	0.023	0.284	0.008	0.260
had12	50.679	63.130	0.682	0.364	0.109	0.418	0.018	0.333
had14	49.456	66.622	0.560	0.297	0.071	0.451	0.013	0.328
had16	48.403	64.829	0.542	0.292	0.050	0.454	0.014	0.300
had18	47.132	63.681	0.542	0.255	0.048	0.397	0.012	0.272
had20	45.957	64.243	0.553	0.221	0.042	0.385	0.006	0.277
nug12	116.580	56.891	0.576	0.424	0.155	0.382	0.055	0.255
nug14	103.566	56.749	0.582	0.374	0.113	0.352	0.043	0.244
nug15	106.476	56.582	0.562	0.362	0.101	0.255	0.040	0.245
nug16a	100.737	57.334	0.542	0.325	0.086	0.325	0.020	0.257
nug16b	115.595	54.772	0.500	0.342	0.093	0.371	0.019	0.238
nug17	104.827	56.259	0.522	0.324	0.078	0.344	0.021	0.236
nug18	104.211	54.935	0.529	0.301	0.072	0.348	0.019	0.229
nug20	103.646	54.102	0.489	0.284	0.061	0.309	0.014	0.228
nug21	117.061	57.385	0.514	0.267	0.053	0.264	0.018	0.235
nug22	114.216	64.086	0.459	0.216	0.038	0.291	0.010	0.262
nug24	112.783	54.130	0.457	0.243	0.043	0.261	0.008	0.221
nug25	110.763	53.033	0.480	0.240	0.041	0.257	0.010	0.217
nug27	111.402	58.614	0.487	0.211	0.032	0.194	0.010	0.229
nug28	112.999	54.499	0.450	0.212	0.032	0.260	0.007	0.217
nug30	112.417	52.725	0.448	0.202	0.029	0.227	0.006	0.210
rou12	67.053	71.538	0.182	0.182	0.055	0.055	0.018	0.018
rou15	68.739	69.073	0.148	0.143	0.033	0.035	0.011	0.011
rou20	65.569	64.352	0.105	0.105	0.018	0.018	0.006	0.006
scr12	256.487	56.891	0.576	0.424	0.155	0.382	0.055	0.255
scr15	247.750	54.921	0.533	0.371	0.103	0.404	0.040	0.236
scr20	254.333	54.102	0.489	0.284	0.061	0.309	0.014	0.228
tai12a	74.765	69.307	0.182	0.174	0.055	0.055	0.021	0.018
tai12b	299.606	79.211	0.212	0.189	0.055	0.082	0.024	0.045
tai15a	70.563	63.777	0.152	0.143	0.035	0.033	0.012	0.013
tai15b	312.935	262.313	0.324	0.176	0.035	0.068	0.015	0.092

The column labeled "Distance Upper Bound Index" denotes the ratio of the number of strict inequalities to the total number of inequalities in the optimum solution of LP1, when solved to optimality with objective function coefficients of $-1/n$. This index value is a measure of the strength of the constructed inequalities for $m = 2$, and can be easily computed by solving a single instance of LP1. Remember that the constructed inequalities for $m = 3$ consist of a single set of coefficients that is applied to all facility triplets. Consider any three facilities and all possible location assignments to these facilities, and count the cases for which the constructed inequalities are strict. The column labeled "Triangle Sum Upper Bound Index" denotes the ratio of this count to the total number of assignments ($\binom{n}{3}$). In other words, this index value measures the strength of the constructed valid inequalities for $p = 3$. This index can be computed by solving a single linear program with n variables and $\binom{n}{3}$ constraints, and executing the counting process ($O(n^3)$). Finally, the column labeled "Triangle Diff. Upper Bound Index" denotes the ratio of the number of location triplets for which the triangle inequalities are strict to the total number of triplets. Computation of this index requires $O(n^3)$ time. The rest of the columns consist of the indices for the lower bound counterparts of the same valid inequalities. Note that, a higher value means a stronger effect, but the values across the columns are not comparable, since the corresponding valid inequalities differ in strength.

Notice that for the instances `chrxx` the values of Distance Upper Bound Index and Triangle Sum Lower Bound Index are very high, justifying the ease of solution for these instances. For the instances `erdxx` and `hadxx`, the values of the indices for Distance Lower Bound, Triangle Sum Upper Bound, and Triangle Difference Upper Bound are notably high. Although the same indices are remarkably high for the instances `nugxx` and `scrxx`, the symmetry factor comes into play and increases the level of difficulty. Instances `burxx` exhibit

high values for Distance Upper Bound Index and Triangle Sum Lower Bound Index. Consequently, the lower bounds generated at the root node are close to the optimum solution value. Instances `rouxx` and `taixx` do not exhibit any high values for any of the indices, and hence, are the hardest of all.

The flow and distance dominance values do not mean much without the rest of the data. For example, the instances `chrxx` exhibit large distance dominance values, the instances `scrxx` exhibit large flow dominance values, and finally the instances `erdxx` exhibit low values for both and distance dominance. All three sets of instances have been solved with reasonable efficiency, so the dominance values seem irrelevant. However, further analysis of the instances `scr12`, `scr15`, `scr20` and `nug12`, `nug15`, `nug20` that have the same distance matrices reveals that the `scrxx` instances are solved with greater efficiency. The only apparent reason for this is the higher flow dominance value of these instances. Likewise the instance `els19`, which does not yield high values for any of the indices but has the highest flow dominance value, is solved efficiently. We find it appropriate to conclude that a higher value of flow dominance helps our branching strategy to find the decisions that are more important than the others.

We can conclude that our method performs best when one or more of the following occur:

1) A value of 0.5 or higher for at least one of Distance Upper Bound and Distance Lower Bound indices.
2) A value of 0.3 or higher for at least one of Triangle Sum Upper Bound and Triangle Sum Lower Bound indices.
3) A value of 0.3 or higher for at least one of Triangle Diff. Upper Bound and Triangle Diff. Lower Bound indices.
4) A Flow Dominance value of 200 or more.

3.3 Concluding Remarks

In this chapter, we have analyzed the Single Commodity Distance Formulation, presented valid inequalities, and gave the results and details of the implementation of a branch-and-cut algorithm. Our main motivation was to exploit the special structure inherent to the data, by using auxiliary variables that inherit the same structure. We were able to solve certain classes of problems for which certain structures were dominant.

An unforeseen consequence of incorporating the data into the constraint matrix is the need to solve auxiliary problems in order to identify valid inequalities. To be more precise, for the Traveling Salesman Problem, for example, one can logically identify the subtour elimination constraints. However, in our case, we need to solve LPs to identify or to *construct* valid inequalities. Thus, we observed that with a constraint matrix dependent on the problem instance at hand, the act of building more information into the model becomes a problem of its own.

We have tried to analyze the behavior of the algorithm we have presented using different metrics we have devised, as well as metrics from the literature. We have observed that our algorithm performs best when one or more of the proposed metrics is significantly high. Using the formulations and valid inequalities we have presented, we have been able to solve an instance of size 30 that exhibits high values for the metrics we have presented. In contrast, we have failed to solve instances from the randomly created sets of problems that are of size larger than 15.

We have focused on identifying all violated valid inequalities, for the sake of a better analysis. It is our belief that with high performance heuristics to

identify violated valid inequalities, and access to higher amounts of computing power, the proposed formulation may be used to solve larger problem instances.

In the next chapter, we turn our attention to flow, distance, and general cost coefficient matrices with special structure and uncover polynomially solvable classes of instances for the QAP. Before doing so, we repeat some additional insights that we have gained during our computational experimentation.

Additional Insights

- The valid inequalities proposed in this chapter are also applicable to Multicommodity Distance Formulation, by using the fact that $\sum_{j=1}^{n} t_{ijk} = t_{ik}$, where t_{ijk} is the variable definition for the Multicommodity Distance Formulation, and t_{ik} is the variable definition for the Single Commodity Distance Formulation. A brief computational experiment showed that introducing only a few triangle inequalities to the Multicommodity Distance Formulation results in a formulation that exhibits high levels of degeneracy.
- The constraint set (57) is a subset of lower bounding counterpart of valid inequalities (60). Hence (57) can be discarded if all violated lower bounding counterpart of valid inequalities of type (60) are to be identified and added to the formulation.
- The decreasing CPU times for the instances `nug20`, `nug21`, and `nug22` suggest that symmetry plays a great role in determining the CPU time. Developing a more sophisticated and general symmetry test may be a good field of research.
- Two other branching strategies were used, the first one being the traditional 0/1 branching, and the second one being a hybrid of 0/1 branching, row branching, and column branching (where child problems

are formed by assigning an unassigned location to all unassigned facilities). Both rules performed significantly worse than row branching.

- Three other branching rules were carried out, the first one being "branch on the unassigned location with the largest total distance to other unassigned locations", the second one being "branch on the unassigned facility with the largest total flow to other unassigned facilities", and the third one being "branch on the facility or location for which the assignment variables are farthest away from integrality". For the third rule, the value $a_i^1 = \sum_{j=1}^{n} (x_{ij})^2$ was computed for every unassigned facility and $a_j^2 = \sum_{i=1}^{n} (x_{ij})^2$ was computed for every unassigned location. If the maximum value among a_i^1 and a_j^2 is attained by a facility, row branching is used. Else, column branching is used.
- Let Δ_{ik} denote the change in objective function value when the locations of facilities *i* and *k* are exchanged. Denote the induced distance between facilities *i* and *k* as $d_{(i)(k)}$. Then, for an instance with symmetric flow and distance matrices $\Delta_{ik} = \left(\sum_{m \neq i,k} (f_{im} d_{(i)(m)} + f_{km} d_{(k)(m)}) \right) - \left(\sum_{m \neq i,k} (f_{im} d_{(k)(m)} + f_{km} d_{(i)(m)}) \right)$ where the second part stands for the decrease in cost when we "unassign" *i* and *k* and the first part stands for the increase in cost when we "reassign" *i* and *k*. Further derivation yields $\Delta_{ik} = \sum_{m \neq i,k} (d_{(k)(m)} - d_{(i)(m)})(f_{im} - f_{km})$. Obviously, for a solution to be 2-optimal, the set of conditions $\Delta_{ik} \geq 0 : \forall i, k = 1..n, i < k$ is necessary and sufficient. For the formulations with Lawler's variables, $d_{(i)(k)} = \sum_{j,l=1}^{n} d_{jl} y_{ijkl}$. Similarly for IP8, $d_{(i)(k)} = t_{ik}$. Thus, the condition $\Delta_{ik} \geq 0$ can be expressed as a linear inequality and may be used for

reducing the search space. However, experimentation showed that fractional solutions of the subproblems in the branch-and-cut tree do not usually violate these inequalities. Use of the inequalities results in an insignificant decrease in the number of nodes traversed, together with an increase in the run times.

Chapter 4

CLASSES OF POLYNOMIAL TIME SOLVABLE INSTANCES

We have stated in the introduction chapter that the formulations exploiting the binary structure of the QAP involve too many variables to be computationally effective. In the previous chapter we have attempted to exploit the possible structures existing in the flow and distance matrices using the single Commodity Flow and Distance Formulations. In this chapter we take one more step and we restrict our attention to the classes of instances that exhibit special structures that lead to polynomial time solution techniques. In Sections 4.1 and 4.2, we present our findings on instances with additively and multiplicatively decomposable cost coefficients, respectively. In Section 4.3, we identify the class of problems which are partially reducible to LAPs. In Section 4.4 we focus on the graphs associated with the flow and distance matrices. In Section 4.5, we give a result on instances with specially ordered entries. Finally, in Section 4.6, we present our concluding remarks.

4.1 Additive Decomposition

For the Koopmans-Beckmann form with flow and distance matrices $F = [f_{ik}]$ and $D = [d_{jl}]$, Burkard et al. (1997) showed that if $2n$ numbers f_i^r, f_i^c $(i \in \{1,...,n\})$ can be found associated, respectively, with the n rows and the n columns of the flow matrix such

that $f_{ik} = f_i^r + f_k^c \ \forall i,k \in \{1,...,n\}$, the problem is reducible to the LAP. The result is also valid if a similar decomposition is available for *D*.

We now give a significant generalization of this class for the case of general costs C_{ijkl}. The decomposition we propose requires solving a linear system of equations with $O(n^3)$ variables and $O(n^4)$ equations. Because the system is overdetermined, it may or may not have a solution. Whenever it does, the QAP instance on hand is solved as a LAP in polynomial time.

Let $I = \{1,2,\ldots,n\}$ and I^k be the *k*-fold Cartesian product of *I* by itself. Denote by *q* any quadruplet in I^4. We define a quadruplet *ijkl* to be *incompatible* if $(i,j) \neq (k,l)$ and either $i = k$ or $j = l$; and *compatible* otherwise. Incompatible quadruplets correspond to the cases where either two distinct facilities are assigned to the same location or one facility is assigned to two distinct locations. Such assignments are infeasible in the QAP. Define $\bar{I}$ to be the subset of I^4 consisting of all quadruplets in I^4 that are compatible. Note that there are $n^4 - 2n^3 + 2n^2$ compatible quadruplets. We write C_q to mean the cost C_{ijkl} for which $q = ijkl$. For a nonempty subset *S* of {1,2,3,4}, we define $q(S)$ to be the ordered $|S|$-tuple obtained from the quadruplet *q* by retaining the indices in *q* that correspond to positions in *S* while deleting all other indices. For example, if $q = k_1k_2k_3k_4$ and $S = \{1,2,4\}$, then $q(S) = k_1k_2k_4$. If $S = \{2,4\}$ then $q(S) = k_2k_4$. Define also $q(\phi) = \phi \ \forall q \in I^4$.

Corresponding to each proper subset *S* of {1,2,3,4} and each $t \in I^{|S|}$ (if $S = \phi$, take $t = \phi$), define a variable u_t^S. There are $4n^3 + 6n^2 + 4n + 1$ such variables. Let *A* be the $n^4 - 2n^3 + 2n^2$ by $4n^3 + 6n^2 + 4n + 1$ matrix of zeros and ones where the element in row *q* (with *q* being a compatible quadruplet) and column corresponding to the pair (*S*,*t*) (with *S* being a proper subset of {1,2,3,4} and $t \in I^{|s|}$) is denoted by $a_q^{S,t}$. Define $a_q^{S,t} = 1$ if $q(S) = t$ and 0 otherwise. Let $A = [a_q^{S,t}]$

and u be the vector of u_t^S values where the columns of A and the elements of u are assumed to be identically ordered by (S,t). Let C be the vector of costs C_{ijkl}, $ijkl \in I^4$, and $\overline{C}$ be the vector obtained from C by deleting all cost components C_{ijkl} corresponding to incompatible quadruplets $ijkl \in I^4$. We assume the rows of A and the elements of $\overline{C}$ are identically ordered by $q \in \bar{I}$.

