AN EXACT BRANCH AND BOUND ALGORITHM FOR THE GENERAL QUADRATIC ASSIGNMENT PROBLEM

by

PEERAYUTH CHARNSETHIKUL, B.E., M.S. in I.E.

A DISSERTATION

IN

INDUSTRIAL ENGINEERING

Submitted to the Graduate Faculty of Texas Tech University in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Approved

December, 1988
ACKNOWLEDGMENTS

I am deeply indebted to Dr. Eric L. Blair for his direction and continuous support throughout this work. It has been an honor for me to have had an opportunity to work closely with Dr. Blair as his student for the last four and a half years. My sincere appreciation also goes to the other members of my committee, Dr. Milton L. Smith, Dr. Rafael G. Moras, Dr. William J. Kolarik, Dr. Ronald M. Anderson and Dr. Lee Allen, for their constructive comments and suggestions.

I appreciate Miss Myra McNeil for her excellent typing and editing which helped me to complete the final version of this dissertation. Finally, I would like to thank my parents, Mr. Yos and Mrs. Ampha Charnsethikul, my brother, Mr. Ayuth Charnsethikul, and my sister, Miss Yossawadee Charnsethikul, for their support and encouragement.
TABLE OF CONTENTS

ACKNOWLEDGMENTS ............................................................. ii

ABSTRACT ................................................................. iii

LIST OF TABLES ............................................................. vi

LIST OF FIGURES ............................................................ vii

CHAPTER

I. INTRODUCTION ............................................................. 1

1.1. The Koopmans-Beckmann Formulation .................. 2

1.2. Difficulty in Solving the QAP Optimally .......... 3

1.3. Branch and Bound Methods: An Introduction .................. 4

1.4. Motivation for the Proposed Exact Algorithm ................. 5

II. LITERATURE REVIEW .................................................. 8

2.1. Exact Procedures .................................................. 8

2.2. Linear Integer Programming Formulations ................. 9

2.2.1. Lawler's Integer Programming Formulation ................. 9

2.2.2. The Mixed Integer Program of Bazaraa and Sherali ............ 10

2.2.3. The Mixed Integer Program of Kaufman and Broeckx ............ 12
LIST OF TABLES

1. Statistical Results of $T_1$ (sec.) ....................... 74
2. Statistical Results of $T_2$ (sec.) ....................... 75
3. Statistical Results of N ................................. 76
4. Estimated Exponential Function for Each Performance Measurement ........................................... 86
5. Comparisons Replication by Replication Between PM and KT .................................................. 87
6. Results of GQAP from the Sign Test ..................... 89
7. Results of KBP from the Sign Test ....................... 90
LIST OF FIGURES

1. Tree structure with single assignment, with level corresponding to locations [1] .......................... 17
2. Tree structure with single assignment, with level corresponding to objects [2] ............................ 18
3. Flow and distance for illustrative problem ........ 33
4. Tree completed for the problem of Figure 3 using the Gilmore-Lawler procedure .......................... 38
5. Tree elaborated for the problem of Figure 3 (by the Kaku-Thompson method) ............................ 46
6. Data for the problem of Figure 3 represented in terms of pairs of assignments ......................... 48
7. Tree elaborated for the problem of Figure 3 by the Gavett-Plyter algorithm .............................. 52
8. Partial tree elaborated by the pair exclusion algorithm ............................................................. 56
9. The complete search tree determined by the proposed algorithm ................................................. 70
10. $T_1$ versus n for GQAP ........................................... 79
11. $T_1$ versus n for KBP ........................................... 80
12. $T_2$ versus n for GQAP ........................................... 81
13. $T_2$ versus n for KBP ........................................... 82
14. N versus n for GQAP ........................................... 83
15. N versus n for KBP ........................................... 84
16. The illustrated tree ............................................ 104
ABSTRACT

This research is concerned with the development of an exact algorithm for a general quadratic assignment problem (QAP), of which the Koopmans-Beckmann formulation, in the context of an analysis of the location of economic activities or facilities, is a special case. The algorithm is based on the linearization of a general QAP of size n into a linear assignment problem of size n(n-1)/2. The objective value and the dual solution of this subproblem are used to compute the lower bound used in an exact branch and bound procedure. Computational experience and comparisons to other well known methods are discussed.
CHAPTER I
INTRODUCTION

The aim of the classical quadratic assignment problem (QAP) is to seek the assignment of n facilities to n locations in order to minimize the total quadratic interactive cost. This problem has a wide variety of applications such as the location of interdependent plants or facilities, the layout of interacting departments in an office building, the location of medical facilities in a hospital, the backboard wiring problem in the design of computer and other electronic equipment and some production scheduling problems with interactive cost and so on. Formally, its mathematical model may be stated as follows: given $n^4$ cost coefficients $C_{ijkl}$ $(i,j,k,L = 1,2,\ldots,n)$, determine values of the $n^2$ variables $x_{ij}$ $(i,j = 1,2,\ldots,n)$ so as to minimize

$$
\sum_{i} \sum_{j} \sum_{k} \sum_{L} C_{ijkl}x_{ij}x_{kL}
$$

(P1)

subject to
\[ \sum_{i} X_{ij} = 1 \text{ for all } j \]
\[ \sum_{j} X_{ij} = 1 \text{ for all } i \]

\[ X_{ij} = \{0,1\} \text{ for all } i \text{ and } j. \]

1.1. The Koopmans-Beckmann Formulation

Koopmans and Beckmann (1957) were the first to formulate the QAP in assigning a number of plants or indivisible operations to a number of different geographical locations. Their formulation of \( C_{ijkl} \) in problem P1, which is somewhat less general than that given above, is as follows.

\[ C_{ijkl} = f_{ik} \cdot d_{jL} + b_{ij}, \text{ if } i = k \text{ and } j = L \]
\[ = f_{ik} \cdot d_{jL}, \text{ otherwise,} \]

where

\( f_{ik} = \) the number of units of commodity to be transported from facility \( i \) to facility \( k \),

\( d_{jL} = \) the cost of transporting one unit of commodity from location \( j \) to location \( L \),

\( b_{ij} = \) the cost expected from the operation of facility \( i \) at location \( j \) and is independent of other facility locations.
Each assignment of plants to locations is represented by $X_{ij}$ where

$X_{ij} = 1$, if facility $i$ is assigned to location $j$,

$= 0$, otherwise.

Lawler (1963) illustrated that the traveling salesman problem is a special case of the above formulation, in which $b_{ij} = 0$ for all $i, j$, $d_{jL}$ is the distance traveled from city $j$ to city $L$ and

$f_{ik} = 1$, if $k = i + 1$ and $i < n$

$= 1$, if $i = n$ and $k = 1$

$= 0$, otherwise,

where

$X_{ij} = 1$, if the salesman visits city $j$ on the $i$-th position of the tour sequence,

$= 0$, otherwise.

1.2. Difficulty in Solving the QAP Optimally

The discrete nature and the complexity of the quadratic objective function do cause difficulty in verifying the optimal solution of the QAP. Horowitz and Sahni (1984)
proved that the QAP is NP-hard, which requires an exponential time algorithm to solve the problem optimally. Evidently, the implicit enumeration method which is usually referred to as "branch and bound" is the only appropriate strategy in this case. From a computational point of view, all branch and bound methods previously developed are probably not computationally feasible for \( n > 15 \) in the case of the Koopmans-Beckmann formulation and for \( n > 10 \) in the general case. Nevertheless, some strategies such as premature termination, upper bounding and stepped fathoming, which are investigated by Bazaraa and Elshafei (1979) and Bazaraa and Kirca (1983), can be used in order to obtain suboptimal solutions within a reasonable time (i.e., one hour by a mainframe computer) for larger problems \( (n > 15) \).

1.3. Branch and Bound Methods: An Introduction

Among the most general approaches to the solution of integer constrained optimization problems is that of "branch and bound." Like dynamic programming, branch and bound is an intelligently structured tree search of the space of all feasible solutions. Most commonly, this space is repeatedly decomposed into smaller subspaces, and a lower bound (in case of minimization) is computed for the cost of the solutions within each subspace. After each
partitioning, those subspaces with a bound that exceeds the cost of a known feasible solution (an upper bound) are excluded from all further partitioning. The search continues until a feasible solution is found such that its cost is no greater than the bound for any subspace. Generally, a branch and bound algorithm relies on two elements: branching and bounding. Branching is concerned with choosing a reset positioning of the solution space to elaborate and evaluate. Bounding deals with the lower bound of the objective function of each partitioning. The complete description of a branch and bound algorithm requires a specification of the rule that determines which of the currently active subspaces is to be chosen for branching, as well as the method for computing the lower bound of each active subspace (each node in the search tree). Historically, branch and bound methods have been applied to a variety of problems in integer programming, nonlinear programming, scheduling, decision processes, etc. A complete survey was reported by Lawler and Wood (1966).

1.4. Motivation for the Proposed Exact Algorithm

Although several heuristic procedures have been suggested as the only appropriate way to solve the QAP, it is still necessary to find a lower bound to evaluate the quality of the solution. Basically, the easiest way is to
find the ratio between the solution found and the lower bound obtained. A good lower bound is usually obtained from an exact procedure. With the recent development of parallel processing used in computers, it is possible to solve a QAP by the exact and inexact procedure simultaneously. Both routines can provide an updated lower bound and upper bound to each other. According to this scheme, a good solution may be detected more quickly and the optimal solution may be found in a reasonable time for some large problems.

Recently, it has been suggested that one of the most successful ways to solve a combinatorial optimization is to combine both exact and inexact methods together. This is referred to as the "hybrid method" by Bazaraa and Kirca (1983). Other desirable attributes of branch and bound methods which have received attention from researchers over the past 25 years are as follows.

First, there is a possibility of obtaining usable solutions and terminating problem-solving prior to the ultimate completion of the problem-solving process (premature termination).