Theorem 7: If the linear equality system

$$Au = \overline{C} \tag{63}$$

has a solution, then the instance of the QAP defined by C can be solved as a LAP.

Proof: Assume $\hat{u} = (\hat{u}_t^S)$ solves (63). Then $A\hat{u} = \overline{C}$ implies that

$$\sum_{(S,t):q(S)=t} \hat{u}_t^S = C_q \quad , q \in \bar{I} \tag{64}$$

Using (64), the objective function value of the QAP for any feasible solution $X = (x_{ij})$ can be rewritten as:

$$\sum_{ijkl \in I^4} C_{ijkl} x_{ij} x_{kl} =$$

$$\sum_{ijkl \in \bar{I}} C_{ijkl} x_{ij} x_{kl} = \sum_{ijkl \in \bar{I}} \begin{pmatrix} \hat{u}_{ijk}^{123} + \hat{u}_{ijl}^{124} + \hat{u}_{ikl}^{134} + \hat{u}_{jkl}^{234} + \hat{u}_{ij}^{12} + \hat{u}_{ik}^{13} + \hat{u}_{il}^{14} \\ + \hat{u}_{jk}^{23} + \hat{u}_{jl}^{24} + \hat{u}_{kl}^{34} + \hat{u}_i^1 + \hat{u}_j^2 + \hat{u}_k^3 + \hat{u}_l^4 + \hat{u}_0^0 \end{pmatrix} x_{ij} x_{kl} \tag{65}$$

$$= \sum_{ijkl \in \bar{I}} \hat{u}_{ijk}^{123} x_{ij} x_{kl} + \ldots + \sum_{ijkl \in \bar{I}} \hat{u}_0^0 x_{ij} x_{kl} \tag{66}$$

where the first equality follows from the fact that feasibility ensures $x_{ij} x_{kl} = 0$ for any incompatible quadruplet $ijkl$. Each of the fifteen summations in (66) can be written in such a way as to separate out the omitted index (indices) from u_t^s terms. For example, the first summation gives

$$\sum_{ijkl\in\bar{I}} \hat{u}^{123}_{ijk} x_{ij} x_{kl} = \sum_{ijk\in I^3} \hat{u}^{123}_{ijk} x_{ij} \sum_{l:ijkl\in\bar{I}} x_{kl} \ . \tag{67}$$

We will analyze the last summation of (67) in two cases. If $i = k$, then compatibility of *ijkl* implies $\{l : ijil \in \bar{I}\} = \{j\}$. This gives

$$\sum_{l:ijkl\in\bar{I}} x_{kl} = \sum_{l:ijil\in\bar{I}} x_{il} = x_{ij} \tag{68}$$

in which case (67) becomes

$$\sum_{ijkl\in\bar{I}} \hat{u}^{123}_{ijk} x_{ij} x_{kl} = \sum_{ijk\in I^3} \hat{u}^{123}_{ijk} (x_{ij})^2 = \sum_{ijk\in I^3} \hat{u}^{123}_{ijk} x_{ij} \tag{69}$$

where the last equality follows from the fact that $x_{ij} \in \{0,1\}\ \forall i, j$. Similarly, if $i \neq k$, then compatibility of *ijkl* implies $\{l : ijkl \in \bar{I}\} = I - \{j\}$. This gives

$$\sum_{l:ijkl\in\bar{I}} x_{kl} = 1 - x_{kj} \tag{70}$$

and consequently (67) becomes

$$\sum_{ijkl\in\bar{I}} \hat{u}^{123}_{ijk} x_{ij} x_{kl} = \sum_{ijk\in I^3} \hat{u}^{123}_{ijk} x_{ij} (1 - x_{kj}) = \sum_{ijk\in I^3} \hat{u}^{123}_{ijk} x_{ij} \tag{71}$$

where the last equality follows from the fact that if $x_{ij} = 1$, then $x_{kj} = 0$ by the assignment constraints. Hence, we can conclude that

$$\sum_{ijkl\in\bar{I}} \hat{u}^{123}_{ijk} x_{ij} x_{kl} = \sum_{ijk\in I^3} \hat{u}^{123}_{ijk} x_{ij} \tag{72}$$

The other summations can be similarly processed (see Appendix A for details) using the assignment constraints and the fact that $x_{ij} \in \{0,1\}\ \forall i, j$ to obtain the following equality:

$$\begin{aligned}
&\sum_{ijkl\in I^4} C_{ijkl} x_{ij} x_{kl} = \\
&\sum_{ijk\in I^3} \hat{u}^{123}_{ijk} x_{ij} + \sum_{ijl\in I^3} \hat{u}^{124}_{ijl} x_{ij} + \sum_{ikl\in I^3} \hat{u}^{134}_{ikl} x_{kl} + \sum_{jkl\in I^3} \hat{u}^{234}_{jkl} x_{kl} + \\
&n \sum_{ij\in I^2} \hat{u}^{12}_{ij} x_{ij} + \sum_{ik\in I^2} \hat{u}^{13}_{ik} + \sum_{il\in I^2} \hat{u}^{14}_{il} + \sum_{jk\in I^2} \hat{u}^{23}_{jk} + \sum_{jl\in I^2} \hat{u}^{24}_{jl} + n \sum_{kl\in I^2} \hat{u}^{34}_{kl} x_{kl} + \\
&n \sum_{i\in I} \hat{u}^{1}_{i} + n \sum_{j\in I} \hat{u}^{2}_{j} + n \sum_{k\in I} \hat{u}^{3}_{k} + n \sum_{l\in I} \hat{u}^{4}_{l} + \\
&+ n^2 \hat{u}^{0}_{0}
\end{aligned} \tag{73}$$

The resulting LAP has the following cost coefficient for the variable x_{ij}:

$$\hat{c}_{ij} = \sum_{r \in I} \hat{u}_{ijr}^{123} + \sum_{r \in I} \hat{u}_{ijr}^{124} + \sum_{r \in I} \hat{u}_{rij}^{134} + \sum_{r \in I} \hat{u}_{rij}^{234} + n.\hat{u}_{ij}^{12} + n.\hat{u}_{ij}^{34} \tag{74}$$

so that the objective function $\sum_{ijkl \in I^4} C_{ijkl} x_{ij} x_{kl}$ of the QAP is equal to the objective function $\sum_{ij \in I^2} \hat{c}_{ij} x_{ij}$ of the resulting LAP plus the constant $\hat{K}$ where

$$\begin{aligned} \hat{K} = & \sum_{ik \in I^2} \hat{u}_{ik}^{13} + \sum_{il \in I^2} \hat{u}_{il}^{14} + \sum_{jk \in I^2} \hat{u}_{jk}^{23} + \sum_{jl \in I} \hat{u}_{jl}^{24} + \\ & n\left(\sum_{i \in I} \hat{u}_i^1 + \sum_{i \in I} \hat{u}_i^2 + \sum_{i \in I} \hat{u}_i^3 + \sum_{i \in I} \hat{u}_i^4 + n.\hat{u}_0^0 \right). \end{aligned} \tag{75}$$

Thus, the instance of the QAP defined by C is solvable as a LAP whenever the system $Au = \overline{C}$ has a solution. □

Theorem 7 has a generalization for a larger class of problems that includes the QAP that we give in Appendix B. The result also extends to the related problems of Axial 3D Assignment Problem and Planar 3D Assignment Problem that we give in Appendix C.

Define Class 1 to be the set of instances of the QAP for which (63) has a solution. The following algorithm checks whether or not an instance belongs to Class 1 and solves it whenever it does:

Step 1. Solve $Au = \overline{C}$ to obtain a solution $\hat{u}$, if it exists. If no solution exists, stop. The instance does not belong to Class 1. Else, continue.

Step 2. Define the cost coefficients $\hat{c}_{ij}$ using $\hat{u}$ in (74).

Step 3. Solve the resulting LAP to get an optimal solution $\hat{X} = (\hat{x}_{ij})$. Then $\hat{X}$ solves the QAP instance and its optimal objective value is

$$\sum_{ijkl \in I^4} C_{ijkl} \hat{x}_{ij} \hat{x}_{kl} = \sum_{ij \in I^2} \hat{c}_{ij} \hat{x}_{ij} + \hat{K} \tag{76}$$

where $\hat{K}$ is as defined in (75).

It would be unreasonable to expect arbitrary costs to be additively decomposable. For example, QAPLIB instances tabulated in Table 8 are not additively decomposable. An additively decomposable instance closest in some sense to a given instance can be found by solving a linear program that minimizes a predefined distance between the two instances. For example, an L_∞ norm between a given instance *C* and an additively decomposable instance, say, *C*' of the same dimension can be found by minimizing $L_{\inf(S)}$ subject to $AX + D = C$ where the vector *D* defines the coordinatewise deviations between the instance *C* and the approximating instance *C*' defined by *C*-*D* and objective function selects *X* and *D* in such a way that the maximum deviation is minimized. We may solve the instance *C*' as a LAP (using the accompanying decomposition *X*) and compare the optimal objective value *Z*(*C*) of the QAP instance *C* by the objective value *Z*'(*C*') where *Z*' gives the QAP objective value corresponding to the solution for *C*'. The deviations $(Z'(C') - Z(C))/Z(C)$ for the listed QAPLIB problems are given in Table 8. The tabulated results show that the deviation from optimality via an approximating additive decomposition could be quite high and it would be misleading to suggest that the decomposition proposed in this study can provide heuristically good approximations to QAP instances that do not conform to additive decomposition. Nevertheless, whenever

a large QAP instance is encountered that is not solvable using known methods, it can be solved as a LAP if an attempt to decompose it additively is successful.

Table 8: Deviations of the objective function values of the solutions obtained by solving the closest element of Class 1, from the optimal objective function values

Instance	***Z'(C')***	***Z(C)***	***(Z'(C') - Z(C))/Z(C)***
chr12a	45362.00	9552.00	374.90%
had12	1764.00	1652.00	6.78%
nug12	782.00	578.00	35.29%
rou12	295208.00	235528.00	25.34%
scr12	52784.00	31410.00	68.05%
tai12a	303494.00	224416.00	35.24%

4.2 Multiplicative Decomposition

We now propose another way of decomposing the general cost coefficients. This decomposition requires solving a nonlinear system of equations with $O(n^2)$ variables and $O(n^4)$ equations. Chen (1995) gave a similar decomposition that results in a parametric LAP whose complexity status is open whereas our decomposition implies polynomial time solvability of the QAP whenever the decomposition proposed in Theorem 8 is valid.

Define first $\underline{z}(c)$ and $\bar{z}(c)$ to be the minimum and maximum objective values of the LAP, respectively, for which the cost data is $c = (c_{ij})$.

Theorem 8: If there exists $v = (v_{ij}, ij \in I^2)$ that satisfies

$$v_{ij} v_{kl} = C_{ijkl}, \quad ijkl \in \bar{I}, \tag{77}$$

and if $0 \le \underline{z}(v)$ or $\bar{z}(v) \le 0$, then the instance of the QAP defined by costs $C_{ijkl}, \quad ijkl \in I^4$, is equivalent to the LAP with costs $v_{ij}, ij \in I^2$, for the case $0 \le \underline{z}(v)$, and to the LAP with costs $-v_{ij}, ij \in I^2$, for the case $\bar{z}(v) \le 0$.

Proof: Assume that such $v_{ij}, ij \in I^2$, exist. Then the objective function becomes:

$$\sum_{ijkl \in I^4} v_{ij} v_{kl} x_{ij} x_{kl} = \sum_{ijkl \in \bar{I}} v_{ij} v_{kl} x_{ij} x_{kl} \tag{78}$$

Reorganizing the terms, (78) can be rewritten as:

$$\sum_{ij \in I^2} v_{ij} x_{ij} \sum_{kl \in I^2} v_{kl} x_{kl} = (\sum_{ij \in I^2} v_{ij} x_{ij})^2 \tag{79}$$

If $0 \le \underline{z}(v)$, all feasible assignments induce a nonnegative objective value in the LAP with cost vector $v = (v_{ij})$ so that any feasible assignment that minimizes $\sum_{ij \in I^2} v_{ij} x_{ij}$ also minimizes $(\sum_{ij \in I^2} v_{ij} x_{ij})^2$.

If $\bar{z}(v) \le 0$, all feasible assignments yield a nonpositive objective value in the LAP so that any feasible assignment that minimizes $\sum_{ij \in I^2} - v_{ij} x_{ij}$ also minimizes $(\sum_{ij \in I^2} v_{ij} x_{ij})^2$. □

Theorem 8 also has a generalization for a larger class of problems that includes the QAP. This is given in Appendix B.

Define Class 2 to be the set of instances of the QAP that fulfills the assumptions of Theorem 2. Notice that every element of this class must satisfy $v_{ij}{}^2 = C_{ijij}$ (or equivalently $v_{ij} = \pm\sqrt{C_{ijij}}$), implying that an instance for which $C_{ijij} < 0$ for some $ij \in I^2$ is not an element of Class 2. Note that if all $C_{ijij}, ij \in I^2$, are nonnegative, two possible values can be assigned to each v_{ij} corresponding to the plus or minus roots so that there are 2^{n^2} possible choices of the multipliers $(v_{ij}, ij \in I^2)$. Despite the exponential number of possibilities, the

following procedure identifies the correct values of the multipliers in $O(n^2)$ time (followed by an $O(n^4)$ secondary check). The procedure determines whether or not a given instance belongs to Class 2.

Step 1. Pick an arbitrary facility-location pair *ij*. Set $v_{ij} = \sqrt{C_{ijij}}$. Note that whenever a multiplicative decomposition with multipliers $v_{ij}, ij \in I^2$ exists, another multiplicative decomposition with multipliers $-v_{ij}, ij \in I^2$, also exists. Hence, setting $v_{ij} = \sqrt{C_{ijij}}$ for a single pair *ij* does not result in a loss of generality.

Step 2. For every facility-location pair *ab* where $i \neq a$ and $j \neq b$, go to a) or b) depending on if $C_{ijab} < 0$ or $C_{ijab} \geq 0$, respectively.

a) Case with $C_{ijab} < 0$: Check the equality $v_{ij}.(-\sqrt{C_{abab}}) = C_{ijab}$. If the equality fails, then stop (no multiplicative decomposition exists), else set $v_{ab} = -\sqrt{C_{abab}}$.

b) Case with $C_{ijab} \geq 0$: Check the equality $v_{ij}.(\sqrt{C_{abab}}) = C_{ijab}$. If the equality fails, then stop (no multiplicative decomposition exists), else set $v_{ab} = \sqrt{C_{abab}}$.

If termination has not occurred for any of the pairs checked in Step 2, continue to Step 3.

Step 3. For the facility-location pairs $il \in I^2, l \in I - \{j\}$, pick a facility-location pair $ab \in I^2$, where $a \neq i$ and $b \notin \{j, l\}$. Check the equality $(\sqrt{C_{ilil}}).v_{ab} = C_{ilab}$. If the equality is satisfied, set $v_{il} = \sqrt{C_{ilil}}$. Else, check the equality $(-\sqrt{C_{ilil}}).v_{ab} = C_{ilab}$. If the equality is satisfied, set $v_{il} = -\sqrt{C_{ilil}}$; else, stop (no multiplicative decomposition exists).

If termination has not occurred for any of the pairs checked in Step 3, continue to Step 4.

Step 4. For the facility-location pairs $kj \in I^2, k \in I - \{i\}$, pick a facility-location pair $ab \in I^2$, where $a \notin \{i,k\}$ and $b \neq j$. Check the equality $(\sqrt{C_{kjkj}}).v_{ab} = C_{kjab}$. If the equality is satisfied, set $v_{kj} = \sqrt{C_{kjkj}}$. Else, check the equality $(-\sqrt{C_{kjkj}}).v_{ab} = C_{kjab}$. If the equality is satisfied, set $v_{kj} = -\sqrt{C_{kjkj}}$; else, stop (no multiplicative decomposition exists).
If termination has not occurred for any of the pairs checked in Step 4, continue to Step 5. All multipliers $v_{pq}, pq \in I^2$, have now been determined.

Step 5. Check the set of equalities $v_{pq}v_{st} = C_{pqst}$ for any of the quadruplets $pqst$ in $\bar{I}$ not checked yet in the previous steps. If all equations are satisfied, a multiplicative decomposition is on hand (found at the end of Step 4), else no multiplicative decomposition exists with multipliers $v_{ij}, ij \in I^2$.

The steps of the algorithm above take $O(1)$, $O(n^2)$, $O(n)$, $O(n)$, and $O(n^4)$ time, respectively. If a multiplicative decomposition has been found, the next step of the procedure is to solve the LAPs with the objective function $\min \sum_{ij \in I^2} v_{ij}x_{ij}$ and $\min \sum_{ij \in I^2} -v_{ij}x_{ij}$ to get the values $\underline{z}(v)$ and $\bar{z}(v)$, respectively. If $0 \leq \underline{z}(v)$ or $\bar{z}(v) \leq 0$, then the solution of the corresponding LAP qualifies as optimal for the QAP instance on hand. If the last condition does not hold, then the QAP on hand is equivalent to what we refer to as "the absolute Linear Assignment Problem" which we prove to be NP-Hard (see Theorem 9).

We note here that if a multiplicative decomposition exists for a QAP with *nonnegative* costs, then Theorem 2 applies without requiring $0 \leq \underline{z}(v)$ or

$\bar{z}(v) \le 0$. This follows from the fact that all $v_{ij}, ij \in I^2$, must be of the same sign, otherwise there is a negative cost $C_{ijkl} = v_{ij} v_{kl}$ with multipliers of opposite signs. Consequently, $\sum_{ij \in I^2} v_{ij} x_{ij}$ is either nonnegative or nonpositive regardless of the assignment. Hence, all assumptions of Theorem 2 are satisfied.

Note also that any QAP with arbitrary costs can be transformed into an equivalent QAP with nonnegative costs by adding a sufficiently large constant to each cost term. If a multiplicative decomposition exists for the transformed costs, the transformed as well as the original QAP on hand are polynomial time solvable. If no multiplicative decomposition exists for the transformed QAP, it is still possible that there exists a multiplicative decomposition for the original QAP with arbitrary costs. In this case, the multipliers may be of mixed signs and it is necessary to check the condition $0 \le \underline{z}(v)$ or $\bar{z}(v) \le 0$. If this condition does not hold, then we have $\underline{z}(v) < 0 < \bar{z}(v)$ and the QAP on hand is equivalent to the minimization of $(\sum_{ij \in I^2} v_{ij} x_{ij})^2$ subject to (2), (3), and (4) which is equivalent, in turn, to the minimization of $\left| \sum_{ij \in I^2} v_{ij} x_{ij} \right|$ subject to the same constraints. This last problem, which we refer to as the *absolute LAP*, seeks an assignment where the objective value is as close to 0 in absolute value as possible. We prove now the absolute LAP is NP-Hard.

Theorem 9: $\min \left| \sum_{ij \in I^2} v_{ij} x_{ij} \right|$ subject to (15), (16), and (22) is NP-Hard.

Proof: Consider the problem of finding an assignment which has a specified value of the LAP. That is, find a solution $\mathrm{X} = \left(x_{ij} \right)$, if it exists, such that X satisfies (15), (16), and (22) while also satisfying $\sum_{ij \in I^2} c_{ij} x_{ij} = r$ for a given

constant *r*. This problem is NP-Complete when c_{ij} and *r* are general integers (Chandrasekaran, Kabadi, and Murty, 1982). To reduce this problem to the absolute LAP, observe that $\sum_{ij\in I^2} c_{ij}x_{ij} = r$ is equivalent to $\sum_{ij\in I^2}(c_{ij} - \frac{r}{n})x_{ij} = 0$ since the sum of the assignment variables is *n*. The last equality is equivalent to $\left|\sum_{ij\in I^2}(c_{ij} - \frac{r}{n})x_{ij}\right| \le 0$. Thus, finding an assignment for which $\sum_{ij\in I^2} c_{ij}x_{ij} = r$ is equivalent to the recognition form of the absolute LAP with an upper bound of 0 on the objective value $\left|\sum_{ij\in I^2} v_{ij}x_{ij}\right|$ where $v_{ij} = c_{ij} - \frac{r}{n}$. Consequently, the recognition form of the absolute LAP is NP-Complete and the optimization form is NP-Hard. □

As a consequence, whenever there is a multiplicative decomposition for which $\underline{z}(v) < 0 < \bar{z}(v)$, the QAP on hand reduces to an absolute LAP which is also NP-Hard. Despite that, it may be easier, on the average, to solve the absolute LAP than the QAP.

We now focus on the Koopmans-Beckmann form and specialize Theorem 8 to this case. It suffices to decompose the flow and distance matrices separately. Çela (1998) proved that the instances with multiplicatively decomposable flow and distance matrices are NP-Hard in general. We prove that the nonnegativity of both flow and distance data guarantees reduction to the LAP whenever a multiplicative decomposition exists for the flow and distance matrices.

Corollary 1 to Theorem 8: For an instance of the Koopmans-Beckmann form of the QAP with nonnegative flow and distance data $F = (f_{ij})$ and $D = (d_{ij})$, respectively, if two *n*-vectors $u = (u_1,...,u_n)$ and $v = (v_1,...,v_n)$ exist such that

$f_{ij} = u_i u_j, ij \in I^2$, and $d_{ij} = v_i v_j, ij \in I^2$, then this instance is equivalent to an instance of the LAP with costs $u_i v_j, ij \in I^2$.

Proof: When such *u* and *v* exist, the objective function of the QAP becomes

$$\sum_{ijkl \in I^4} u_i u_k v_j v_l x_{ij} x_{kl} \tag{80}$$

Reorganizing the terms, (80) can be rewritten as

$$\sum_{ij \in I^2} u_i v_j x_{ij} \sum_{kl \in I^2} u_k v_l x_{kl} = (\sum_{ij \in I^2} u_i v_j x_{ij})^2 \tag{81}$$

Because f_{ij} and d_{ij} are nonnegative, *u* and *v* are also nonnegative (by multiplying either vector by -1 as necessary). It follows that the minimization of $(\sum_{ij \in I^2} u_i v_j x_{ij})^2$ is equivalent to the minimization of $\sum_{ij \in I^2} u_i v_j x_{ij}$ since the objective value is nonnegative for every feasible assignment. Hence, the QAP on hand is solvable as a LAP. □

We define Class 3 to be the set of instances of the Koopmans-Beckmann form of the QAP for which Corollary 1 to Theorem 8 is valid. Note that Class 3 is a subclass of Class 2. Because the Koopmans-Beckmann form of the QAP is a sufficiently important special form of the QAP, Class 3 is in similar standing relative to the Koopmans-Beckmann form as Class 2 is relative to the general form.

4.3 Instances Partially Reducible to LAPs

We now turn our attention to a class of instances of the QAP that reduce to LAPs after a set of assignments are made. If the number of required assignments is small, the instance on hand can be solved efficiently by solving a series of LAPs.

Let $\bar{c}_{ik} = 1$ if there exist locations j and l such that $C_{ijkl} \neq 0$, and $\bar{c}_{ik} = 0$, otherwise.

Theorem 10: For a size n instance of the QAP, consider a set $S \subseteq I$, if it exists, such that the following property holds: $i,k \in I$ and $\bar{c}_{ik} = 1 \Rightarrow \{i,k\} \cap S \neq \phi$. Then the instance can be solved by solving $\binom{n}{p} p!$ LAPs, each of size $n - p$, where $p = |S|$.

Proof: The objective function of the Lawler form can be rewritten as

$$\begin{aligned} &\sum_{i \in S, k \in S} \sum_{jl \in I^2} C_{ijkl} x_{ij} x_{kl} + \\ &\sum_{i \in S, k \notin S} \sum_{jl \in I^2} C_{ijkl} x_{ij} x_{kl} + \\ &\sum_{i \notin S, k \in S} \sum_{jl \in I^2} C_{ijkl} x_{ij} x_{kl} + \\ &\sum_{i \notin S, k \notin S} \sum_{jl \in I^2} C_{ijkl} x_{ij} x_{kl} \end{aligned} \tag{82}$$

The last summation is zero since $i,k \in S^c \equiv I - S$ implies $\bar{c}_{ik} = 0$. If the facilities in S are assigned to some p (=|S|) locations in I, then the values of x_{ij} are fixed $\forall\ i \in S$ and $j \in I$ so that the first summation becomes a constant while the second and the third summations become linear in the remaining variables. Hence, the resulting subproblem is a LAP of size $n - p$. Since there are $\binom{n}{p} p!$ possible assignments of the p facilities (in S) to some p of n available locations, one can find the optimal solution of the instance on hand by solving $\binom{n}{p} p!$ LAPs each of size $n - p$. □

Remark: It is desirable to have the cardinality p of S as small as possible to minimize the number of LAPs to be solved.

We now specialize the foregoing theorem to the Koopmans-Beckmann form. We may apply the theorem to either F or D. We only analyze the case with F. The remaining case is similar.

Theorem 11: For a size n instance of the Koopmans-Beckmann form of the QAP, consider a set $S \subseteq I$ for which the following property holds: $i, j \in I$ and $f_{ij} > 0 \Rightarrow \{i, j\} \cap S \neq \phi$. If $|S| = p$, then the instance can be solved by solving $\binom{n}{p} p!$ LAPs of size $n - p$.

Proof: The objective function of the Koopmans-Beckmann form can be rewritten as:

$$\begin{aligned} &\sum_{i \in S, k \in S} \sum_{jl \in I^2} f_{ik} d_{jl} x_{ij} x_{kl} + \\ &\sum_{i \in S, k \notin S} \sum_{jl \in I^2} f_{ik} d_{jl} x_{ij} x_{kl} + \\ &\sum_{i \notin S, k \in S} \sum_{jl \in I^2} f_{ik} d_{jl} x_{ij} x_{kl} + \\ &\sum_{i \notin S, k \notin S} \sum_{jl \in I^2} f_{ik} d_{jl} x_{ij} x_{kl} \end{aligned} \tag{83}$$

The last summation is zero since $f_{ij} = 0 \; \forall (i, j) \in S^c \times S^c$. If the facilities in S are assigned to some p locations in I, then the values of x_{ij} are fixed $\forall \; i \in S$ and $j \in I$ so that the first summation becomes a constant while the second and the third summations become linear. Hence, the resulting subproblem is a LAP of size $n - p$. Since there are $\binom{n}{p} p!$ possible assignments of p facilities to n locations,

one can find the optimal solution of the instance on hand by solving $\binom{n}{p} p!$ LAPs each of size $n - p$. □

A set S that fulfills the assumptions of Theorem 11 can be computed by constructing a graph as follows.

1. For each pair of facility indices $ik \in I^2$, compute $\bar{c}_{ik}$.
2. Construct the undirected graph $G = (I, E)$ with node set I and edge set E where the edge set is defined as follows: $(i,k) \in E \Leftrightarrow \bar{c}_{ik} = 1$.
3. A subset of the nodes of G defines a set S if every edge in E has at least one node in S. Note that any such set S is referred to as a covering by nodes (Wolsey, 1998).

A covering by nodes with minimum cardinality can be found by solving the following set covering problem. Let y_i be 1 if i is an element of the set S, and 0 otherwise.

$$\min \sum_{i \in I} y_i \tag{84}$$

$$y_i + y_k \geq 1, (i,k) \in E \tag{85}$$

$$y_i \in \{0,1\}, i \in I \tag{86}$$

For the Koopmans-Beckmann form, we define the edge set E of G as follows: $f_{ik} > 0 \Leftrightarrow (i,k) \in E$.

Note that although the set covering problem is NP-Hard, it is well studied and can be solved quite efficiently for instances of size up to 50. If the minimum covering set has, say, no more than 5 facilities, then the resulting series of LAPs

is not too many. Hence, the proposed method for solving the QAP is quite efficiently handled (even though non-polynomial in general). The minimum covering set of facilities (or locations) can also be used as a branching list for other exact solution methods.

4.4 Flow and Distance Matrices with Special Structure

During a recent study of ours (Erdoğan and Tansel, 2005), we have noticed that a naïve lower bound on the objective value of the QAP has been attained by one of the test problems, `chr18b` (Christofides and Benavent, 1989), available in the QAPLIB. A close examination of `chr18b` has revealed that the "flow" data can be characterized by a Hamiltonian path while the "distance" data can be characterized by that of a "grid" graph. While the structure of the flow data for `chr18b` can be extracted quite directly, it is not at all obvious that its distance data comes from a grid structure. In this section, we present results that explain why and how `chr18b` (and similar instances) can be solved in polynomial time. We note that the polynomial time solvability of `chr18b` has gone unnoticed for many years until our work in this dissertation and Erdoğan and Tansel (2005).