Second, when a feasible solution (an upper bound) is known from experience or has been derived with the aid of a heuristic procedure (branch and bound methods always subsequently search for solutions with an objective value
better than the best known so far and will terminate if a solution is discovered attaining a known lower bound), the use of a priori knowledge of upper and/or lower bounds serves to reduce the regions that need be searched. Therefore, in contexts where good heuristic procedures are available, a good upper bound may be found helpful in reducing time spent in the branch and bound process.

Third, all exact branch and bound algorithms can be modified and employed to find not an optimal solution, but all optimal solutions or a specified number of most preferred solutions, or all solutions having a value within a specified interval of the optimal objective value, and so on. Such possibilities may be of interest in contexts of multiple objective QAP.

The purpose of this research is to present a new branch and bound algorithm for the QAP. Some previous works will be reviewed in Chapter 2 and Chapter 3. Chapter 4 will present the proposed method in full detail while computational experiences will be discussed and analyzed in Chapter 5. Finally, concluding remarks and some extensions will be described in Chapter 6.
CHAPTER II
LITERATURE REVIEW

In this chapter, a brief literature review of the QAP is discussed. In solving the QAP, a number of both exact and heuristic procedures have been reported in the literature over the past twenty-five years. Recently, more complicated models incorporating more realistic characteristics such as dynamic QAP, stochastic QAP and multiple objective QAP have been investigated by many researchers. Reviews of previous research are enumerated as follows.

2.1. Exact Procedures

the exact algorithms are based on two approaches: linear integer programming formulations or branch and bound methods.

2.2. Linear Integer Programming Formulations

Due to the complexity of the quadratic objective function in problem P1, several researchers have suggested linearizations, resulting in formulations of linear (mixed) integer programs, in order to develop some appropriate solution techniques. Unfortunately, these approaches are usually impractical for even small n. Nevertheless, these formulations are important contributions from the modeling viewpoint and can be illustrated as follows.

2.2.1. Lawler's Integer Programming Formulation

Lawler (1963) showed that a quadratic assignment problem with \( n^2 \) variables \( X_{ij} \) can be linearized by defining \( n^4 \) variables \( Y_{ijkl} \) where, effectively, \( Y_{ijkl} = X_{ij}X_{kl} \). Consider the following integer linear program in the \( n^4 + n^2 \) variables of \( Y_{ijkl} \) and \( X_{ij} \).

Minimize

\[
\sum_{ijkl} C_{ijkl} Y_{ijkl} \tag{P2}
\]
subject to

\[ \sum_{i} X_{ij} = 1 \text{ for all } j \]

\[ \sum_{j} X_{ij} = 1 \text{ for all } i \]

\[ X_{ij} + X_{kL} - 2Y_{ijkL} \geq 0 \text{ for all } i,j,k,L \]

\[ \sum_{ijkL} Y_{ijkL} = n^2 \]

\[ X_{ij} = \{0,1\} \text{ for all } i,j \]

\[ Y_{ijkL} = \{0,1\} \text{ for all } i,j,k,L. \]

From the above model, the number of constraints and variables increases from \(2n\) to \(n^4 + 2n + 1\) and from \(n^2\) to \(n^4 + n^2\), respectively. Evidently, the approach is impractical for all but the smallest problem.

2.2.2. The Mixed Integer Program of Bazaraa and Sherali

Bazaraa and Sherali (B&S) (1980) reformulated the QAP as a 0-1 mixed linear integer program. The QAP model used in their work was introduced by Graves and Whinston (1970) and is slightly different from problem P1. In that model, it is assumed that there are \(p\) commodities that are transported among \(n\) facilities. The first transformation is to transform the model to problem P1 by comparing
coefficients between two models. The second transformation uses the following formula:

\[ S_{ijkl} = \frac{b_{ij} + b_{kL}}{n - 1} + C_{ijkl} + C_{klij} \]

\( i = 1, \ldots, n-1, \ K = i+1, \ldots, n, \ L, j = 1, \ldots, n, \ L \neq j. \)

This transformation leads to the following equivalent problem.

Minimize

\[ \sum_{i=1}^{n-1} \sum_{j=1}^{n} \sum_{K=i+1}^{n} \sum_{i'=1}^{n} S_{ijkl} X_{ij} X_{kL} \]

subject to all constraints in problem P1.

To solve the above model, B&S transform P2 into the following 0-1 mixed linear integer program.

Minimize

\[ \sum_{i=1}^{n-1} \sum_{j=1}^{n} \sum_{K=i+1}^{n} \sum_{i'=1}^{n} S_{ijkl} Y_{ijkl} \]  \hspace{1cm} (P3)

subject to

\[ \sum_{j} X_{ij} = 1 \quad \text{for all } i \]
\[ \sum_{i} X_{ij} = 1 \quad \text{for all } j \]

\[ \sum_{i=1}^{n} \sum_{L=1}^{n} \sum_{L \neq j}^{n} Y_{ijkL} - (n-k)X_{ij} = 0 \quad \text{for } i = 1, \ldots, n-1 \]

\[ j = 1, \ldots, n \]

\[ \sum_{i=1}^{k-1} \sum_{j=1}^{n} \sum_{j \neq L}^{n} Y_{ijkL} - (n-k)X_{kL} = 0 \quad \text{for } k = 1, \ldots, n \]

\[ L = 1, \ldots, n \]

\[ X_{ij} = \{0,1\} \]

\[ 0 \leq Y_{ijkL} \leq 1 \quad \text{for } i = 1, \ldots, n-1 \]

\[ k = i+1, \ldots, n \]

\[ j, L = 1, \ldots, n, j \neq L. \]

Note that problem P3 has \( n^2 \) integer and \( n^2(n-1)^2/2 \) continuous variables and \( 2n^2 \) linear constraints. B&S applied Bender's decomposition to this formulation but were unable to solve problems larger than \( n = 6 \) optimally due to storage limitations.

2.2.3. The Mixed Integer Program of Kaufman and Broechx

Kaufman and Broechx (1978) have proposed a linearization method suggested by Glover (private communication). To define this method, the following two types of variables are introduced.
\[ W_{ij} = X_{ij} \sum_{k} \sum_{L} C_{ijkL} X_{kL} \]

\[ U_{ij} = \max \left( \sum_{k} \sum_{L} C_{ijkL}, 0 \right) \]

Then, problem P1 can be reduced to the following mixed integer program.

Minimize

\[ \sum_{i} \sum_{j} W_{ij} \]  \hspace{1cm} (P4)

subject to

\[ \sum_{i} X_{ij} = 1 \quad \text{for all } j \]

\[ \sum_{j} X_{ij} = 1 \quad \text{for all } i \]

\[ U_{ij} X_{ij} + \sum_{k} \sum_{L} C_{ijkL} X_{kL} - W_{ij} \leq U_{ij} \quad \text{for all } i,j \]

\[ X_{ij} \geq \{0, 1\} \quad \text{for all } i,j \]

\[ W_{ij} \geq 0 \quad \text{for all } i,j. \]

This formulation adds \( n^2 \) new continuous variables and \( n^2 \) new constraints. This gives a total of \( n^2 \) binary variables plus \( n^2 \) continuous variables, and \( n^2 + 2n \) constraints. Although this is an improvement over problem P3, it was reported by Kaku and Thompson (1986) that for a problem of
size $n = 8$, the core requirements become too large for the mixed integer code they used and optimality could not be proved.

2.2.4. Integer Program for the QAP with Rectangular Distances

Love and Wong (1976) proposed the following problem. Suppose $n$ locations are given as points in the first quadrant of a two-dimensional plane (given points can always be represented on the first quadrant by translation of axes), and suppose the flow between each pair of facilities is given. If one uses rectangular distances, which sometimes are more appropriate in industrial settings than straight-line distances, then a binary mixed integer program to solve the QAP is as follows.

Minimize

$$\sum_{i=j}^{n-1} \sum_{j=i+1}^{n} W_{ij}(R_{ij} + L_{ij} + A_{ij} + B_{ij})$$

subject to

$$R_{ij} - L_{ij} = X_i - X_j \quad i = 1, \ldots, n-1$$

$$A_{ij} - B_{ij} = Y_i - Y_j \quad j = i+1, \ldots, n$$

$$X_i + Y_i = \sum_k S_k Z_{ik} \quad \text{for all } i$$
\[ X_i - Y_i = \sum_k d_k Z_{ik} \quad \text{for all } i \]

\[ \sum_i Z_{ik} = 1 \quad \text{for all } k \]

\[ \sum_k Z_{ik} = 1 \quad \text{for all } i \]

\[ Z_{ik} = \{0,1\} \quad \text{for all } i,k \]

\( R_{ij}, L_{ij}, A_{ij}, B_{ij}, X_1, \ldots, X_n, Y_1, \ldots, Y_n \geq 0, \)

where

\( n = \) the number of facilities and the number of locations,

\( W_{ij} = \) the nonnegative weight or flow between facility \( i \) and facility \( j, \)

\( R_{ij} = \) the horizontal distance between facility \( i \) and facility \( j \) if facility \( i \) is to the right of facility \( j; \) otherwise, \( R_{ij} \) is zero,

\( L_{ij} = \) the horizontal distance between facility \( i \) and facility \( j \) if facility \( i \) is to the left of facility \( j; \) otherwise, \( L_{ij} \) is zero,

\( A_{ij} = \) the vertical distance between facility \( i \) and facility \( j \) if facility \( i \) is above facility \( j; \) otherwise, \( A_{ij} \) is zero,
B_{ij} = the vertical distance between facility i and facility j if facility i is below facility j; otherwise, B_{ij} is zero,

(X_i, Y_i) = the location of facility i, i=1,...,n, S_k is the sum of coordinates of location k,

d_k = the difference of coordinates of location k, values of the first coordinate minus value of the second coordinate, and

Z_{ik} = 1 if facility i is assigned to location k
   = 0 otherwise.

The above formulation has n^2 binary variables and n^2 + 3n constraints. It involves ordered sets which can be processed by some integer programming codes. According to the results presented in Love and Wong (1976), the approach cannot be applied successfully to QAPs with size n > 8.