Let $F = [f_{ik}]$ and $D = [d_{jl}]$ be the n by n matrices specifying the problem data. Let $G_F = (I, A_F)$ be the undirected graph with node set I and edge set $A_F = \{(i,j): f_{ij} > 0 \text{ or } f_{ji} > 0\}$. We refer to G_F as the *flow graph* induced by F. We also associate a graph with the distance data D.

Theorem 12: Let d^* be the smallest positive element of D and G^* be the undirected graph with node set I and arc set A^* consisting of arcs (j,l) for which $d_{jl} = d^*$. If the flow graph G_F is isomorphic to a subgraph of G^*, then an assignment defined by this isomorphism is optimal.

Proof: Since every entry of the flow matrix f_{ik} is to be mapped to some entry d_{jl} of the distance matrix, $d^* \sum_{(i,k)\in A_F} f_{ik}$ is a valid lower bound on the objective value. The objective function value of the solution defined by the isomorphism described above attains this lower bound, and qualifies as an optimal solution. □

Theorem 12 is a general result concerning subgraph isomorphishm between G_F and G^*. In the following two subsections, we observe this result for two special cases.

4.4.1 G_F has a Path Structure and D is Induced by a Grid Graph

We say the flow graph has a *path structure* if it has no cycles and every node has a degree of 0, 1, or 2. A path structure implies each component of the graph is either a path or an isolated node. If the graph is connected, then a path structure is equivalent to a Hamiltonian path.

Given two positive integers *a* and *b*, we define an *a* by *b grid graph* G_{ab} to be an undirected graph with *ab* nodes such that the nodes are arranged in *a* rows and *b* columns and the node in row *i* and column *j* is labeled *ij* (i=1,…,a; j=1,…,b). The arc set consists of arcs that connect nodes *ij* and *kl* if and only if either $i=k$ and $l \in \{j-1, j+1\}$ or $j = l$ and $k \in \{i-1, i+1\}$. Assign the length 1 to each arc of a grid graph. We say an *n* by *n* matrix $D=[d_{jl}]$ is induced by a grid graph if there exists two positive integers *a* and *b* such that $n=ab$ and that the *n* by *n* distance matrix (defined by shortest path lengths) D_{ab} of the grid graph is identical to *D* up to a positive multiplier; that is, $D=hD_{ab}$ for some positive constant *h*.

Theorem 13: A size n instance of the QAP defined by flow and distance matrices F and D, respectively, is solvable in $O(n)$ time if the flow graph G_F has a path structure and D is induced by an a by b grid graph with $ab=n$.

Proof: Theorem 12 implies that whenever G_F has a path structure and G^* is Hamiltonian (a graph in which a Hamiltonian path can be identified in polynomial time), G_F is a subgraph of such a Hamiltonian path in G^* so that the QAP instance is solvable in polynomial time. A special case occurs when D is induced by a grid graph G_{ab} since G^* in this case is G_{ab} itself. Finding a Hamiltonian path in G_{ab} is done in constant time and the evaluation of the objective value takes $O(n)$ time that completes the proof. □

We now construct a solution that attains the lower bound $d^* \sum_{(i,k)\in A_F} f_{ik}$.

Renumber the nodes of G_{ab} so that the new node number for node ij (i=1,…,a;j=1,…,b) is $(i\text{-}1)b + j$. If G_F is a Hamiltonian path that traverses the nodes in the order, say, $j_1, j_2, \ldots, j_n$, then construct a solution to the QAP on hand by assigning facilities $j_1, j_2, \ldots, j_b$ to nodes 1, 2, …, b of G_{ab}, respectively; then assigning facilities $j_{b+1},\ldots,j_{2b}$ to nodes $2b,2b\text{-}1,\ldots,b\text{+}1$, respectively; and continuing in like manner so that for each odd integer $k \in \{1,...a\}$, the facilities $j_{(k-1)b+1}$, $j_{(k-1)b+2},\ldots,j_{(k-1)b+b}$ are assigned to the nodes $(k\text{-}1)b\text{+}1$, $(k\text{-}1)b\text{+}2,\ldots$, $(k\text{-}1)b\text{+}b$ respectively, while for each even integer $k \in \{1,...a\}$, they are assigned to the nodes $(k\text{-}1)b\text{+}b$, $(k\text{-}1)b\text{+}b\text{-}1,\ldots,(k\text{-}1)b\text{+}1$, respectively. Figure 17 illustrates the type of solution constructed in this manner. We may call this a "serpentine" solution since it is obtained by laying out the Hamiltonian path G_F on the grid graph G_{ab} in the form of a serpentine starting from node 11 and ending in node ab where all horizontal arcs of G_{ab} are covered by the Hamiltonian path while all vertical arcs are uncovered except those in the last column or the first column where the transitions are made from one row to the next one.

If G_F is not Hamiltonian, then we set the order $j_1, j_2, \ldots, j_n$ by merging the given disjoints paths, and isolated nodes (if any) in an arbitrary order (while preserving the order of the nodes in any given path). The assignment of $j_1, \ldots, j_n$ to nodes of the grid graph is done as in the previous case.

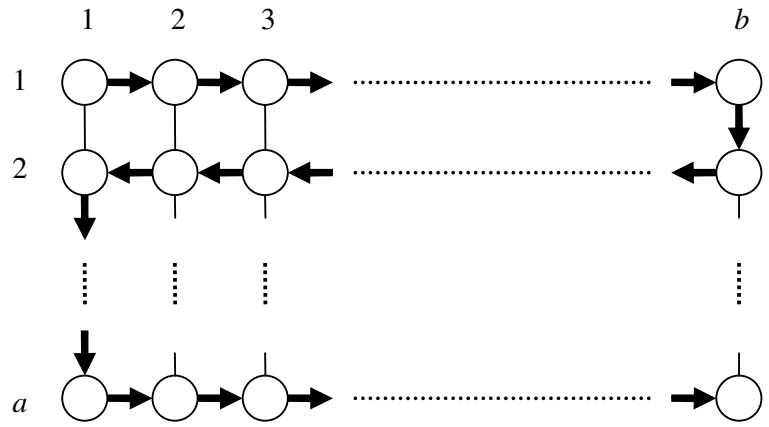

Figure 17: Hamiltonian Path on a Grid Graph (a is odd)

Assign now the length d^* to each arc of the grid graph G_{ab}. The objective function value of the constructed solution is:

$$\sum_{i,j,k,l \in I} f_{ik} d_{jl} x_{ij} x_{kl} = \sum_{i=1}^{n-1} f_{j_i j_{i+1}} d^* \qquad (87)$$

since any two consecutive nodes j_i and j_{i+1} in the Hamiltonian path occupy two adjacent locations (nodes) in the grid graph so that their separation distance is d^*. The right side of (87) equals the lower bound, proving that the constructed solution attains the lower bound. Observe that we have constructed the solution in $O(1)$ time.

We define Class 4 to be the set of instances of the Koopmans-Beckmann form of the QAP for which G_F has a path structure and D is induced by a grid graph. The test problem `chr18b` from the QAPLIB qualifies as a member of

Class 4. We have found that its flow graph is a Hamiltonian path and its distance matrix is induced by a grid graph (with $a = 6$ and $b = 3$).

We now turn our attention to the problem of identifying the members of Class 4. If G_F has more than n-1 arcs, it is not a forest and cannot have a path structure. In the remaining case, a breadth-first search (Cormen, Leiserson, and Rivest, 2000) identifies a path structure in $O(n)$ time whenever such a structure exists. Determining if G^* has a grid structure or not is relatively more complicated but is still done in $O(n)$ time by a procedure that we outline next. If G^* is a path, it has a grid structure with $a = 1$ and $b = n$. If not, there must be four nodes of degree 2 and all remaining nodes must have degrees of 3 or 4. Nodes of degree 2 and 3 make up the border while the remaining nodes make up the inner nodes. Initially, mark all nodes of degree 4 and their incident arcs as *colored*. If the uncolored subgraph is a Hamiltonian cycle, then it uniquely qualifies as the border. A one-pass traversal along this cycle beginning at a node of degree 2 determines in linear time both the labels of the nodes on the border and the dimensions a and b. Begin now uncoloring the colored subgraph by first uncoloring the colored arcs that are incident to border nodes and then uncoloring their colored end nodes. Next, uncolor the colored arcs whose both ends are incident to already uncolored nodes. This last step defines a new border that consists of the most recently uncolored arcs and nodes. Repeat this process relative to the new border until all colored arcs and nodes are uncolored. In this process, every arc is processed once. Since the number of arcs in a grid graph is bounded above by $2n$, the whole process is done in $O(n)$ time.

It follows that determining whether or not a given QAP instance qualifies as a polynomial time solvable case is done in $O(n)$ time whenever G_F and G^* (equivalently, the positions of the positive entries in F and of the minimal positive elements in D) are available as part of the input. If this is not the case, constructing G_F and G^* directly from F and D is done in $O(n^2)$ time, thereby dominating the time bound of the subsequent steps.

4.4.2 G_F is a Hamiltonian Cycle and D is Induced by an a by b Grid Graph with a > 1, b > 1, and at least one of a and b is even

Theorem 14: A size n instance of the QAP defined by flow and distance matrices F and D, respectively, is solvable in $O(n)$ time if the flow graph G_F is a Hamiltonian cycle and D is induced by an a by b grid graph with $ab=n$, $a > 1$, $b > 1$, and at least one of a and b is even.

Proof: Without loss of generality, assume that a is even. We know that G_F traverses the nodes in the order, say, $j_1, j_2, \ldots, j_n, j_1$. Construct an optimal solution by first constructing a serpentine solution for facilities j_2 to $j_{n\text{-a}+1}$ and the subgrid consisting of rows 1 to a and columns 2 to b of the grid graph (such a subgrid exists since $b > 1$, and since a is even, the serpentine path will end at the node $(a,2)$). Next, assign facilities j_{n-a+2}, $j_{n-a+3},\ldots,j_n$ to nodes $(a,1)$, $(a\text{-}1,1),\ldots,(2,1)$. Finally, assign facility j_1 to node (1,1). The last a assignments merge the path $P=\{(a,2),(a,1), (a\text{-}1, 1),\ldots,(2,1),(1,1), (1,2)\}$ with the serpentine path, yielding a Hamiltonian cycle, which satisfies the given lower bound (Figure 18). Thus, it is optimal. □

Similar to the previous case, if $|A_F|$ is not equal to n, G_F cannot be a Hamiltonian cycle. If $|A_F| = n$, a breadth-first search identifies the Hamiltonian cycle in $O(n)$ time whenever such a structure exists, or concludes that the graph is not a Hamiltonian cycle.

We define Class 5 to be the set of instances of the Koopmans-Beckmann form of the QAP for which G_F is a Hamiltonian cycle and D is induced by an a by b grid graph with $a > 1$, $b > 1$, and at least one of a and b is even.

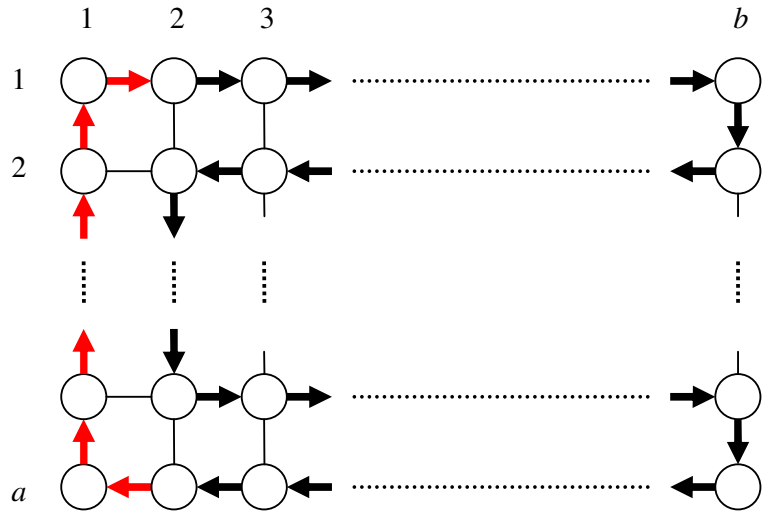

Figure 18: Hamiltonian Cycle on a Grid Graph (a is even)

4.4.3 G_F is a Star Graph

Theorem 15: A size n instance of the QAP defined by flow and distance matrices F and D, respectively, is solvable in $O(n^4)$ time if the flow graph G_F is a star graph.

Proof: If the flow graph is a star graph, then there exists a node to which every edge on the graph is incident. Denote the facility corresponding to the node as f. Using Theorem 11 with $S = \{f\}$, one can find the optimal solution by solving n LAPs of size $n - 1$, in $O(n^4)$ time. □

For a graph to qualify as a star graph, there must be exactly one node with degree greater than one, which must be connected to all other nodes. This property can be checked in $O(n)$ time using breadth-first search. We define Class 6 to be the set of instances of the Koopmans-Beckmann form of the QAP for which G_F is a star graph.

4.4.4 D is Induced by a Star Graph

Theorem 16: A size n instance of the QAP defined by flow and distance matrices F and D, respectively, is solvable in $O(n^3)$ time if D is induced by a star graph.

Proof: If D is induced by a star graph, then the equality system $d_j + d_l = d_{jl}\ \forall j,l \in I$ has a solution, where d_j denotes the distance of location j from the central node. Hence, by applying the result of Burkard et al. (1997) stated in the beginning of the section on additive decomposition, this problem can be solved as a LAP in $O(n^3)$ time. □

We define Class 7 to be the set of instances of the Koopmans-Beckmann form of the QAP for which D is induced by a star graph. Note that checking if the equality system $d_j + d_l = d_{jl}\ \forall j,l \in I$ has a solution can be done in $O(n^3)$ time using Gaussian elimination.