2.2.5. Branch and Bound Methods

Gilmore (1962) and Lawler (1963) developed a branch and bound procedure based on the systematic consideration of single assignments X_{ij}, as shown in Figure 1 and Figure 2.

Figure 1 illustrates a tree structure with the i-th level of nodes representing the permissible assignment for
Level 1: OBJECT 1

Level 2: OBJECT 2

Level n: OBJECT n

Figure 1. Tree structure with single assignment, with level corresponding to locations [1].
Figure 2. Tree structure with single assignment, with level corresponding to objects [2].
object \(i\), while if we interchange objects and locations in the cited procedure, we have a tree with level corresponding to locations rather than objects, as illustrated in Figure 2. The lower bound in each node can be computed by solving \(n^2 + 1\) linear assignment problems (LAPs) of size \(n - 1\). These LAPs are the result of positioning \(C_{ijkl}\) corresponding with the restriction of \(X_{ij}\) in each level into \(n^2\) minors of \(n^2\) elements each. Gilmore stated in his paper that this algorithm is probably not computationally feasible for \(n\) much larger than 15.

Land (1963) and Gavett and Plyter (1966) have developed algorithms in which search proceeds on the basis of a controlled enumeration of the variables \(Y_{ijkl} = X_{ij} \cdot X_{kL}\), where as before each \(X_{ij}\) denotes the locating of object \(i\) at location \(j\). These authors also view the underlying problem as a linear assignment problem, but one of assigning a pair of object \(i\) and \(k\) to location \(j\) and \(L\). In both papers, the algorithm developed applied to the symmetric QAP with \(C_{ijkl} = C_{ILkj}\). In general, there are \(n(n-1)/2\) pairs of objects and pairs of locations in the problem which represent the dimension of the associated linear assignment. However, there are many feasible solutions to this assignment problem which are not feasible to the original QAP. For example, it is entirely acceptable in the linear assignment problem for objects A and B to be
assigned to locations 1 and 2, and objects A and C to locations 3 and 4. This solution is clearly not feasible for the original QAP. Gavett and Plyter (1966) developed a binary search tree which uses the alternate cost method of Little et al. (1963), while Land (1963) always selects from the column having the least number of feasible elements in the column reduced cost matrix.

Pierce and Crowston (1971) modified both the Gilmore-Lawler procedure and the Land-Gavette-Plyter procedure to solve the QAP. In the modified Gilmore-Lawler procedure, instead of considering all possible assignments in each level, they use the binary search tree with the alternate cost method of Little (1963) to search for the solution. They presented several pairs of exclusion algorithms which are modifications of the Land-Gavett-Plyter procedure: the basic branching rule is to exclude all pair assignments determined in each node, while a node is fathomed when a feasible assignment for the QAP is found.

Cabot and Francis (1969) developed a procedure for solving certain nonconvex quadratic minimization problems by ranking the extreme points. The procedure involves determining a related linear program having the same constraints. The extreme point ranking approach of Murty (1968) is then applied to this linear program to obtain an optimum solution to the quadratic program. They suggested
that solving the QAP is one of the applications of their
method but no further investigation has been made.

Bazaraa and Elshafei (1979) developed an exact branch
and bound procedure which is able to produce optimal solu-
tions for problems with twelve facilities or less. The
method incorporates the concept of stopped fathoming to
reduce the effort expended in searching the decision trees.
They use the binary search tree with the lower bound com-
ted by the summation of cost of the partial assignment, the
lower bound in the cost of interaction between assigned
objects and unassigned objects plus the fixed cost of
locating the unassigned objects, and the lower bound on the
cost of interaction among the unassigned objects them-
selves.

Burkard and Derigs (1980) developed a computer program
for the binary search tree with the consideration of single
assignments in each node as described in Pierce and
Crowston (1971). The lower bound of each node is computed
by a row-column reduction (referred to as "the Perturbation
method") developed by Burkard and Stratmann (1978). They
reported some computational tests for the test examples
presented in Nugent et al. (1968). For the largest test
problem of 15 facilities, the CPU time for the CDC CYBER 76
is 2947.32 seconds.
Edwards (1980) developed a binary branch and bound algorithm for the QAP which has the property that at each node an associated solution can be obtained simultaneously, thereby rendering any premature termination of the algorithm less wasteful. They also developed another method for computing the lower bound by considering the minimal quadratic residual terms in the canonical form of the QAP. No computational test was reported.

Bazaraa and Kirca (1983) uses the Gilmore-Lawler method incorporated with the property referred to as "Mirror Images" to solve the QAP with Koopmann and Beckmann Formulation. They were able to solve a hospital layout planning of size $n = 19$ optimally within 110 seconds on the CDC-70.

Kaku and Thompson (1986) developed an exact algorithm that is based on the linearization and decomposition of a general QAP of size $n$ into $n^2$ linear assignment problems of size $n - 1$. The solutions of these subproblems are used to calculate a lower bound for the original problem and this bound is then used in an exact branch and bound procedure. These subproblems are similar to the minors defined by Lawler (1963) but provide tighter bounds when the branching rule uses the concept of alternate cost as suggested by Pierce and Crowston (1971). Computational experience is given for solution to optimality of general QAPs of sizes
up to \( n = 10 \). A comparison with the Gilmore-Lawler method and some integer programming approaches as described in this section shows that this method performs better, especially as the size of the problem grows. The average time for size \( n = 10 \) by a DEC-20 was reported as 1305.65 seconds. Nevertheless, this size of problem is still relatively small and further work is suggested.

It should be pointed out that some of the exact procedures previously described are designed to solve only the QAP with the Koopmann-Beclcmann formulation. In fact, only the Gilmore-Lawler procedure, the Land-Gavett-Plyter procedure, the pair exclusion method of Pierce and Crowston and the method of Kaku and Thompson are designed and developed for the general case of QAP.

2.3. Heuristic Procedures

Due to the complexity of the QAP, in general, as shown in the previous discussion, none of the exact methods can solve a problem with high dimension (\( n > 15 \)) effectively. It is even worse in the general case of QAP for which Kaku and Thompson can solve the problems of size up to only \( n = 10 \). Thus, for larger problems, a considerable amount of effort has been given to the development of heuristic methods that obtain a good quality solution with reasonable computational efforts. A comprehensive survey of these
heuristic procedures can be found in the works of Sherali (1979), Obata (1979), and Burkard and Stratmann (1978). A brief summary of these methods is given as follows.

2.3.1. Construction Methods

Starting with a partial solution of the null assignment, a complete assignment is reached iteratively by locating one or more objects at each iteration. Basically, these approaches have been based on biased random sampling, clustering methods (e.g., Hanan and Kurtzberg, 1972), rules such as nearest neighbor and pairwise assignment, $A_i - B_j$ rule, and methods with increasing degree of freedom which were described and tested in Burkard and Stratmann (1978). Liggett (1981) developed a constructive procedure based on the enumeration strategy presented by Graves and Whinston (1970). Liggett's results are compared favorably with other constructive procedures. However, it has been found by Burkard and Stratmann (1978) that constructive approaches, in general, do not produce solutions that are sufficiently near optimal.

2.3.2. Improvement Procedures

Starting with a complete assignment of objects, an improvement over the incumbent objective function value is
sought by exchanging assignments. The procedure is terminated when no improvements are possible. Generally, the initial solution (a complete assignment) can be constructed by a constructive procedure. CRAFT (see Armour and Buffa, 1963; and Armour, 1961), or the steepest-descent pairwise-interchange procedure (according to Francis and White, 1974), is considered to be one of the best improvement procedures. The procedure examines all possible pair assignment exchanges and carries out the one which yields the largest improvement, then starts the whole procedure again until no improvement is found. Parker (1976) examined the pairwise exchanges until an improvement of the objective was found, carried out this exchange, and started a new iteration. Three-, four- and five-way exchanges have been investigated by Liggett (1981). Generally, λ-wise exchanges have produced slight improvements in solution quality, but computation times have been found to increase at prohibitive rates. Heider (1972) examined the pairwise exchanges in fixed order and carried out the exchange as soon as an improvement was found. Therefore, it is possible that there were several improvements in one iteration. Burkard and Strattmann (1978) tested the three alternatives of CRAFT and Parker and Heider, and found a variation of the last to give a superior solution. Vollmann, Nugent and Zartler (1968) presented an alternative procedure which is
a modification of CRAFT, which requires less storage and has a faster computation time. They also stated that the procedure produces results that are not different with statistical significance from CRAFT. The Hillier procedure (Hillier, 1963; Hillier and Connors, 1966) defines a "p-steps" neighborhood to search for improvement. A facility can be moved p-steps (grid-squares) horizontally, vertically, or diagonally. A move number is calculated to estimate the cost reduction that would result from moving facility i from location j to location k. The largest move number is used to identify an exchange pair and the exchange is implemented, providing a reduced cost. Move numbers are updated as necessary and p-steps changes are made until no favorable exchanges can be identified. The parameter p is then altered so that p is allowed to range all feasible solutions. Generally, p can be determined by considering the location layout or the site arrangement. Nugent et al. (1968) compared the Hillier procedure with CRAFT and biased random sampling. They concluded that the biased random sampling provided the best solution on average, but the Hillier procedure was the "best buy" considering run time as well as the quality of solution produced. This result motivated Picone and Wilhelm (1984) to develop a perturbation scheme called "the revised Hillier" to improve Hillier's solution. This new heuristic combines
the Hillier procedure with 3-way and 4-way exchanges which expand the neighborhood that is searched by the Hillier procedure. The revised Hillier procedure was compared with the Hillier procedure and CRAFT using the standard test problem shown in Nugent et al. (1968). In these tests, they found that the revised Hillier procedure consistently gave the best solution ever published. Other improvement schemes can be found in Hitching and Cottam (1976), Sherali and Rajgopal (1986), Gaschutz and Ahrens (1968), Tompkins and Reed (1976), Foulds (1983), and Khalil (1973). More comparisons among CRAFT, the Hillier procedure, and some other random sampling schemes can also be found in Ritzman (1972).