4.5 Flow and Distance Matrices with Ordered Entries

Theorem 17: Let $r(A,i,j)$ denote the rank of the entry (i,j) (for $i < j$) of matrix A among the entries of the upper triangular region of the matrix, and let p denote the number of entries in this region ($p = \frac{n(n-1)}{2}$). A size n instance of the QAP defined by symmetric flow and distance matrices F and D, respectively, is solvable in $O(1)$ time if the following relation holds:

$$r(F,i,j) = p - r(D,i,j) + 1 \quad \forall i,j \in I, i < j \tag{88}$$

Proof: We first give a lower bound for the QAP and then show that it is attained by a certain solution of this class. Remember that every entry of the flow matrix is to be multiplied with some entry of the distance matrix. It is known (Hardy, Littlewood, and Polya, 1952) that sorting the elements of one vector in increasing order, and the other in decreasing order minimizes the result of the dot product of the vectors. Hence, the following lower bound is valid for an instance of the Koopmans-Beckmann form of the QAP with symmetric flow and distance matrices:

$$\sum_{i,j,k,l \in I} f_{ik} d_{jl} x_{ij} x_{kl} \geq 2 \sum_{i=1}^{q} f_{[i]} d_{[q-i+1]}\,, \tag{89}$$

where $f_{[i]}$ and $d_{[i]}$ denote the i'th smallest off-diagonal entry of the flow and distance matrices, respectively.

We now show that the solution $X^* : x^*_{ii} = 1, \forall i \in I$ attains the lower bound in (89). This solution yields the following objective function value:

$$\sum_{i,j,k,l \in I} f_{ik} d_{jl} x^*_{ij} x^*_{kl} = \sum_{i,j \in I} f_{ij} d_{ij} = 2 \sum_{\substack{i,j \in I \\ i<j}} f_{ij} d_{ij} = 2 \sum_{k=1}^{p} f_{[k]} d_{[p-k+1]} \tag{90}$$

where the last equality results from the assumption of the theorem. Hence, the solution X^* attains the lower bound. □

We define Class 8 to be the set of instances of the Koopmans-Beckmann form of the QAP that fulfill the necessary conditions of Theorem 17. Members of this class can be identified by first constructing vectors containing the off-diagonal entries of flow and distance matrices, sorting one in ascending and the other in descending order, and checking if the indices of the corresponding elements of the vector match. This procedure takes $O(n^2 \log n)$ time. On the other

hand, identifying the members of this class whose flow and distance matrices are permuted appears to be a nontrivial task.

4.6 Concluding Remarks:

In this chapter, we have identified eight classes of instances that are solvable in polynomial time. We have also given an exact solution procedure that is based on identifying a subset S of facilities which, when fixed at specified locations, result in a LAP. Identifying and exploiting special structure of the input data seems to be a fertile ground for research.

The classes of problems we have presented may also be used to devise new measures of hardness for the instances of the QAP based, for example, on the deviation of the cost coefficient matrix from a "closest" easy instance. The result of the set covering problem in Section 4.3 may also be used as a measure of the complexity of an instance. The greater the size of the set S, the harder the instance.

It is unrealistic to expect randomly generated data to fall into one of these classes. However, more often than not, optimization problems encountered in the real world exhibit definite structures. The results we have presented suggest new directions to explore for discovering possibly exploitable structures.

We note here that, even if it is possible to use Theorems 7 and 8 jointly by adding the multiplicative terms $v_{ij}v_{kl}$ to the equation system (63). The resulting equation system would be nonlinear and will not be solved in polynomial time in general. Hence, there is no point in doing so.

In the next chapter, we present a lower bound generation method based on Bender's decomposition.

Chapter 5

STRONG LOWER BOUNDS BASED ON BENDER'S DECOMPOSITION

In this chapter, we present our results on a new exact solution method for the QAP. We devise a valid inequality generation method, based on decomposing a weaker form of the formulation by Adams and Johnson (1994). We prove that if all violated valid inequalities are added, the resulting formulation yields lower bounds that are at least as strong as the well known Gilmore-Lawler bound (GLB), proposed independently by Gilmore (1962) and Lawler (1963).

Recall that the MIP formulation that yields the tightest lower bounds presented by Adams and Johnson (1994) was given in Chapter 2 as IP1. We restate the formulation:

(IP1)

$$\min \sum_{i,j,k,l=1}^{n} C_{ijkl}\, y_{ijkl} \tag{14}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,...,n \tag{15}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,...,n \tag{16}$$

$$\sum_{k=1}^{n} y_{ijkl} = x_{ij} \quad \forall i, j, l = 1,...,n \tag{17}$$

$$\sum_{l=1}^{n} y_{ijkl} = x_{ij} \quad \forall i, j, k = 1,...,n \tag{18}$$

$$\sum_{i=1}^{n} y_{ijkl} = x_{kl} \quad \forall j, k, l = 1,...,n \tag{19}$$

$$\sum_{j=1}^{n} y_{ijkl} = x_{kl} \quad \forall i, k, l = 1,...,n \tag{20}$$

$$y_{ijkl} = y_{klij} \quad \forall i, j, k, l = 1,...,n : i < k, j \neq l \tag{21}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j = 1,...,n \tag{22}$$

$$y_{ijkl} \in \{0,1\} \quad \forall i, j, k, l = 1,...,n \tag{23}$$

Now consider the following constraint set:

$$y_{ijkl} \leq x_{kl} \quad \forall i, j, k, l = 1,...,n \tag{91}$$

Let us remove the constraint sets (19), (20), and (21); and add the constraint set (91) to IP1, to obtain a new formulation which we refer to as IP9. We now prove that IP9 is a valid formulation for the QAP.

Theorem 18: Let x be a feasible solution to an instance of the QAP with objective value $z_{QAP}(x)$. Then, there exists a unique vector y such that (x, y) is feasible to IP9 with objective value $z_{IP9}(x, y) = z_{QAP}(x)$.

Proof: It suffices to show that for any integral solution (x,y) of IP9 the equality $y_{ijkl} = x_{ij}x_{kl}$ holds. Let $abcd \in I^4$. There are two cases depending on if $x_{ab}x_{cd} = 0$ or 1. This results in the following subcases.

a) $x_{ab} = 0$. Constraint set (17) implies $\sum_{k=1}^{n} y_{abkd} = 0$ which in turn forces $y_{abcd} = 0$ (by nonnegativity).

b) $x_{cd} = 0$. Constraint set (91) forces $y_{abcd} \leq 0$ which in turn forces $y_{abcd} = 0$ (by nonnegativity).

c) $x_{ab} = 1$ and $x_{cd} = 1$. Constraint set (17) implies $\sum_{k=1}^{n} y_{abkd} = 1$ and constraint sets (15) and (91) imply $y_{abkd} \leq 0, \forall k : k \neq c$, which together force $y_{abcd} = 1$.

Since $y_{ijkl} = x_{ij}x_{kl}$, we can conclude that $z_{IP9}(x, y) = z_{QAP}(x)$. Note also that the equalities $y_{ijkl} = x_{ij}x_{kl}\ \forall ijkl \in I^4$ imply that y is uniquely determined by x.□

Obviously, IP9 is weaker than IP1, since (19) and (20) imply (91). Although it seems counterintuitive to replace a formulation with a weaker one, IP9 has a block angular structure that leads to decomposition. An example of the constraint structure of IP9 for $n = 3$ is depicted in Figure 19. We now propose a way of generating valid inequalities, using the idea of Bender's decomposition on IP9, for a formulation based on the variables of Kaufman and Broeckx (see Chapter 2). Before going into the details, we present a brief description of Bender's decomposition. We refer the reader to Lasdon (1970) and Bazaraa and Sherali (1990) and for a more complete exposition.

Consider an LP of the following form:

(P)

$$\min cx + fy \tag{92}$$

s.t.

$$Ax + Ey \geq b \tag{93}$$

$$x, y \geq 0 \tag{94}$$

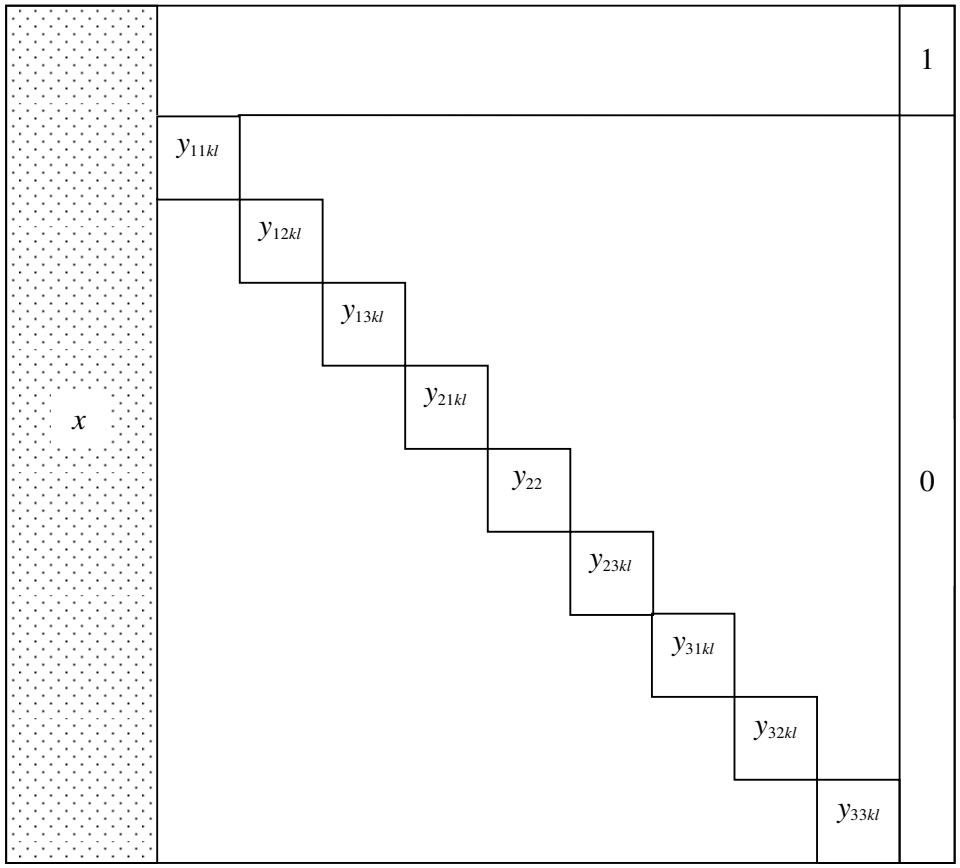

Figure 19: Block angular structure of IP9

Observe that if y is fixed at some arbitrary value, we obtain an LP in the variables x. This leads to the idea of partitioning the problem P in the following manner:

(P1)

$$\min_{x \geq 0} \{cx + \min\{fy \mid Ey \geq b - Ax, y \geq 0\}\} \tag{95}$$

Taking the dual of the inner optimization problem, we may rewrite P1 as:

(P2)

$$\min_{x\geq 0}\{cx + \max\{w(b - Ax) \mid wE \leq f, w \geq 0\}\} \quad (96)$$

Assume that the polyhedron $\{w : wE \leq f, w \geq 0\}$ is nonempty and bounded. Then P2 can be restated as:

(P3)

$$\min_{x\geq 0}\{cx + \max_{i=1,\ldots,q}\{w^i(b - Ax)\}\} \quad (97)$$

where q denotes the number of extreme points of the polyhedron $\{w : wE \leq f, w \geq 0\}$ and w^i denotes the i'th extreme point. Let z denote the objective function of P3. P3 may be rewritten as:

(P4)

$$\min z \quad (98)$$

s.t.

$$z \geq cx + w^i(b - Ax) \quad \forall i = 1,\ldots,q \quad (99)$$

$$x \geq 0 \quad (100)$$

P4 is called the *master problem*. Since the number of constraints (99) is typically too many, they are usually relaxed. The relaxed master problem is then solved to find (x^*, z^*). Then the *subproblem* $\max\{w(b - Ax^*) \mid wE \leq f, w \geq 0\}$ is solved to identify any violated inequalities of type (99). If there are no violated inequalities, then the algorithm stops. Else, the inequality is added to the relaxed master problem and the algorithm starts over. Note that more than one subproblem may exist if the original problem has a block angular structure: a variable z_i and a corresponding subproblem for the i'th block.

Following the ideas presented above, we form the following relaxed master problem:

(MP)

$$\min \sum_{i,j=1}^{n} w_{ij} \tag{101}$$

s.t.

$$w_{ij} \geq 0 \quad \forall i, j = 1,...,n \tag{102}$$

(15), (16), (22)

We refer to the variables w_{ij} as Kaufman-Broeckx variable since they have the same meaning, the cost contribution of assignment variable x_{ij}. The master problem involves the following subproblem for every *i,j* pair:

(SP_{ij})

$$\min \sum_{k,l=1}^{n} C_{ijkl}\, y_{ijkl} \tag{103}$$

s.t.

$$\sum_{k=1}^{n} y_{ijkl} = x_{ij} \quad \forall l = 1,...,n \tag{104}$$

$$\sum_{l=1}^{n} y_{ijkl} = x_{ij} \quad \forall k = 1,...,n \tag{105}$$

$$y_{ijkl} \leq x_{kl} \quad \forall k, l = 1,...,n \tag{106}$$

$$y_{ijkl} \geq 0 \quad \forall k, l = 1,...,n \tag{107}$$

where *x* is taken as a given vector of zeroes and ones. The dual of SP_{ij} is:

(DSP_{ij})

$$\max \sum_{l=1}^{n} x_{ij}\pi_l^1 + \sum_{k=1}^{n} x_{ij}\pi_k^2 + \sum_{k,l=1}^{n} x_{kl}\pi_{kl}^3 \tag{108}$$

s.t.

$$\pi_l^1 + \pi_k^2 + \pi_{kl}^3 \leq C_{ijkl} \quad \forall k, l = 1,...,n \tag{109}$$

By weak duality,

$$w_{ij} \geq \sum_{l=1}^{n} x_{ij}\pi_l^1 + \sum_{k=1}^{n} x_{ij}\pi_k^2 + \sum_{k,l=1}^{n} x_{kl}\pi_{kl}^3 \qquad (110)$$

for every π^1, π^2, π^3 feasible for DSP_{ij}.

Since IP9 is a valid formulation, SP_{ij} must have a solution for every integral solution x. The assignment constraints (15) and (16) together with nonnegativity of x defines a polytope whose extreme points are the zero/one solutions of the assignment constraints (since no zero/one solution to the assignment constraints can be expressed as a convex combination of other feasible solutions). It follows that every fractional solution x^* that satisfies the assignment constraints is a convex combination of a set of zero/one solutions. This last fact implies that SP_{ij} (and DSP_{ij}) has a solution for every fractional solution x^*. Hence, the infeasibility cuts mentioned in the generic Bender's decomposition are not required.

The valid inequality generation algorithm we propose is:

1. Solve the relaxed master problem (contrary to original Bender's decomposition that requires solving the master problem to integrality) to find x^*.
2. Using the x^*, solve the dual subproblems DSP_{ij} and add any violated inequalities of the form (110) to the master problem.
3. If at least one inequality is added, go to step 1; else, the optimal objective value of the relaxed master problem is a valid lower bound for the optimal solution of the QAP instance on hand.

Now we focus on another formulation that we form by removing the constraint set (90) from IP9. We refer to this formulation as IP10. We prove that the application of our foregoing algorithm to IP10 produces the GLB.

Theorem 19: The GLB for a given instance of QAP is equal to the lower bound generated by applying Bender's decomposition based method to IP10.

Proof: Recall from Chapter 1 that GLB is based on solving $n^2 + 1$ LAPs of size n. The first n^2 LAPs answer the following question: "What is the minimum objective function value for $x_{ij} \sum_{k,l=1}^{n} f_{ik} d_{jl} x_{kl}$ when $x_{ij} = 1$?". Each answer is recorded in the corresponding parameter l_{ij}. A final LAP is solved to obtain the bound for the QAP, with l_{ij}'s as objective function cost coefficients. Now assume that we apply Bender's decomposition to IP10. The subproblems will be

(SP'$_{ij}$)

$$\min \sum_{k,l=1}^{n} C_{ijkl} y_{ijkl} \tag{111}$$

s.t.

$$\sum_{k=1}^{n} y_{ijkl} = x_{ij} \quad \forall l = 1,...,n \tag{112}$$

$$\sum_{l=1}^{n} y_{ijkl} = x_{ij} \quad \forall k = 1,...,n \tag{113}$$

$$y_{ijkl} \geq 0 \quad \forall i, j, k, l = 1,...,n \tag{114}$$

and the master problem will be

(MP')

$$\min \sum_{i,j=1}^{n} w_{ij} \tag{115}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,...,n \tag{116}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,...,n \tag{117}$$

$$w_{ij} \geq 0 \quad \forall i, j = 1,...,n \tag{118}$$

The dual of subproblem SP'$_{ij}$ is

(DSP'$_{ij}$)

$$\max \sum_{l=1}^{n} \pi_l^1 x_{ij} + \sum_{k=1}^{n} \pi_k^2 x_{ij} \tag{119}$$

s.t.

$$\pi_l^1 + \pi_k^2 \leq C_{ijkl} \quad \forall k, l = 1,...,n \tag{120}$$

Using the dual solution π^* of SP$_{ij}$, we will be adding a valid inequality of the form:

$$w_{ij} \geq \sum_{l=1}^{n} (\pi_l^1)^* x_{ij} + \sum_{k=1}^{n} (\pi_k^2)^* x_{ij} \quad \forall i, j = 1,...,n \tag{121}$$

Notice that all the objective function coefficients of (DSP$_{ij}$) are the same. Consequently, the valid inequality returned by (DSP$_{ij}$) will be of the form

$$w_{ij} \geq x_{ij} (\sum_{l=1}^{n} (\pi_l^1)^* + \sum_{k=1}^{n} (\pi_k^2)^*) \quad \forall i, j = 1,...,n \tag{122}$$

regardless of the value of x_{ij}, where $\sum_{l=1}^{n} (\pi_l^1)^* + \sum_{k=1}^{n} (\pi_k^2)^* = l_{ij}$. In other words, every subproblem can return at most one valid inequality. After adding all such inequalities, the master problem will become:

(MP'')

$$\min \sum_{i,j=1}^{n} w_{ij} \tag{123}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1,...,n \tag{124}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1,...,n \tag{125}$$

$$w_{ij} \geq 0 \quad \forall i, j = 1,...,n \tag{126}$$

$$w_{ij} \geq l_{ij} x_{ij} \quad \forall i, j = 1,...,n \tag{127}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j = 1,...,n \tag{128}$$

Observe that taking constraint (127) as an equality gives a feasible solution that produces the GLB in the objective function of MP''.□

For any configuration of the decision variables in the master problem, the valid inequalities generated by the subproblems return exactly the same value as their counterparts in the original problem (by the strong duality theorem of Linear Programming). Hence, the lower bound generated by applying the algorithm above to IP9 (IP10) is equal to the lower bound generated by the LP relaxation of IP9 (IP10).

Theorem 20: Applying the valid inequality generation algorithm to IP9 generates a lower bound at each node of a branch-and-bound tree that is at least as strong as the GLB.

Proof: Since IP9 incorporates additional constraints, the LP relaxation of IP9 is stronger than that of IP10. Hence, our lower bound will be at least as strong as the GLB at the root node. Finally, since at every node of a branch-and-bound tree identical branching constraints will be added to both IP9 and IP10, the relationship between the bounds will not change. □

Using a set of problems from the QAPLIB, we have performed a brief computational experiment with a branch-and-cut algorithm using MP and the valid inequality generation algorithm we have proposed above. CPLEX 9.1 hybrid network/primal simplex solver was used for optimizing the resulting network flow subproblems. The runs were conducted on a single PC (3.0 Ghz Dell OPTIPLEX with 2GB RAM). The results are presented in Table 9. The first column lists the problems solved from the QAPLIB. The lower bound generated by an algorithm is given in the second column while the GLB is given in the third column. It can be seen that the lower bounds generated by our algorithm are much closer to the optimal objective value than GLB.

Table 9: Computational results for the branch-and-cut algorithm using Kaufman-Broeckx formulation, and the proposed valid inequalities

Data File	*Lower Bound at the Root Node*	*GLB at the Root Node*	*Optimum Solution value*	*B&C Nodes Traversed*	*CPU Time (sec)*
chr12a	9028.90	7245.00	9552.00	20	4.09
chr15a	7465.78	5625.00	9896.00	1335	536.55
had12	1559.65	1536.00	1652.00	1890	999.57
had14	2538.63	2492.00	2724.00	17181	20200.55
nug12	496.86	493.00	578.00	2040	1159.18
nug14	864.02	852.00	1014.00	27183	34850.07
rou12	209397.70	202272.00	235528.00	1500	2642.61
scr12	28337.25	27858.00	31410.00	630	272.72
scr15	45558.25	44737.00	51140.00	2076	3344.30
tai12a	204868.64	195918.00	224416.00	264	752.81

Although the lower bounds returned from our algorithm are stronger than the GLB, they are still not strong enough for large instances. For example rou12, one of the smallest instances in the QAPLIB, requires about 45 minutes of computing time with our lower bound generation scheme. Although the computational results are not encouraging enough yet, this study shows that there are still structures within the Pairwise Assignment Matrix that are waiting to be exploited.

Additional Insights

- IP9 exhibits the same degenerate behavior as IP1.
- We have made two attempts to use Dantzig-Wolfe decomposition for IP1. In our first attempt, we decomposed the assignment variables x_{ij} using the assignment constraints (2) and (3). In our second attempt, we have introduced the redundant equalities $\sum_{j,l=1}^{n} y_{ijkl} = 1, \forall i,k = 1,...,n$ and $\sum_{i,k=1}^{n} y_{ijkl} = 1, \forall j,l = 1,...,n$ to be used for Dantzig-Wolfe decomposition. Both attempts were failures. The introduction of the second column introduces high levels of degeneracy which makes solving the LP relaxations for instances for size 12 extremely difficult.
- Computing the GLB for general cost coefficients takes $O(n^5)$ time which can be reduced to $O(n^3)$ for the Koopmans-Beckmann costs. Our algorithm requires the solution of n^2 minimum cost network flow problems with $2n$ nodes and n^2 arcs for every pass of our valid inequality generation procedure regardless of the form of the cost coefficients. The complexity of minimum cost network flow is known to be $O(m \times log\ n \times (m + log\ n))$ (Orlin, 1988), which translates to $O(n^4 \times log\ n)$ for our case. This brings the complexity of our algorithm to $O(k \times n^4 \times log\ n)$ at each node, where k is the number of passes. Note that there is no theoretical limit on the number of passes. The GLB should be computed independently at each node of a branch-and-bound tree, whereas the effect of a valid inequality persists once it is added to the constraint set. Hence, it is hard to compare the theoretical complexities of the methods in an objective way.

Chapter 6

CONCLUSION

In this dissertation, we have performed an analysis of existing and new exact solution methods for the QAP. We first focused on the Pairwise Assignment Matrix, and devised seven new variable definitions describing the cost contribution of different subsets of pairwise assignment variables. Based on our observations, we next focused on a flow-based variable definition and the corresponding formulation, presented sets of valid inequalities, implemented a branch-and-cut algorithm, and provided the results of the algorithm for the instances in the QAPLIB. Our results suggested that while instances that have an apparent structure could be solved relatively easily, randomly created instances were out of our computational reach. Next, we analyzed the instances with structures that allow a polynomial time solution. Finally, we gave a lower bound generation scheme based on Bender's decomposition that produces bounds that are at least as strong as the GLB at each node of a branch-and-bound tree.

Linear Integer Programming proved to be very useful for a class of hard problems including the Traveling Salesman, Uncapacitated Facility Location, and Hub Location problems. This success mainly depends on binary formulations with strong valid inequalities. The variables representing the binary structure of the QAP, namely the pairwise assignment variables of Lawler, are too many ($O(n^4)$) for an efficient implementation. In addition, a high degree of degeneracy has been observed in the corresponding formulations. This led us to concentrate on smaller formulations, the variable definitions of which represent

the cost incurred by certain subsets of the binary variables. Our track of research in turn resulted in new formulations that heavily depend on the instance data. Instead of the general paradigm that puts more emphasis on solution methods independent from the instance data, we have concentrated on finding methods that find and exploit case-specific structures. Although we were able to solve large instances with definite structures, we failed to solve randomly generated instances of size larger than $n > 15$. This result may seem discouraging, but research in this field is far from complete. We suggest the following tracks of research for developing exact algorithms that may take into account special structures in data.

- To search for different partitionings of the Pairwise Assignment Matrix that can result in better linearizations.
- To determine the dominance relationships between the linearizations presented in Chapter 2.
- To conduct a polyhedral analysis of the linearizations, determining the dimensions and if possible, facets of the corresponding polyhedra.
- To construct hybrid formulations involving variables from one or more of the linearizations.
- To identify which formulations perform better for certain classes of instances.
- To determine which linearization is the best choice for a given class of instances in QAPLIB.
- To devise identification heuristics for detecting violated valid inequalities presented in Chapter 2.
- To implement a parallel branch-and-cut algorithm that can compete in the race for solving larger instances from the QAPLIB.

Our studies on the polynomially solvable classes of the QAP revealed that this field is a fertile ground for research. The following tracks of research on polynomial solvability are suggested.

- Although it may be overly optimistic to encounter a definite structure in every instance, subproblems encountered during a branch-and-bound algorithm may exhibit certain properties that can be exploited. A library of known polynomially solvable cases may be constructed, together with exact and heuristic identification algorithms, to be applied in a branch-and-bound setting to prune subproblems that conform with one of the cases in the library.
- The construction of a metric that measures the distance between the instance at hand and the closest "easy" instance, as a measure of the complexity of a given instance.
- The construction of "branching lists" at the end of which every resulting subproblem would be polynomially solvable.
- To apply the results related to grid graphs in the parallel processing domain, where grids are common structures.

Our computational experience showed that the branching efforts for large instances are rendered useless by symmetry inherent in QAP. Theoretical studies for formally establishing the symmetry and devising a metric that can measure it would be an interesting field of research.

All in all, we tried to analyze how to exploit the cost data of a given instance to achieve a provably optimal solution in a reasonable time. We believe that this track of research requires further attention and insights gained through such studies would result in discovering new discrete optimization methods.

BIBLIOGRAPHY

[1] Adams WP, and Johnson TA. Improved Linear Programming Bounds for the Quadratic Assignment Problem. In: Pardalos PM, and Wolkowicz H. (Eds.). Quadratic Assignment and Related Problems. DIMACS Series on Discrete Mathematics and Theoretical Computer Science 1994; 16: 43-75.

[2] Applegate D, Bixby R, Chvatal V, Cook W, and Helsgaun K. Traveling Salesman Problem Homepage. url: http://www.tsp.gatech.edu/, 2004.

[3] Anstreicher KM, Personal communication, 2005.

[4] Anstreicher KM, and Brixius NW. A New Bound for the Quadratic Assignment Problem based on Convex Quadratic Programming. Technical Report, Dept. of Management Sciences, University of Iowa, 1999.

[5] Anstreicher KM, and Brixius NW. Solving Quadratic Assignment Problems Using Convex Quadratic Programming Relaxations. Optimization Methods and Software 2001; 16: 49 - 68.

[6] Anstreicher KM, Brixius NW, Goux J-P, and Linderoth J. Solving Large Quadratic Assignment Problems on Computational Grids. Mathematical Programming 2002; 91: 563-588.

[7] Balas E., and Mazzola JB. Nonlinear 0-1 programming: I. Linearization Techniques. Mathematical Programming 1980; 30: 1-20.

[8] Ball MO, Kaku BK, and Vakhutinsky A. Network-Based Formulations of the Quadratic Assignment Problem. European Journal of Operational Research 1998; 104: 241-249.

[9] Bazaraa MS, Jarvis JJ, and Sherali HD. Linear Programming and Network Flows, 2nd ed. John Wiley & Sons, New York, 1990.

[10] Bazaraa MS, and Sherali MD. Benders' Partitioning Scheme Applied to a New Formulation of the Quadratic Assignment Problem. Naval Research Logistics Quarterly 1980; 27: 29-41.

[11] Brixius NW, and Anstreicher KM. Solving Quadratic Assignment Problems Using Convex Quadratic Programming Relaxations. Optimization Methods and Software 2001; 16: 49-68.

[12] Brixius NW, and Anstreicher KM. The Steinberg Wiring Problem. Working Paper, The University of Iowa, October 2001.

[13] Bruengger A, Clausen J, Marzetta A, and Perregaard M. Joining Forces in Solving Large-Scale Quadratic Assignment Problems in Parallel. DIKU Technical Report, University of Copenhagen, 1996.

[14] Burkard RE, Çela E, Demidenko VM, Metelski NN, and Woeginger GJ. Perspectives of Easy and Hard Cases of the Quadratic Assignment Problems. SFB Report 104, Institute of Mathematics, Technical University Graz, Austria, 1997.

[15] Burkard RE, Çela E, Pardalos PM, and Pitsoulis LS. The Quadratic Assignment Problem. SFB Report 126, Institute of Mathematics, Technical University Graz, Austria, 1998.