2.3.3. Hybrid Methods

Methods in this class combine several features of exact and inexact procedures. According to several computational experiences reported in the literature, it seems that these methods are emerging as one of the most successful approaches for solving large QAPs. Examples of such procedures are the methods of Bazaraa and Sherali (1980) and Burkard and Stratmann (1978). Both methods use an exact solution scheme in conjunction with some improvement procedures. Bazaraa and Sherali implement Benders' partitioning method to a mixed-integer formulation of the problem and apply several improvement procedures to the
solutions found throughout the course of partitioning. The method of Burkard and Stratmann alternates between a branch and bound (perturbation) routine and an exchange routine (VERBES) until no better solution can be obtained. Bazaraa and Kirca (1983) modified a single assignment branch and bound procedure to include the use of several improvement routines in order to accelerate the computations resulting in an efficient heuristic procedure. The extent of using these improvement routines is a function of the branch-and-bound tree level. Several heuristics are utilized to eliminate the search effort at branches which are likely not to lead to objective value improvements. Furthermore, variable upper bounds are used to reduce the number of solutions examined. Another hybrid method can also be found in Heider (1973) where a search tree terminates N steps after a feasible solution is found. The algorithm is applied to solve the component placement problem described in Steinberg (1961).

2.3.4. Simulated Annealing

Simulated annealing is an improvement procedure bearing some similarity to the biased random sampling approach. Monte Carlo sampling is used occasionally to accept solution to discrete minimization problems which increase rather than decrease the objective value. Thus, the
process seeks another neighborhood space randomly instead of getting "stuck" on inferior solutions produced by the normal "steepest-descent" algorithmic approach. Wilhelm and Ward (1987) presented an application of the simulated annealing method to solve the QAP. It was shown that under certain conditions simulated annealing can yield higher quality (lower cost) solutions at reasonable CPU times compared to those from other traditional heuristics (CRAFT, biased random sampling, and the revised Hillier procedure).

2.4. Extensions of the QAP

A comprehensive review of applications of the QAP can be found in Sherali and Rajgopal (1986). Elshafei (1977) discusses an application in the context of a hospital layout design. Other popular applications are the production scheduling problem of Geoffrion and Graves (1976) and the backboard wiring problem of Steinberg (1961), the problem of scheduling classes and examinations in an educational system as formulated by Carlson and Nemhauser (1966) and the problem of minimizing latency in magnetic drums on disc storage computers as discussed by Lawler (1963).

Recently, several extensions of the QAP have appeared, such as dynamic QAP by Rosenblatt (1986), stochastic QAP by Rosenblatt and Lee (1987), multiple objective programming
for the QAP by Malakooti and D'Souza (1987) and Dutta and Sahu (1982) and the multiperiod assignment problem by Aronson (1986). Finally, computer programs of both exact and some inexact methods in FORTRAN are presented in Burkard and Derigs (1980).
CHAPTER III
EXACT PROCEDURES FOR THE GENERAL QAP

In this chapter, exact procedures for the general QAP will be described in more detail. As described in Chapter 2, these are the Gilmore-Lawler method, the method of Kaku and Thompson, the Land-Gavett-Plyter method and the pair exclusion method of Pierce and Crowston. Each method will be explained and illustrated by solving a given QAP including discussions on previous computational results.

3.1. The Gilmore-Lawler Method

Gilmore (1962) and Lawler (1963) introduced the first exact procedure for the QAP. Gilmore focused this method on the Koopmans-Beckmann formulation while Lawler was concerned with the general form. The method can be organized as follows. Consider the general form of the objective function of the QAP.

\[ Z = \sum_{i} \sum_{j} \sum_{k} \sum_{L} C_{ijkl} X_{ij} \cdot X_{kL} \cdot \]
Let

$$L_{ij} = \min \left[ \sum_{k \in S} \sum_{L_{ij}} C_{ijkl} X_{kL} \right], \; X_{kL} \in S, \; \text{for all } k, L,$$

where $S$ is the set of all feasible assignments. Let

$$L^* = \min \left[ \sum_{i \in S} \sum_{j \in S} L_{ij} X_{ij} \right], \; X_{ij} \in S, \; \text{for all } i, j.$$

This is a linear assignment problem. Let $X^*$ be the optimal solution to this problem with objective value $L^*$, and $Z^*$ be the objective value for the QAP with the solution $X^*$. Now let $L^* \leq Z^*$ since $L_{ij}$ is chosen to be the minimum. Hence, $L^*$ is a lower bound which can be computed by solving $n^2 + 1$ linear assignment problems of size $n$. Gilmore and Lawler used this method incorporated into the single assignment tree to find the optimal solution of the QAP. During the tree search process, $S$ and $L_{ij}$ are updated by the restriction of partial assignments and the process terminates when the lower bounds of all active nodes are greater than or equal to the upper bound (the best objective value found so far).

As an illustration, we will solve the Koopmann-Beckmann problem of Gavett and Plyter (1966) shown in Figure 3, computing bounds according to $L^*$. 
### Figure 3. Flow and distance for illustrative problem.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(FROM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td></td>
<td>1</td>
<td>28</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td></td>
<td>2</td>
<td>28</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td></td>
<td>3</td>
<td>25</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td></td>
<td>4</td>
<td>13</td>
<td>4</td>
<td>23</td>
</tr>
<tr>
<td>TO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(TO)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>28</td>
<td>25</td>
<td>13</td>
</tr>
</tbody>
</table>
From Section 1.1 of Chapter 1,

\[ C_{ijkl} = f_{ik} \cdot d_{jL} \]

(no linear cost involved).

In this case, let \( i, k = A, B, C, D \) and \( j, L = 1, 2, 3, 4 \). Now, consider \( L_{A1} \) and let

\[ A_{1L} = \frac{C_{AlkL}}{3_{L}} \text{ for all } k, L, \]

and \( C_{AlkL} = f_{Ak} \cdot d_{1L} \). Therefore,

\[
\begin{array}{cccc}
  & 1 & 2 & 3 & 4 \\
\hline
A & 0 & -- & -- & -- \\
B & -- & 168 & 150 & 78 \\
C & -- & 196 & 175 & 91 \\
D & -- & 56 & 50 & 26 \\
\end{array}
\]

Next, solve the above linear assignment problem and get the optimal objective value of 297. From the definition of \( L_{A1} \),

\[ L_{A1} = 297 \].
Other $L_{ij}$ elements can be found in the same manner to produce the matrix below.

\[
L = \begin{array}{cccc}
\text{A} & 297 & 174 & 293 & 152 \\
\text{B} & 368 & 254 & 353 & 217 \\
\text{C} & 244 & 131 & 245 & 116 \\
\text{D} & 156 & 82 & 161 & 73
\end{array}
\]

Solving the linear assignment problem defined by this matrix gives us $L^* = 792$, which is the initial lower bound. Next, the single assignment tree search process is started at the first level. Let us consider $A-1$ as the partial assignment as illustrated below.

![Lower Bound = 792](image)

Now consider

\[
C_{kL}^{B2} = [C_{B2kL}], \text{ for all } k,L.
\]
In this case $C_{B2kL} = f_{Bk} d_{2L}$ for all $k, L$ with the additional constraint that $k = A$ if and only if $L = 1$ and $C_{B2A1} = f_{BA} d_{21}$. Therefore,

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>168</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>B</td>
<td>--</td>
<td>0</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>C</td>
<td>--</td>
<td>--</td>
<td>75</td>
<td>20</td>
</tr>
<tr>
<td>D</td>
<td>--</td>
<td>--</td>
<td>90</td>
<td>24</td>
</tr>
</tbody>
</table>

Then we have $L_{B2} = 267$ (the objective value of the linear assignment problem with the above cost matrix) + 168 ($C_{A1B2}$ due to partial assignment $A-1$) = 435. In a similar fashion, the other elements of this subproblem (node $A-1$) can be determined and shown as follows.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>B</td>
<td>--</td>
<td>435</td>
<td>505</td>
<td>295</td>
</tr>
<tr>
<td>C</td>
<td>--</td>
<td>427</td>
<td>448</td>
<td>225</td>
</tr>
<tr>
<td>D</td>
<td>--</td>
<td>151</td>
<td>213</td>
<td>99</td>
</tr>
</tbody>
</table>

Solving this linear assignment problem, we obtain a lower bound for node $A-1$ of 873 ($L^*$). In the same way, the level 1 of the enumeration tree is developed as follows.
Then, node A-2 with the smallest lower bound is selected to develop first at level 2. Now the partial assignments are A-2 and B-1 or B-3 or B-4. By repeating the procedure in the same manner at levels 2, 3 and 4, the complete search tree for this problem is as shown in Figure 4 with the optimal assignment of A-2, B-4, C-3, D-1 and $Z^* = 806$.