[16] Burkard RE, Çela E, Rote G, and Woeginger GJ. The Quadratic Assignment Problem with an Anti-Monge and a Toeplitz Matrix: Easy and Hard Cases, SFB Report 34, Institute of Mathematics, Technical University Graz, Austria, 1995.

[17] Burkard RE, and Derigs U. Assignment and Matching Problems. European Journal of Operational Research 1983; 13: 374 - 386.

[18] Burkard RE, Karisch SE, and Rendl F. QAPLIB --- A Quadratic Assignment Problem Library, Journal of Global Optimization 1997; 10: 391-403.

[19] Burkard R, and Offermann J. Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. Zeitschrift für Operations Research 1977; 21: B121-B132.

[20] Burkard R, and Stratmann K. Numerical Investigations on Quadratic Assignment Problems. Naval Research Logistics Quarterly 1978; 25: 129-148.

[21] Carraresi P, and Malucelli F. A New Lower Bound for the Quadratic Assignment Problem. Operations Research 1992: 40 Supplement No. 1: 22-27.

[22] Çela E. The Quadratic Assignment Problem: Theory and Algorithms. Kluwer Academic Publishers, Dordrecht/Boston/London, 1998.

[23] Chandrasekaran R, Kabadi SN, and Murty KG. Some NP-Complete Problems in Linear Programming. Operations Research Letters 1982; 1: 101-104.

[24] Chen B. Special Cases of the Quadratic Assignment Problem. European Journal of Operational Research 1995; 81: 410-419.

[25] Christofides N, and Benavent E. An Exact Algorithm for the Quadratic Assignment Problem on a Tree. Operations Research 1989; 37: 760-767.

[26] Clausen J, Espersen T., Karisch SE, Perregaard M, Sensen N, and Tschöke S. Benchmark Testing for Quadratic Assignment Problems on a Portable Parallel Branch-and-Bound Library. Work in progress, 1996.

[27] Clausen J, and Perregaard M. Solving Large Quadratic Assignment Problems In Parallel. Computational Optimization and Applications 1997; 8: 11-127.

[28] Cormen TH, Leiserson CE, and Rivest RL. Introduction to Algorithms, MIT Press, 2000.

[29] Deineko VG, and Woeginger GJ. A Solvable Case of the Quadratic Assignment Problem. Operations Research Letters 1998; 22: 13-17.

[30] Edmonds J. Maximum Matching and a Polyhedron with 0-1 Verices. Journal of Research of the National Bureau of Standards 1965; 69B: 125-130.

[31] Erdoğan G, and Tansel B. A Branch-and-Cut Algorithm for Quadratic Assignment Problems based on Linearizations. Computers and Operations Research (in press, 2005).

[32] Erdoğan G, and Tansel B. A Note on a Polynomial Time Solvable Case of the Quadratic Assignment Problem. Discrete Optimization (in press, 2006).

[33] Erdoğan G, and Tansel B. Quadratic Assignment Problems that are Solvable as Linear Assignment Problems, working paper, Bilkent University, Department of Industrial Engineering 06800 Bilkent, Ankara, TURKEY, 2005.

[34] Eschermann B, and Wunderlich HJ. Optimized Synthesis of Self-testable Finite State Machines. In 20th International Symposium on Fault-Tolerant Computing (FFTCS 20), Newcastle upon Tyne, 26-28th June, 1990.

[35] Finke G, Burkard RE, and Rendl F. Quadratic Assignment Problems. Annals of Discrete Mathematics 1987; 31: 61-82.

[36] Frieze AM, and Yadegar J. On the Quadratic Assignment Problem. Discrete Applied Mathematics 1983; 5: 89-90.

[37] Gilmore PC. Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem. SIAM Journal on Applied Mathematics 1962; 10: 305-313.

[38] Guest, MF. GAMESS-UK Benchmarks. url: http://www.cfs.dl.ac.uk/benchmarks/gamess_uk.html , 2005.

[39] Hadley SW, Rendl F, and Wolkowicz H. A New Lower Bound via Projection for the Quadratic Assignment Problem. Mathematics of Operations Research 1992; 17: 727-739.

[40] Hahn P, Grant T, and Hall N. A Branch-and-Bound Algorithm for the Quadratic Assignment Problem Based on the Hungarian Method. European Journal of Operational Research 1998; 108: 629-640.

[41] Hahn P, Hightower WL, Johnson TA, Guignard-Spielberg M, and Roucairol C. Tree Elaboration Strategies in Branch and Bound Algorithms for Solving the Quadratic Assignment Problem. Yugoslavian Journal of Operational Research 2001; 11: 41-60.

[42] Hardy GH, Littlewood JE, and Polya G. Inequalities. Cambridge University Press, London and New York, 1952.

[43] Jünger M, and Kaibel V. The QAP Polytope and the Star Transformation. Discrete Applied Mathematics 2001; 111: 283-306.

[44] Jünger M, and Kaibel V. Box-inequalities for Quadratic Assignment Polytopes. Mathematical Programming Sermon A 2001; 91: 175-197.

[45] Kaku BK, and Thompson GL. An Exact Algorithm for the Quadratic Assignment Problem. European Journal of Operational Research 1986; 23: 382-390.

[46] Kaufman L, and Broeckx F. An Algorithm for the Quadratic Assignment Problem. European Journal of Operational Research 1978; 2: 204-211.

[47] Kettani O, and Oral M. Reformulating Quadratic Assignment Problems for Efficient Optimization. IIE Transactions 1993; 25: 97-107.

[48] Koopmans TC, and Beckmann M. Assignment Problems and the Location of Economic Activities. Econometrica 1957; 25: 53-76.

[49] Lasdon LS. Optimization Theory for Large Systems. MacMillan, 1970.

[50] Lawler E. The Quadratic Assignment Problem. Management Science 1963; 9: 586-599.

[51] Li Y, and Pardalos PM. Generating Quadratic Assignment Test Problems with Known Optimal Permutations. Computational Optimization and Applications 1992; 1: 163-184.

[52] Li Y, Pardalos PM, and Resende MGC. A Greedy Randomized Adaptive Search Procedure For The Quadratic Assignment Problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 1994; 16: 237-261.

[53] Love RF, and Wong JY. Solving Quadratic Assignment Problems with Rectangular Distances and Integer Programming. Naval Research Logistics Quarterly 1976; 23: 623-627.

[54] Marzetta A, and Brungger A. A Dynamic Programming Bound For The Quadratic Assignment Problem. In: Computing and Combinatorics: 5th Annual International Conference COCOON'99, LNCS vol.1627, Heidelberg: Springer, 1999: 339-348.

[55] Mautor T, and Roucairol C. A New Exact Algorithm for the Solution of Quadratic Assignment Problems. Discrete Applied Mathematics 1994; 55: 281-293.

[56] Nugent CE, Vollman TE, and Ruml J. An Experimental Comparison of Techniques for the Assignment of Facilities to Locations. Operations Research 1968; 16: 150-173.

[57] Nyström M. Solving Certain Large Instances of the Quadratic Assignment Problem: Steinberg's Examples. Working paper, California Institute of Technology, 1999.

[58] Orlin, JB. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. Proceedings of the 20th ACM Symposium on the Theory of Computation 1988: 377-387.

[59] Pardalos PM, Rendl F, and Wolkowicz H. The Quadratic Assignment Problem: A Survey and Recent Developments. DIMACS Series in Discrete Mathematic and Theoretical Computer Science 1994; 16: 1-42.

[60] Ramachandran B, and Pekny JF. Higher Order Lifting Techniques in the Solution of the Quadratic Assignment Problem. In State of the Art in Global Optzmzzation: Computational Methods and Applications. Kluwer Academic Publishers, 1996: 75-92.

[61] Ramakrishnan KG, Resende MGC, and Pardalos PM. A Branch-and-Bound algorithm for the Quadratic Assignment Problem Using a Lower Bound Based On Linear Programming. In: Floudas, C., and Pardalos, P.M. (Eds.). State of the Art in Global Optimization: Computational Methods and Applications. Kluwer Academic Publishers, 1995.

[62] Ramakrishnan KG, Resende MGC, Ramachandran B, and Pekny J.F. Tight QAP bounds vias linear programming. In Local to Global Optimization. Pardalos PM, Migdalas A, and Burkard RE, eds. World Scientific Publishing Co., Singapore, 2002 : 297-303.

[63] Rendl F, and Wolkowicz H. Applications of Parametric Programming and Eigenvalue Maximization to the Quadratic Assignment Problem, Mathematical Programming 1992; 53: 63-78.

[64] Resende MGC, Ramakrishnan KG, and Drezner Z. Computing Lower Bounds for the Quadratic Assignment Problem with an Interior Point Algorithm for Linear Programming. Operations Research 1995; 43: 781-791.

[65] Sahni S, and Gonzalez T. P-complete Approximation Problems. Journal of the Association of Computing Machinery 1976; 23: 555-565.

[66] Steinberg L. The Backboard Wiring Problem: A Placement Algorithm. SIAM Review 1961; 3: 37-50.

[67] Wolsey LA. Integer Programming. John Wiley and Sons, 1998.

APPENDIX

A. Derivation for Additive Decomposition

The derivations below use the following pattern: The assignment variables which include indices that are not contained in the $\hat{u}$ term are factored out by the use of assignment constraints and integrality. A detailed example is given in Chapter 4, Section 1.

$$\sum_{ijkl\in\bar{I}} \hat{u}^{124}_{ijl} x_{ij} x_{kl} = \sum_{ijl\in I^3} \hat{u}^{124}_{ijl} x_{ij} \sum_{k:ijkl\in\bar{I}} x_{kl} = \sum_{ijl\in I^3} \hat{u}^{124}_{ijl} x_{ij} \tag{A1}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{134}_{ikl} x_{ij} x_{kl} = \sum_{ikl\in I^3} \hat{u}^{134}_{ikl} x_{kl} \sum_{j:ijkl\in\bar{I}} x_{ij} = \sum_{ikl\in I^3} \hat{u}^{134}_{ikl} x_{kl} \tag{A2}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{234}_{jkl} x_{ij} x_{kl} = \sum_{jkl\in I^3} \hat{u}^{234}_{jkl} x_{kl} \sum_{i:ijkl\in\bar{I}} x_{ij} = \sum_{jkl\in I^3} \hat{u}^{234}_{jkl} x_{kl} \tag{A3}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{12}_{ij} x_{ij} x_{kl} = \sum_{ij\in I^2} \hat{u}^{12}_{ij} x_{ij} \sum_{k.l:ijkl\in\bar{I}} x_{kl} = n \sum_{ij\in I^2} \hat{u}^{12}_{ij} x_{ij} \tag{A4}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{13}_{ik} x_{ij} x_{kl} = \sum_{ik\in I^2} \hat{u}^{13}_{ik} \sum_{j\in I} x_{ij} \sum_{l:ijkl\in\bar{I}} x_{kl} = \sum_{ik\in I^3} \hat{u}^{13}_{ik} \sum_{j\in I} x_{ij} = \sum_{ik\in I^2} \hat{u}^{13}_{ik} \tag{A5}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{14}_{il} x_{ij} x_{kl} = \sum_{il\in I^2} \hat{u}^{14}_{il} \sum_{j\in I} x_{ij} \sum_{k:ijkl\in\bar{I}} x_{kl} = \sum_{il\in I^2} \hat{u}^{14}_{il} \sum_{j\in I} x_{ij} = \sum_{il\in I^2} \hat{u}^{14}_{il} \tag{A6}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{23}_{jk} x_{ij} x_{kl} = \sum_{jk\in I^2} \hat{u}^{23}_{jk} \sum_{i\in I} x_{ij} \sum_{l:ijkl\in\bar{I}} x_{kl} = \sum_{jk\in I^2} \hat{u}^{23}_{jk} \sum_{i\in I} x_{ij} = \sum_{jk\in I^2} \hat{u}^{23}_{jk} \tag{A7}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{24}_{jl} x_{ij} x_{kl} = \sum_{jl\in I^2} \hat{u}^{24}_{jl} \sum_{i\in I} x_{ij} \sum_{k:ijkl\in\bar{I}} x_{kl} = \sum_{jl\in I^2} \hat{u}^{24}_{jl} \sum_{i\in I} x_{ij} = \sum_{jk\in I^2} \hat{u}^{24}_{jl} \tag{A8}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{34}_{kl} x_{ij} x_{kl} = \sum_{kl\in I^2} \hat{u}^{34}_{kl} x_{kl} \sum_{ij:ijkl\in\bar{I}} x_{ij} = n \sum_{kl\in I^2} \hat{u}^{34}_{kl} x_{kl} \tag{A9}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{1}_{i} x_{ij} x_{kl} = \sum_{i\in I} \hat{u}^{1}_{i} \sum_{j\in I} x_{ij} \sum_{kl:ijkl\in\bar{I}} x_{kl} = \sum_{i\in I} \hat{u}^{1}_{i} \sum_{j\in I} n x_{ij} = n \sum_{i\in I} \hat{u}^{1}_{i} \tag{A10}$$

$$\sum_{ijkl\in\bar{I}} \hat{u}^{2}_{j} x_{ij} x_{kl} = \sum_{j\in I} \hat{u}^{2}_{j} \sum_{i\in I} x_{ij} \sum_{kl:ijkl\in\bar{I}} x_{kl} = \sum_{j\in I} \hat{u}^{2}_{j} \sum_{i\in I} n x_{ij} = n \sum_{j\in I} \hat{u}^{2}_{j} \tag{A11}$$

$$\sum_{ijkl\in\bar{I}}\hat{u}_k^3 x_{ij}x_{kl}=\sum_{k\in I}\hat{u}_k^3\sum_{l\in I}x_{kl}\sum_{ij:ijkl\in\bar{I}}x_{ij}=\sum_{k\in I}\hat{u}_k^3\sum_{l\in I}nx_{kl}=n\sum_{k\in I}\hat{u}_k^3 \tag{A12}$$

$$\sum_{ijkl\in\bar{I}}\hat{u}_l^4 x_{ij}x_{kl}=\sum_{l\in I}\hat{u}_l^4\sum_{k\in I}x_{kl}\sum_{ij:ijkl\in\bar{I}}x_{ij}=\sum_{l\in I}\hat{u}_l^4\sum_{k\in I}nx_{kl}=n\sum_{l\in I}\hat{u}_l^4 \tag{A13}$$

$$\sum_{ijkl\in\bar{I}}\hat{u}_0^0 x_{ij}x_{kl}=\hat{u}_0^0\sum_{ij\in I^2}x_{ij}\sum_{kl:ijkl\in\bar{I}}x_{kl}=n\hat{u}_0^0\sum_{ij\in I^2}x_{ij}=n^2\hat{u}_0^0 \tag{A14}$$

B. *p*'th power problem

Let the *p*'th power assignment problem be defined as the generic problem of minimizing the sum of costs incurred by the simultaneous effect of subsets of assignments with cardinality *p* or less i.e.

$$\min\sum_{S\in I^2\cup I^4\cup\ldots\cup I^{2p}}C_S\prod_{ij\in S}x_{ij} \tag{B1}$$

$$\sum_{j=1}^{n}x_{ij}=1,\forall j=1,\ldots,n \tag{B2}$$

$$\sum_{i=1}^{n}x_{ij}=1,\forall i=1,\ldots,n \tag{B3}$$

$$x_{ij}\in\{0,1\},\forall i,j=1,\ldots,n \tag{B4}$$

Theorem B1: An instance of the *p*'th power problem involving at least one nonzero cost coefficient C_S for which $S\in I^2\cup I^4\cup\ldots\cup I^{2p-2}$ can be recast as a *p*'th power problem where all nonzero cost coefficients are associated with elements $S\in I^{2p}$.