From the procedure previously described, the lower bound in each node is obtained by solving at most $n^2 + 1$ linear assignment problems. However, in the special case of the Koopmann-Beckmann formulation, both Lawler and Gilmore have pointed out that the lower bound is more easily obtained by matching the largest value of $f_{ik}$ with the smallest $d_{jl}$, the next largest $f_{ik}$ with the second smallest $d_{jl}$ and so on under restrictions of partial assignments. For fixed $i$ and $j$, $L_{ij}$ can be obtained faster and only one linear assignment problem (L) is solved in each subproblem (node).
Figure 4. Tree completed for the problem of Figure 3 using the Gilmore-Lawler procedure.
3.2. The Kaku-Thompson Method

Kaku and Thompson (1986) presented a binary tree search for the general QAP. The initial lower bound is calculated by the same method used by Gilmore (1962) and Lawler (1963). The branching rule uses the concept of alternate cost method as suggested by Little et al. (1963). Let us say that the linear assignment problem (L) is solved resulting in an optimal cost of \( L^* \) with some partial assignment restrictions. The dual variables of the optimal solution \((u_i, v_j)\) are used to reduce \( L \), i.e., by replacing \( L_{ij} \) by \( L_{ij} - u_i - v_j \). For every \( i \) and \( j \) which has not been assigned and for which \( L_{ij} = 0 \), the next smallest element in row \( i \) and column \( j \) is found and their sum determined. The resulting amount is the minimum additional cost if assignment \((i, j)\) is not made. Also, this value plus the lower bound \((L^*)\) gives the alternate cost of this assignment. In the next level of the decision tree, the assignment which has the maximum alternate cost is made and the lower bound \((L^*)\) is computed by using the method presented in Section 3.1. While backtracking, if the alternate cost is greater than the best known solution (upper bound), the node can be fathomed and the algorithm backtracked to the next higher level. Otherwise, the next node in that level is developed by blocking out the previous assignment in
this level and resolving the LAP (L) and using the alter­
nate cost method to find the next assignment. The process
terminates when the algorithm backtracks to level 0 (start­
ing node).

From the above procedure, the algorithm can be pre­
sented as follows.

STEP 0. Initialization: set up subproblem $L_{ij}$. $k = 0$, $UB = \infty$.

STEP 1. Solve the subproblems to obtain the values $L_{ij}$ and
solve L to get the initial lower bound $L^*$. Reduce
$[L_{ij}]$ by the dual variables $u_i$ and $v_j$ and calcu­
late the alternate cost. Choose an assignment
with the maximum alternate cost, say P to Q.

STEP 2. Assign $X_{pq} = 1$, $k = k + 1$ and set P -> Q as a
partial assignment. Calculate the lower bound
($L^*$) with the partial assignment restrictions. If
$L^* \geq UB$, go to Step 5. Otherwise, if $k = n - 2$, go to Step 4. Otherwise, go to Step 3.

STEP 3. Use the dual variables to reduce L and choose the
next assignment, say P and Q. If backtracking,
choose the next assignment by blocking out the
previous assignment in this level. Calculate the
alternate cost. Go to Step 2.
STEP 4. Calculate exact cost of two remaining completions. If total cost < UB, update UB and state assignments.

STEP 5. If K = 0, stop. An optimal solution has been found. Otherwise, K = K - 1 and if the alternate cost ≥ UB, repeat Step 5. Otherwise, determine all C_{ijPQ}, C_{PQij} = ∞ and solve relevant subproblems. Then set \( L_{PQ} = ∞ \). Solve L to get a tighter \( L^* \). If \( L^* ≥ UB \), repeat Step 5. Otherwise, go to Step 3.

Again, the problem of Figure 3 is solved by the above algorithm. In this procedure, the alternate cost method will be reviewed.

The method begins with the first L matrix developed previously in Section 3.1 and the lower bound on all solutions = 792. The reduced L matrix \((L_{ij} - u_i - v_j)\) obtained from solving L by the Hungarian algorithm can be shown below.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>14</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>20</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>48</td>
</tr>
</tbody>
</table>
Candidates for the next assignment are A-2, A-4, B-3, B-4, C-2, C-3, D-1 and D-3 (zero elements in the above matrix). The alternate cost of each zero element is equal to the lower bound plus the sum of the next smallest element in the row and column of that element. For example, consider D-1 element, the next smallest element in row D and column 1 is 0 and 4, respectively. Therefore, the alternate cost for this element = 792 + 4 = 796. By examining all zero elements in the same way, the alternate cost of D-1 is the highest so D-1 is used to branch first.

Next, the lower bound at node D-1 is computed by using the Gilmore-Lawler method.

\[
L = \begin{array}{|c|cccc|}
\hline
 & 1 & 2 & 3 & 4 \\
\hline
A & -- & 230 & 343 & 218 \\
B & -- & 435 & 505 & 295 \\
C & -- & 159 & 270 & 169 \\
D & 0 & -- & -- & -- \\
\hline
\end{array}
\]
Solving the above linear assignment problem, \( L^* = 795 \) with the following reduced \( L \).

\[
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
A & \text{--} & 0 & 2 & 0 \\
B & \text{--} & 128 & 87 & 0 \\
C & \text{--} & 0 & 0 & 22 \\
D & 0 & \text{--} & \text{--} & \text{--} \\
\end{array}
\]

Now, candidates for the next assignment are A-2, A-4, B-4, C-2 and C-3. D-1 is not included because it is already a partial assignment. The assignment with the largest alternate cost is B-4 with 882. Therefore, the decision tree branch on B-4 is as follows.

\[
\begin{array}{c}
\text{D-1} & 795 \\
\end{array}
\]

\[
\begin{array}{c}
882 & \begin{array}{c}
\text{B-4} \\
\text{B-4} & 796 \\
\end{array}
\end{array}
\]

Then, all possible alternatives can be enumerated at node B-4.
where the lower bound 796 is computed from the $L$-matrix with the restrictions $D-1$ and $B-4$. At this point, the backtracking process is started and it is found that node $D-1$ has the lower bound less than 806 (UB). Therefore, the procedure looks for the next assignment by blocking out $D-1$ in the corresponding $L$ matrix as follows.

\[
L = \begin{array}{cccc}
1 & 2 & 3 & 4 \\
A & 297 & 174 & 293 & 152 \\
B & 368 & 254 & 353 & 217 \\
C & 244 & 131 & 245 & 116 \\
D & -- & 82 & 161 & 73 \\
\end{array}
\]

Solving the above problem, $L^* = 796$ with the reduced $L$ matrix as follows.
In this case, the highest alternate cost is 831 on D-3.

The tree search branch is as below.

Then, the lower bound $L^*$ with the restriction of D-3 is computed and the process continues in the same manner. Finally, the complete search tree is illustrated in Figure 5.

Kaku and Thompson (1986) showed that their method was capable of solving to optimality problems of sizes up to $n = 10$ in reasonable computation times.

### 3.3. The Land-Gavett-Plyter Method

In contrast to the algorithm discussed in the previous section, both Land (1963) and Gavett and Plyter (1966) have developed algorithms which search solutions by the assignment of variables $Y_{ijkL} = X_{ij} \cdot X_{kL}$ where as before each
Figure 5. Tree elaborated for problem of Figure 3 (by the Kaku-Thompson method).
Xij denotes the locating of object i at location j. These authors also developed the subproblem as a linear assignment problem, but one of assigning a pair of objects i and k to locations j and L, respectively. In both papers, the algorithms developed apply to the symmetric Koopmann-Beckmann problem with $C_{ijkl} = f_{ik} \cdot d_{jL}$. Nevertheless, it will be shown that this method can be modified to handle the general QAP.

In Figure 6, the relevant pair assignment for the problem of Figure 3 is shown. In general, there are $n(n-1)/2$ pairs of objects and pairs of locations in the problem.

However, there are many feasible solutions to the above linear assignment problem which are not feasible to the original QAP. For example, assigning A-B to 1-2 and A-C to 3-4 which is clearly infeasible for the QAP are acceptable in the above solution. For a feasible solution to the QAP, the problem becomes a linear assignment problem with the following additional constraints.
<table>
<thead>
<tr>
<th>LOCATION PAIR</th>
<th>2-1</th>
<th>3-1</th>
<th>4-1</th>
<th>3-2</th>
<th>4-2</th>
<th>4-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBJECT PAIR</th>
<th>A-B</th>
<th>A-C</th>
<th>A-D</th>
<th>B-C</th>
<th>B-D</th>
<th>C-D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>168</td>
<td>196</td>
<td>56</td>
<td>140</td>
<td>168</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>175</td>
<td>50</td>
<td>125</td>
<td>150</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>78</td>
<td>91</td>
<td>26</td>
<td>65</td>
<td>78</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>105</td>
<td>30</td>
<td>75</td>
<td>90</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>28</td>
<td>8</td>
<td>20</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>138</td>
<td>161</td>
<td>46</td>
<td>115</td>
<td>138</td>
<td>23</td>
</tr>
</tbody>
</table>

Figure 6. Data for the problem of Figure 3 represented in terms of pairs of assignments.
IF $Y_{ijkl} = 1$ \hspace{1cm} (3.1)

THEN $Y_{abcd} = 0$, $Y_{cbid} = 0$

$Y_{ajcd} = 0$, $Y_{adcj} = 0$

$Y_{abkd} = 0$, $Y_{kbad} = 0$

$Y_{abcL} = 0$, $Y_{aLcb} = 0$ ,

where $i \neq a$, $j \neq b$, $k \neq c$ and $L \neq d$. Operationally, both the algorithm of Land and that of Gavett and Plyter solve the linear pair assignment problem and determine a reduced matrix with non-negative entries of $C_{ijkl} - u_{ik} - v_{jL}$, where $C_{ijkl}$ is the cost element of object pair $k,L$ and location pair $j,L$, $u_{ik}$ and $v_{jL}$ are the dual variables of object pair $i,k$ and location pair $j,L$, respectively.

Thereafter, the binary search tree as described by Little (1963) is used to search for the optimal solution. To obtain the lower bound, Gavett and Plyter employ the row-column reduction method at each node while Land employs only the column reduction method at each node. As previously discussed in Section 3.2, both of the algorithms use the binary search tree which proceeds each level by assigning $Y_{ijkl} = 1$ to the solution and backtracks to the lowest level in the tree having an unevaluated branch. In selecting the pair to be committed at a given level in the search tree, Gavett and Plyter use the alternate cost method as
described in Section 3.2, while Land always selects from the column having the least number of feasible elements in the column-reduced matrix (a zero element having the largest alternate cost based only on alternate cost in the same column). After committing a pair assignment to the solution at a given node of the search tree, the condition of $Y_{ijkL} = 1$ is involved by setting $C_{abcd} = \infty$ for all $Y_{abcd} = 0$ specified previously to obtain the resulting cost matrix used at the next level.

At this point, the Gavett-Plyter algorithm is illustrated by solving the problem of Figure 3. First, the linear pair assignment problem of Figure 6 is solved and the optimal cost found is 389 with the following reduced matrix (due to the symmetry of $C_{ijkL}$ in this case, the lower bound obtained is one-half of the exact value).

$$
\begin{array}{ccccccc}
1-2 & 1-3 & 1-4 & 2-3 & 2-4 & 3-4 \\
2-1 & 3-1 & 4-1 & 3-2 & 4-2 & 4-3 \\
\hline
A-B & 31 & 16 & 0^* & 0^* & 0^* & 8 \\
A-C & 55 & 37 & 9 & 11 & 0^* & 27 \\
A-D & 3 & 0^* & 32 & 24 & 68 & 0^* \\
B-C & 18 & 6 & 2 & 0^* & 11 & 0^* \\
B-D & 31 & 16 & 0^* & 0^* & 0^* & 8 \\
C-D & 0^* & 0^* & 44 & 34 & 89 & 2 \\
\end{array}
$$
From the above matrix, candidates for the pair assignment at level 1 are (A-B,1-4), (A-B,2-3), (A-B,2-4), (A-C,2-4), (A-D,1-3), (A-D,3-4), (B-C,2-3), (B-C,3-4), (B-D,1-4), (B-D,2-3), (B-D,2-4), (C-D,1-2) and (C-D,1-3). The highest alternate cost is found in (A-C,2-4) with 398. Therefore, the level 1 of the search tree is shown as below.

![Search Tree Diagram]

By continuing in the same manner, the complete search tree can be elaborated in Figure 7 as follows.

In comparison with other search strategies previously discussed, this strategy may require a longer path in the decision tree and a longer problem solution time to determine a first feasible solution. From a computational viewpoint, both Land and Gavett and Plyter are able to solve the QAP optimally up to size n = 8.

We now consider the extension of this class of algorithms to the general case in which \( C_{ijkl} \neq C_{iklj} \). For this problem, two approaches can be used.
Figure 7. Tree elaborated for the problem of Figure 3 by the Gavette-Plyter algorithm.
The first approach is to convert

$$\sum_{i} \sum_{j} \sum_{k} C_{ijkL} X_{ij} X_{kL}$$

to a scalar form of $X^T C X$, where $C$ is the assignment-to-
assignment matrix of size $n^2$. Then, this $C$-matrix is used
to compute the initial lower bound and the dual variables
in the search tree instead of the pair assignment matrix.

The second approach is to consider the following
associated linear assignment problem.

Minimize

$$\sum_{ijkL} (C_{ijkL} \cdot Y_{ijkL} + C_{iLkj} \cdot Y_{iLkj})$$

subject to

$$\sum_{j,L} (Y_{ijkL} + Y_{iLkj}) = 1$$

$$\sum_{i,k} (Y_{ijkL} + Y_{iLkj}) = 1$$

$Y_{ijkL}, Y_{iLkj} = \{0, 1\}, \quad i,j,k,L.$

By setting $C_{ijkL} = \min(C_{ijkL}, C_{iLkj})$ and $Y_{ijkL} =$
$Y_{ijkL} + Y_{iLkj}$, it follows that the lower bound to the
original QAP can be obtained by solving the above linear
assignment problem of size $n(n-1)/2$ with cost elements
Cijkl. Again, the binary search tree previously described in this section can be used cooperatively with the above formulation in order to find the optimal assignment.

3.4. Pair-Exclusion Algorithm

In all of the algorithms previously discussed, the solution process has proceeded on the basis of an assignment in each level of the search tree. Upon backtracking, a particular assignment would then be excluded from the solution and the included assignment process resumed. This has been the nature of the search tree of both the single assignment and the pair assignment. In this section, the algorithm of Pierce and Crowston (1971) which proceeds on the basis of an exclusion of assignments from a solution in each stage of the search tree is presented.

Operationally, the algorithm is similar to the one used in solving the traveling salesman problem by Eastman (1958). It will be illustrated with the problem of Figure 3 as follows.

First, consider the reduced matrix formed from the pair assignment matrix shown in Figure 6 and the optimal assignment which is represented by a zero element each row and column of this matrix. From Section 3.3, in this case, the optimal assignment is (A-B,1-4), (A-C,2-4), (A-D,1-3),
(B-C,3-4), (B-D,2-3) and (C-D,1-2) where (i-k,j-L) represents object pair i and k being assigned to location j and L. Next, \( n(n-1)/2 \) subproblems are generated by prohibiting each pair assignment in this solution. The linear assignment problem associated with each subproblem (node) is solved and stored as a lower bound. This process proceeds until a feasible solution to the QAP found has the upper bound less than or equal to any lower bound of all active subproblems (have not been branched or fathomed). For this example, the result in terms of a tree is partially shown in Figure 8.

In a similar manner, the process can now proceed by selecting any active node with the lowest bound, solve the associated linear assignment problems and check feasibility of the solution. Then, continue until the optimality condition (the upper bound \( \geq \) all lower bounds) is verified.

In general, it is difficult to anticipate the performance of this type of algorithm relatively to the types as discussed in the earlier section. For the related traveling salesman problem, this general approach has proved significantly more efficient than level-by-level assignment algorithms. Undoubtedly, this is basically due to the fact that the optimal traveling salesman problem is frequently quite close to the optimal linear assignment solution in
Figure 8. Partial tree elaborated by the pair exclusion algorithm.
the respect that a large majority of assignments in the former are present in the latter, so that a relatively small decision tree need be explicitly elaborated. In the QAP, it does not appear that the optimal quadratic assignment is close to the optimal linear pair assignment (Pierce and Crowston, 1971). Therefore, a much larger search tree is required (even to determine a first feasible solution).

3.5. Computation Issues

Kaku and Thompson (1986) reported a comparison between their method (Section 3.2) and the Gilmore-Lawler method (Section 3.1). The results showed that their method performs better, especially as the size of the problem grows. Nevertheless, the largest size of problem reported is only \( n = 10 \) and the average run time is over 20 minutes on a mainframe DEC-20. Burkard and Gerstl (1973) modified the Land-Gavett-Plyter method and reported that this method is competitive with the Gilmore-Lawler method until the size of the QAP is up to \( n = 8 \). So far, no computational comparison has been made for the pair exclusion method of Section 3.4. Technically, this method has a memory management problem. As illustrated in Figure 8 of Section 3.4, the search tree of this method has no structure to aid memory management in the computer program developed (i.e., no certain value of the depth of the search tree) while the
search trees described in Sections 3.1, 3.2 and 3.3 can utilize the strategy presented in Appendix A. Evidently, at present, the binary tree of the single assignment as described by Kaku and Thompson (1986) seems to be the most effective way to solve the QAP. This is also claimed by Pierce and Crowston (1971) when they proposed a similar search tree with no computational results.

In the next chapter, a new branch and bound technique for a general QAP will be presented. The method utilizes the lower bound obtained from solving the linear pair assignment problem as described in Section 3.3 combined with a sum of dual variables contributed by partial assignment to obtain a new lower bound and the single assignment tree to search for the optimal solution.
In Chapter 3, four branch and bound methods for the general QAP have been described. Based upon computational experiences, all methods require a large amount of computation to obtain a lower bound in each node (subproblem). It is well known that all tree search techniques for the QAP requires an exponential time to verify an optimal solution (i.e., the QAP is NP-complete). In view of the time consumed by tree enumerations and computations of the lower bound, it has become prohibitive to solve even a small scale QAP \((n > 10)\). In this chapter, another approach with a weaker lower bound which requires much less time to obtain will be proposed. The trade-off between the number of subproblems generated and strength of the lower bound will be studied in Chapter 5 by comparing it with the binary tree search of Kaku and Thompson and the complete enumeration method.

The proposed algorithm uses the single assignment tree and the optimal solution of the linear-pair assignment matrix previously described in Section 3.3 as the initial
lower bound. The lower bound in each node is computed by adding the dual variables (obtained in the first step) of the relevant partial assignments to the initial lower bound. Mathematically, consider the following formula,

\[ \text{LB}(I,\phi(I)) = \text{LB}_0 + \sum_{i,\phi(i)} \sum_{k,\phi(k)} [C_{i\phi(i)k\phi(k)} - U_{ik} - V_{\phi(i)j,\phi(k)}] \]

(4.1)

where

\text{LB}_0 is the initial lower bound,

\text{LB}[I,\phi(I)] is the lower bound of the subproblem (node) with partial assignment \([I,\phi(I)]\),

\(C_{i\phi(i)k\phi(k)}\) is the cost of assigning \(i\) to \(\phi(i)\) and \(k\) to \(\phi(k)\), \(i,k \in I, \phi(i),\phi(k) \in \phi(I)\),

\(U_{ik}\) is the dual variable of the paired objects \(i\) and \(k\) (obtained from solving the initial linear pair assignment problem), and

\(V_{\phi(i)j,\phi(k)}\) is the dual variable of paired locations \(\phi(i)\) and \(\phi(k)\).

In the next part, a linear pair assignment problem (LPAP) will be determined with objective function which
bounds the objective function of the general QAP. Furthermore, equation 4.1 will be modified and illustrated the bounding condition for the QAP.

4.1. The Bounding Linear Pair Assignment Problem

For \( i, j = 1, 2, \ldots, n \) and \( k = i+1, i+2, \ldots, n \) and \( L = j+1, j+2, \ldots, n \) denoted by \( C_{ijkl} \), an element in the pair-assignment matrix \( P \), let \( P^* \) be the minimum value of the objective function of the following problem:

Minimize

\[
\sum_{ik} \sum_{jL} C_{ijkl} Y_{ijkl} = P(Y) \quad \text{(PR1)}
\]

subject to

\[
\sum_{ik} Y_{ijkl} = 1, \text{ for all } j, L \\
\sum_{jL} Y_{ijkl} = 1, \text{ for all } i, k \\
Y_{ijkl} = \{0, 1\}
\]

It is assumed that this LPAP has finite optimal solutions. Recall from Section 3.3 of Chapter 3 that \( Y_{ijkl} = \)
X_{ij} \cdot X_{kL}$. The following proposition provides the motivation for referring to PR1 as the bounding linear assignment problem for the general QAP.