Proof: Let $C_S\neq 0$ for a subset *S* where $S\in I^{2q}$ for some $q:1\le q\le p-1$. Let *ij* be a pair in *S*. Define *S'* to be the 2*p*-tuple whose first 2*q* components are identical to *S* while the last 2(*p*-*q*) components are the *p*-*q* repetitions of the pair *ij*. That is, $S'=(s,ij,\ldots,ij)$ where *ij* is repeated *p*-*q* times. Put $C_{S'}=C_S$. Observe that $C_{S'}\prod_{uv\in S'}x_{uv}=C_{S'}x_{ij}^{p-q}\prod_{uv\in S}x_{uv}=C_S\prod_{uv\in S}x_{uv}$, where the last equality follows

from $x_{ij} \in \{0,1\}$. Hence, the objective term corresponding to *S* can be replaced by the term corresponding to *S'*. Doing this for each such *S* gives the desired result. □

Corollary 1 to Theorem B1: For $p \geq 3$, any instance of the *p*'th power problem can be cast as a *p*'th power assignment problem with nonzero objective function coefficients only for terms involving *p* assignment variables.

For a nonempty subset *s* of $P = \{1,2,...,2p\}$, we define $q(S)$ to be the ordered $|S|$-tuple obtained from the $2p$-tuple q by retaining the indices in q that correspond to positions in *s* while deleting all other indices. For example, if $q = k_1k_2k_3k_4$ and $S = \{1,2,4\}$, then $q(S) = k_1k_2k_4$. If $S = \{2,4\}$ then $q(S) = k_2k_4$. Define also $q(\phi) = \phi \ \forall q \in I^{2p}$. Define a subset *S* of *P* to be feasible if there is at most one odd integer $k \in s$ for which k+1 is also in *S*. Let $R = \{S : S \subset P \text{ and } S \text{ is feasible}\}$. Corresponding to each element *S* of *R* and each $t \in I^{|S|}$ (if $S = \phi$, take $t = \phi$), define a variable u_t^S. We say $q \in I^{2p}$ is compatible if $q = a_1a_2a_3a_4...a_{2p-1}a_{2p}$ and the assignments defined by the pairs $(a_1,a_2),(a_3,a_4),...,(a_{2p-1},a_{2p})$ are feasible (satisfy (B2), (B3), and (B4)). Let *A* be a matrix of zeros and ones where the element in row q (with q being a compatible $2p$-tuple) and column corresponding to the pair (S,t) (with *S* being a proper subset of *P* and $t \in I^{|S|}$) is denoted by $a_q^{S,t}$. Define $a_q^{S,t} = 1$ if $q(S) = t$ and 0 otherwise. Let $A = [a_q^{S,t}]$ and *u* be the vector of u_t^S values where the columns of *A* and the elements of *u* are assumed to be identically ordered by (S,t). Let C be the vector of costs and $\overline{C}$ be the vector obtained from C by deleting all cost components corresponding to incompatible $2p$-tuples. We assume the rows of *A* and the elements of $\overline{C}$ are identically ordered.

Theorem B2: If the linear equality system

$$Au = \overline{C} \tag{B5}$$

has a solution, then the instance of the *p*'th power problem defined by *C* can be solved as a LAP.

Proof: Assume $\hat{u} = (\hat{u}_t^S)$ solves (B5). Then $A\hat{u} = \overline{C}$ implies that

$$\sum_{(S,t):q(S)=t} \hat{u}_t^S = C_q \quad , q \in \bar{I} \subset I^{2p} \tag{B6}$$

where $\bar{I}$ is the set of 2*p*-tuples that correspond to compatible assignments. Using (B6), the objective function of the *p*'th power problem can be rewritten as a linear function of the assignment variables, as in Theorem 7 (see Chapter 4).□

As an example, for $p = 3$, R = { {1,2,3,5}, {1,2,3,6}, {1,2,4,5}, {1,2,4,6}, {1,3,4,5}, {2,3,4,5}, {1,3,4,6}, {2,3,4,6}, {1,3,5,6}, {2,3,5,6}, {1,4,5,6}, {2,4,5,6}, {1,3,5}, {1,3,6}, {1,4,5}, {1,4,6}, {2,3,5}, {2,3,6}, {2,4,5}, {2,4,6}, {1,3}, {1,4}, {2,3}, {2,4}, {1,5}, {1,6}, {2,5}, {2,6}, {3,5}, {3,6}, {4,5}, {4,6}, {1}, {2}, {3}, {4}, ∅}.

Theorem B3: If there exists $v_{ij} \in R \ , ij \in I^2$, that satisfies

$$v_{ij} v_{kl} \ldots v_{yz} = C_{ijkl\ldots yz} \quad , ijkl\ldots yz \in I^{2p} \tag{B7}$$

and the optimal solution value of the LAP with the objective function $\min \sum_{ij \in I^2} v_{ij} x_{ij}$ is nonnegative, then the instance of the *p*'th power problem defined by the cost matrix *C* can be solved as a LAP with cost coefficients $v_{ij} \in R \ , ij \in I^2$.

Proof: Assume that such a *v* exists. Then the objective function becomes:

$$\sum_{ijkl\ldots yz \in I^{2p}} v_{ij} v_{kl} \ldots v_{yz} x_{ij} x_{kl} \ldots x_{yz} \tag{B8}$$

Reorganizing the terms, (B8) can be rewritten as:

$$\sum_{ij\in I^2} v_{ij}x_{ij} \sum_{kl\in I^2} v_{kl}x_{kl} \ldots \sum_{yz\in I^2} v_{yz}x_{yz} = (\sum_{ij\in I^2} v_{ij}x_{ij})^p \qquad \text{(B9)}$$

By assumption, the optimal solution of the LAP with the objective function $\min \sum_{ij\in I^2} v_{ij}x_{ij}$ is nonnegative. Since minimizing a nonnegative function and its p'th power ($p > 1$) are equivalent, the optimal solution of the LAP with the objective function $\min \sum_{ij\in I^2} v_{ij}x_{ij}$ is also the optimal solution for the instance of the p'th power problem defined by the cost matrix C. □

Remark: We do not need nonnegativity of $\min \sum_{ij\in I^2} v_{ij}x_{ij}$ for odd values of p.

C. Polynomially solvable cases of Axial 3D Assignment Problem and Planar 3D Assignment Problem

A well-known formulation for the Axial 3D Assignment Problem is given below:

$$\min \sum_{i,j,k=1}^{n} c_{ijk}x_{ijk} \qquad \text{(C1)}$$

s.t.

$$\sum_{j,k=1}^{n} x_{ijk} = 1 \quad \forall i = 1,\ldots,n \qquad \text{(C2)}$$

$$\sum_{i,k=1}^{n} x_{ijk} = 1 \quad \forall j = 1,\ldots,n \qquad \text{(C3)}$$

$$\sum_{i,j=1}^{n} x_{ijk} = 1 \quad \forall k = 1,\ldots,n \qquad \text{(C4)}$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j, k = 1,...,n \tag{C5}$$

An equivalent combinatorial formulation is:

$$\min_{\phi,\varphi \in \Pi} c_{i\phi(i)\varphi(i)} \tag{C6}$$

where $\prod$ denotes the set of all possible permutations of the integers $\{1,\ldots,n\}$. This second formulation suggests that two sets of assignment decisions are being taken simultaneously. A nonlinear model can be constructed using (C6).

$$\min \sum_{i,j,k=1}^{n} c_{ijk} x_{ij}^1 x_{ik}^2 \tag{C7}$$

s.t.

$$\sum_{j=1}^{n} x_{ij}^1 = 1 \quad \forall i = 1,...,n \tag{C8}$$

$$\sum_{i=1}^{n} x_{ij}^1 = 1 \quad \forall j = 1,...,n \tag{C9}$$

$$\sum_{j=1}^{n} x_{ij}^2 = 1 \quad \forall i = 1,...,n \tag{C10}$$

$$\sum_{i=1}^{n} x_{ij}^2 = 1 \quad \forall j = 1,...,n \tag{C11}$$

$$x_{ij}^1, x_{ij}^2 \in \{0,1\} \quad \forall i, j = 1,...,n \tag{C12}$$

Theorem C1: If the linear equality system

$$u_{ij}^{12} + u_{ik}^{13} + u_i^1 + u_j^2 + u_k^3 + u_0^0 = c_{ijk} \ \forall i, j, k = 1,...,n \tag{C13}$$

has a solution, then the instance of the Axial 3D Assignment Problem defined by c can be solved by solving two Linear Assignment Problems.

Proof: Assume $\hat{u}$ solves (C13). Then, (C7) can be rewritten as:

$$\min \sum_{i,j,k=1}^{n} \hat{u}_{ij}^{12} x_{ij}^{1} x_{ik}^{2} + \sum_{i,j,k=1}^{n} \hat{u}_{ik}^{13} x_{ij}^{1} x_{ik}^{2} + \sum_{i,j,k=1}^{n} \hat{u}_{i}^{1} x_{ij}^{1} x_{ik}^{2} + \sum_{i,j,k=1}^{n} \hat{u}_{j}^{2} x_{ij}^{1} x_{ik}^{2} + \sum_{i,j,k=1}^{n} \hat{u}_{k}^{3} x_{ij}^{1} x_{ik}^{2} + \sum_{i,j,k=1}^{n} \hat{u}_{0}^{0} x_{ij}^{1} x_{ik}^{2} \quad \text{(C14)}$$

which becomes:

$$\sum_{i,j=1}^{n} \hat{u}_{ij}^{12} x_{ij}^{1} + \sum_{i,k=1}^{n} \hat{u}_{ik}^{13} x_{ik}^{2} + \sum_{i=1}^{n} \hat{u}_{i}^{1} + \sum_{j=1}^{n} \hat{u}_{j}^{2} + \sum_{k=1}^{n} \hat{u}_{k}^{3} + n\hat{u}_{0}^{0} \quad \text{(C15)}$$

using the assignment constraints and integrality of assignment variables. Notice that the final form of the objective function is a constant plus two objective functions for two independent assignment problems. Hence, the original problem can be solved by solving two independent LAPs. □

Let $\hat{x}^1$ and $\hat{x}^2$ denote the optimal solutions of the resulting LAPs, respectively. Then the optimal solution x^* for the original problem can be computed in $O(n^3)$ time using the following formula:

$$x_{ijk}^{*} = \hat{x}_{ij}^{1} \hat{x}_{ik}^{2} \quad \forall i, j, k = 1,...,n \quad \text{(C16)}$$

The formulation for the Planar 3D Assignment Problem is similar to the Axial 3D Assignment Problem:

$$\min \sum_{i,j,k=1}^{n} c_{ijk} x_{ijk} \quad \text{(C17)}$$

s.t.

$$\sum_{k=1}^{n} x_{ijk} = 1 \quad \forall i, j = 1,...,n \quad \text{(C18)}$$

$$\sum_{j=1}^{n} x_{ijk} = 1 \quad \forall i, k = 1,...,n \quad \text{(C19)}$$

$$\sum_{i=1}^{n} x_{ijk} = 1 \quad \forall j, k = 1,...,n \quad \text{(C20)}$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j, k = 1,...,n \quad \text{(C21)}$$

A similar linear decomposition is possible for the Planar 3D Assignment Problem.

Theorem C2: If the linear equality system

$$v_{ij}^{12} + v_{ik}^{13} + v_{jk}^{23} + v_i^1 + v_j^2 + v_k^3 + v_0^0 = c_{ijk} \quad \forall i, j, k = 1,...,n \tag{C22}$$

has a solution, then every solution is optimal for the instance of the Planar 3D Assignment Problem defined by c.

Proof: Assume $\hat{v}$ solves (C22). Then, (C17) can be rewritten as:

$$\min \quad \begin{aligned} &\sum_{i,j,k=1}^{n} \hat{v}_{ij}^{12} x_{ijk} + \sum_{i,j,k=1}^{n} \hat{v}_{ik}^{13} x_{ijk} + \sum_{i,j,k=1}^{n} \hat{v}_{jk}^{23} x_{ijk} + \\ &\sum_{i,j,k=1}^{n} \hat{v}_i^1 x_{ijk} + \sum_{i,j,k=1}^{n} \hat{v}_j^2 x_{ijk} + \sum_{i,j,k=1}^{n} \hat{v}_k^3 x_{ijk} + \\ &\sum_{i,j,k=1}^{n} \hat{v}_0^0 x_{ijk} \end{aligned} \tag{C23}$$

which becomes:

$$\sum_{i,j=1}^{n} \hat{v}_{ij}^{12} + \sum_{i,k=1}^{n} \hat{v}_{ik}^{13} + \sum_{j,k=1}^{n} \hat{v}_{jk}^{23} + n\sum_{i=1}^{n} \hat{v}_i^1 + n\sum_{j=1}^{n} \hat{v}_j^2 + n\sum_{k=1}^{n} \hat{v}_k^3 + n^2 \hat{v}_0^0 \tag{C24}$$

using the constraints (C18)-(C20). Notice that the final form of the objective function is a constant. Hence every solution is optimal for the original problem. □

Since the constraint set of the Planar 3D Assignment Problem is fundamentally different from the LAP, the transformation of this problem to a LAP does not seem likely.