**Proposition 1**

For any $X = (X_{ij}) \in S$ and $Y^* = (Y^*_{ijkL})$,

$$P(Y^*) \leq f(x)$$

where

$Y^*$ is the optimal solution of PR1;

$$f(x) = \sum_{i} \sum_{j} \sum_{k} \sum_{L} C_{ijkL} X_{ij} \cdot X_{kL}$$

and $S$ is the set of all possible assignments.

Proposition 1 is readily proven using the fact from Section 3.3 that the QAP can be formulated as PR1 plus the constraints described in condition 3.1. In fact, $LB_0 = 2 \cdot P(Y^*)$ in equation 4.1 and an immediate consequence of proposition 1 is the following corollary.

**Corollary 1**

If $f^*$ is the optimal objective value to the QAP, then $LB_0 = P(Y^*)$ is a lower bound on $f^*$. 
Next, for a set of partial assignments \( \{i, \phi(i)\} \), \( i \in I \), \( \phi(i) \in \phi(I) \) where I is the set of partial objects and \( \phi(i) \) is the position where object i is located, consider the following proposition.

**Proposition 2**

\( LB[i, \phi(i)] \) obtained from equation 4.1 is a lower bound for a node of the search tree with partial assignments \([I, \phi(I)]\).

**Proof**

First, consider the term \( C_{i\phi(i)k\phi(k)} - U_{ik} - \sum_{\phi(i)\phi(k)} V_{\phi(i)\phi(k)} \)

\( = \overline{C}_{i\phi(i)k\phi(k)} \) and \( i, k \in I \). Due to proposition 1 and the fact that from the reduction step of the linear assignment problem, at least \( 2 \cdot \overline{C}_{i\phi(i)k\phi(k)} \) is contributed to the optimal objective value of PR1 with the additional constraints that \( X_{i\phi(i)} \) and \( X_{k\phi(k)} = 1 \). Thus,

\[ LB_0 + 2 \cdot \overline{C}_{i\phi(i)k\phi(k)} \leq f^*[I, \phi(I)] , \]

where \( f^*[I, \phi(I)] \) is the optimal objective value to the QAP with partial assignment \( I = \{i,k\} \) and \( \phi(I) = \{\phi(i), \phi(k)\} \). Similarly for any \([I, \phi(I)]\),

\[ LB_0 + \sum_{i, \phi(i)} \sum_{k, \phi(k)} \overline{C}_{i\phi(i)k\phi(k)} \leq f^*[I, \phi(I)] \]
which is a lower bound for a node of the search tree with partial assignment \([I, \phi(I)]\).

In this case, the symmetry of \(C_{ijkl}\) (\(C_{ijkl} = C_{klij}\) and \(C_{ijkl} = C_{klij}\)) is assumed. For a general \(C_{ijkl}\), two approaches suggested in Section 3.3 can be modified and obtained the same results. For instance, if \(C_{ijkl} = C_{klij}\), set \(C_{ijkl} = (C_{ijkl} + C_{klij})/2\) and if \(C_{ijkl} = C_{kjiL}\), set \(C_{ijkl} = \min(C_{ijkl}, C_{kjiL})\); then substitute \(C_{ijkl}\) to \(C_{kjiL}\) and analyze the results. Therefore, any QAP can be converted to a symmetric QAP.

### 4.2. A Generalized Lower Bounding Equation

In this section, we will introduce a new lower bounding equation bearing different from equation 4.1. It is determined by using the symmetry of \(C_{ijkl}\), the level of the search tree and partial assignment \([I, \phi(I)]\). First, let us consider the following relationship,

\[
LB_i[I, \phi(I)] = LB_{i-1}[I, \phi(I)] + \sum_{k,L \in [I, \phi(I)]} 2 \cdot C_{pqkL}, \quad (4.2)
\]

where

- \(i\) is the level in the single assignment search tree,
- \(I = I - (P)\) and \(\phi(I) = \phi(I) - (q)\) where \((p, q)\) is the object \(p\) and location \(q\) assigned at level \(i\) of the search tree;
LB^I_i(I, \phi(I)) is the lower bound at level \( \hat{i} \) of the search tree with partial assignment \([I, \phi(I)]\); and

\( \overline{c}_{pqk} \) is the same as described previously in Section 4.1.

It is obvious that equation 4.2 requires less computation than equation 4.1 but it is not immediately clear as to how \( LB^\hat{i}_i(I, \phi(I)) \) is related to \( LB[I, \phi(I)] \). Such relationship can be illustrated by the following proposition.

Proposition 3

\( LB^\hat{i}_i(I, \phi(I)) = LB(I, \phi(I)) \).

Proof

Let \( \hat{i} = 0 \), \( LB_0[I, \phi(I)] = LB[I, \phi(I)] = LB_0 \) with \( I = \phi \) and \( \phi(I) = \{\phi\} \). Next, assume \( (p_1, q_1) \) is assigned at level \( \hat{i} = 1 \). Therefore, from equation, 4.1 and 4.2, \( LB_1[I, \phi(I)] = LB[I, \phi(I)] = LB_0 \) with \( I = \{p_1\} \) and \( \phi(I) = \{q_1\} \). Similarly, assume \( (q_2, q_2) \) is assigned at level \( \hat{i} = 2 \). Thus, \( I = \{p_1, p_2\} \) and \( (I) = \{q_1, q_2\} \) with

\[
LB_2[I, \phi(I)] = LB_0 + 2\overline{c}_{p_1q_1p_2q_2} = LB(I, \phi(I)).
\]

In the same manner for \( \hat{i} = 3 \), \( I = \{p_1, p_2, p_3\} \) and \( \phi(I) = \{q_1, q_2, q_3\} \), with
\[ LB_3[I, \phi(I)] = LB_0 + 2[\overline{c}_{P_1q_1P_2q_2} + \overline{c}_{P_1q_1P_3q_3} + \overline{c}_{P_2q_2P_3q_3} \]
\[ = LB[I, \phi(I)]. \]

In general, for \( i = n, I = (p_1, ..., p_n) \) and \( \phi(I) = (q_1, ..., q_n) \) with

\[ LB_i[I, \phi(I)] = LB_0 + \sum_{i=1}^{2} \sum_{j=1}^{n} \sum_{k=i+1}^{n} \sum_{L=j+1}^{n} \overline{c}_{i,j,k,L} \]
\[ = LB_0 + \sum_{i,j,k,L \in [I, \phi(I)]} \overline{c}_{i,j,k,L} = LB[I, \phi(I)]. \]

Now, for \( i = n + 1 \), it is obvious that \( LB_i[I, \phi(I)] = LB[I, \phi(I)] \). Hence, the proposition.

We can now formally state that equation 4.2 provides a lower bound for partial assignments for my QAP. By comparisons with the methods for calculating lower bounds presented in Chapter 3, equation 4.2 requires much less work, while (to be shown in Chapter 5) its lower bounds are weaker. Therefore, it requires more nodes (subproblems) to be developed in the search tree. The relative efficiency of these two approaches will be discussed based upon some computational experiences presented in Chapter 5.

4.3. The Branch and Bound Procedure

The algorithm starts by setting the upper bound value to a large number and solving the LPAP to obtain the
initial lower bound. Then, the single assignment tree proceeds level by level by assigning an available object to an available location and computing the corresponding lower bound until a complete assignment is found and the upper bound is updated. After that, the process backtracks up the search tree level. If all remaining branches have been fathomed, the process stops and an optimal solution is found. Otherwise, the highest level with the lowest bound is chosen to continue the search tree. This type of backtracking process is usually referred to as "depth first search." Formally, this algorithm can be stated as follows.

STEP 1: Set \( \hat{i} = 0 \), UB (the upper bound) = M (large value). Then, solve the corresponding linear pair assignment and obtain the initial lower bound LB_0.

STEP 2: At level \( \hat{i} \) of the search tree (Figure 1 or Figure 2), define possible assignment at each node and compute the lower bound using equation 4.2. If \( \hat{i} = n \) (number of objects or locations), then update the upper bound and go to Step 4. Otherwise, go to Step 3.

STEP 3: Select the node which has the lowest bound among nodes generated at level \( \hat{i} \) if this bound is smaller than the upper bound, then choose this node as
a partial assignment, set \( \hat{i} = \hat{i} + 1 \) and repeat Step 2. Otherwise, \( \hat{i} = \hat{i} - 1 \) and go to Step 4.

**STEP 4:** Backtracking: If \( \hat{i} = 0 \), then an optimal solution is found and verified. Otherwise, remove the previous partial assignment assigned in level \( \hat{i} \) and choose an available node with the next lowest bound (which is smaller than the upper bound) in this level as the new partial assignment. Set \( \hat{i} = \hat{i} + 1 \) and repeat Step 2. If there is no such node, \( \hat{i} = \hat{i} - 1 \) and repeat Step 4.

### 4.4. Numerical Example

To illustrate how the algorithm presented in Section 4.3 works, the problem shown in Figure 3 will be solved. To accommodate larger problems, a computer program was coded in QBASIC. More computational experiences using this program will be discussed in Chapter 5.

By applying the proposed algorithm step by step, the solution can be obtained as follows. First let us recall the corresponding pair assignment matrix for this problem as shown in Figure 6, Section 3.3. Next, solve the linear assignment problem by the Hungarian algorithm and obtain the final cost matrix shown as follows.
The next step is to use the single assignment tree (in this case, choose the search tree shown in Figure 2) and the above cost matrix with the initial lower bound (778) to find the optimal quadratic assignment. The complete enumeration tree of the branch and bound solution is illustrated in Figure 9 shown as follows.

From the above search tree, the optimal assignment is A-2, B-4, C-3, D-1 with the total cost of 806.

To explain how a lower bound can be computed, suppose the set of partial assignments is \{(A,3),(B,4)\}. From the final cost matrix resulting from solving the LPAP, \(C_{A3B4} = 8\). Therefore, the lower bound in this case is \(778 + 2\cdot8 = 794\). Another example is to compute the lower bound for this set of partial assignment, \{(A,2),(B,4),(C,3)\}. Again, from the final cost matrix, \(C_{A2C3} = 11\), \(C_{B4C3} = 0\). By using equation 4.2, the lower bound for this example is
Figure 9. The complete search tree determined by the proposed algorithm.
778 + 2 \cdot 11 = 800. Similarly, all lower bounds are computed during the search tree proceeds until all nodes are fathomed (all available lower bounds ≤ the upper bound).

The computer program (in BASIC) of this proposed algorithm is developed by using a more elegant way to control the use of memory instead of utilizing the tree type shown in Figure 9. The approach modified from Baker (1974) for the general QAP is explained extensively in Appendix A.

In this chapter, a proposed branch and bound method is presented. The method offers a new formula to obtain a lower bound for a node which requires much less time than any previous methods. Nevertheless, the lower bound obtained is usually weaker and the number of nodes which must be evaluated is greater than the algorithms discussed in Chapter 3. Computational experience with this algorithm and a comparison with the method of Kaku and Thompson is discussed in Chapter 5.
CHAPTER V

COMPUTATIONAL EXPERIENCES

To compare with the algorithm presented in the previous chapter, the method of Kaku and Thompson is coded in QuickBASIC by modifying a FORTRAN program given in Burkard and Derig (1980). Fifty replications for each of six different size levels of the QAP with the Koopmann-Beckmann formulation and the QAP with $C_{ijkl}$ randomly generated were used. These data sets were generated by using the random number generator of the QuickBASIC language. The data input for the Koopmans-Beckmann problem consists of a flow (F) and distance (D) matrix. In this study, $f_{ik}$ and $d_{jL}$ for all $i, j, k$ and $L$ were generated between 0 to 10. For the second group of test problems, the data input consisted of a linear pair assignment matrix with $C_{ijkl}$ randomly generated between 0 to 100. For the QAP, it is well known that the wider the cost range, the more computational time is usually required for most branch and bound methods. Therefore, the absolute computation times obtained are dependent upon the range of costs generated. However, relative time comparisons are valid. The size levels ($n$)
of the problems are 5, 6, 7, 8, 9 and 10. Both algorithms were coded and compiled by the QuickBASIC compiler version 4.0 and run on the COMPAC 386/20 microcomputer. Evaluation of the solution in each replication of each case contains three performance measurements:

1. \( T_1 \) is defined as the time consumed until an optimal solution is verified. An optimal solution is verified when all branches are fathomed.

2. \( T_2 \) is defined as the time consumed until an optimal solution is found. This does not include the time to confirm that remaining branches will be fathomed.

3. \( N \) is defined as the number of nodes generated in branch and bound processes.

\( T_1 \) and \( T_2 \) are estimated in the units of seconds. For convenience, the proposed method and the method of Kaku and Thompson will be referred to as "PM" and "KT," respectively. Additionally, the complete enumeration method, which will be referred to as "CE," will be used to solve all generated problems of each size level to validate solutions found by both methods. The total running times will be used to compare with average \( T_1 \) values obtained by both algorithms.

Results from the experimentation are shown in Table 1 to Table 3. The results consist of the mean value (M), standard deviation (SD), the minimum (MN) and the maximum
Table 1
Statistical Results of $T_1$ (sec.)
(From 50 Replications Each)*

<table>
<thead>
<tr>
<th>(n) Size</th>
<th>GQAP</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PM</td>
<td>KT</td>
<td>PM</td>
<td>KT</td>
<td>CE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>0.212</td>
<td>0.358</td>
<td>0.182</td>
<td>0.172</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>0.076</td>
<td>0.219</td>
<td>0.086</td>
<td>0.160</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>0.200</td>
<td>0.200</td>
<td>0.100</td>
<td>0.100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>0.300</td>
<td>0.800</td>
<td>0.300</td>
<td>0.600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>0.670</td>
<td>2.390</td>
<td>0.432</td>
<td>1.082</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>0.080</td>
<td>0.850</td>
<td>0.157</td>
<td>0.505</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>0.500</td>
<td>1.000</td>
<td>0.200</td>
<td>0.300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>0.800</td>
<td>4.200</td>
<td>0.800</td>
<td>2.500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>1.820</td>
<td>10.560</td>
<td>0.986</td>
<td>5.060</td>
<td>4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>0.384</td>
<td>3.424</td>
<td>0.469</td>
<td>2.500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>1.000</td>
<td>4.000</td>
<td>1.000</td>
<td>2.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>2.000</td>
<td>19.000</td>
<td>2.000</td>
<td>13.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>12.240</td>
<td>67.267</td>
<td>4.680</td>
<td>24.080</td>
<td>25.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>1.305</td>
<td>16.870</td>
<td>1.333</td>
<td>11.793</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>9.000</td>
<td>40.000</td>
<td>2.000</td>
<td>5.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>14.000</td>
<td>108.000</td>
<td>8.000</td>
<td>65.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>105.040</td>
<td>537.380</td>
<td>30.300</td>
<td>124.650</td>
<td>230.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>11.345</td>
<td>150.552</td>
<td>9.214</td>
<td>81.462</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>74.000</td>
<td>177.000</td>
<td>13.000</td>
<td>31.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>133.000</td>
<td>850.000</td>
<td>60.000</td>
<td>362.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>982.620</td>
<td>3886.773</td>
<td>205.233</td>
<td>896.800</td>
<td>2650.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>85.024</td>
<td>874.496</td>
<td>70.255</td>
<td>437.868</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>791.000</td>
<td>1607.000</td>
<td>116.000</td>
<td>164.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>1173.000</td>
<td>5088.000</td>
<td>373.000</td>
<td>2260.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* CE = complete enumeration
GQAP = general case
KBP = Koopman-Beckmann Problem
KT = Kaku-Thompson method
M = mean value
MN = minimum value
MX = maximal value
PM = proposed method
SD = standard deviation
$T_1$ = time consumed until termination
Table 2
Statistical Results of $T_2$ (sec.)*

<table>
<thead>
<tr>
<th>(n) Size</th>
<th>GQAP</th>
<th></th>
<th>KBP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PM</td>
<td>KT</td>
<td>PM</td>
<td>KT</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>0.134</td>
<td>0.146</td>
<td>0.122</td>
<td>0.088</td>
</tr>
<tr>
<td>SD</td>
<td>0.065</td>
<td>0.128</td>
<td>0.064</td>
<td>0.089</td>
</tr>
<tr>
<td>MN</td>
<td>0.100</td>
<td>0.100</td>
<td>0.100</td>
<td>0.050</td>
</tr>
<tr>
<td>MX</td>
<td>0.300</td>
<td>0.500</td>
<td>0.200</td>
<td>0.400</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>0.374</td>
<td>0.994</td>
<td>0.264</td>
<td>0.356</td>
</tr>
<tr>
<td>SD</td>
<td>0.160</td>
<td>0.951</td>
<td>0.129</td>
<td>0.289</td>
</tr>
<tr>
<td>MN</td>
<td>0.100</td>
<td>0.100</td>
<td>0.100</td>
<td>0.100</td>
</tr>
<tr>
<td>MX</td>
<td>0.600</td>
<td>3.300</td>
<td>0.100</td>
<td>1.000</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>1.120</td>
<td>3.920</td>
<td>0.440</td>
<td>1.500</td>
</tr>
<tr>
<td>SD</td>
<td>0.652</td>
<td>4.040</td>
<td>0.496</td>
<td>2.300</td>
</tr>
<tr>
<td>MN</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>MX</td>
<td>2.000</td>
<td>15.000</td>
<td>1.000</td>
<td>11.000</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>6.520</td>
<td>33.267</td>
<td>3.140</td>
<td>7.500</td>
</tr>
<tr>
<td>SD</td>
<td>3.711</td>
<td>21.941</td>
<td>1.625</td>
<td>8.390</td>
</tr>
<tr>
<td>MN</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>MX</td>
<td>13.000</td>
<td>87.000</td>
<td>6.000</td>
<td>32.000</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>53.180</td>
<td>259.640</td>
<td>17.980</td>
<td>59.700</td>
</tr>
<tr>
<td>SD</td>
<td>30.810</td>
<td>189.632</td>
<td>10.542</td>
<td>72.281</td>
</tr>
<tr>
<td>MN</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>MX</td>
<td>120.000</td>
<td>667.000</td>
<td>45.000</td>
<td>269.000</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>540.180</td>
<td>1749.500</td>
<td>105.233</td>
<td>395.800</td>
</tr>
<tr>
<td>SD</td>
<td>345.350</td>
<td>1197.497</td>
<td>69.512</td>
<td>415.900</td>
</tr>
<tr>
<td>MN</td>
<td>1.000</td>
<td>47.000</td>
<td>2.000</td>
<td>1.000</td>
</tr>
<tr>
<td>MX</td>
<td>1076.000</td>
<td>3968.000</td>
<td>263.000</td>
<td>1426.000</td>
</tr>
</tbody>
</table>

*T2 = time consumed until an optimal solution is found
### Table 3

Statistical Results of N (units)*

<table>
<thead>
<tr>
<th>(n) Size</th>
<th>GQAP</th>
<th></th>
<th></th>
<th>KBP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PM</td>
<td>6</td>
<td>KT</td>
<td>PM</td>
<td>7</td>
<td>KT</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>244.94</td>
<td>52.52</td>
<td>172.20</td>
<td>29.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>33.95</td>
<td>25.08</td>
<td>48.02</td>
<td>20.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>164.00</td>
<td>12.00</td>
<td>80.00</td>
<td>8.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>293.00</td>
<td>116.00</td>
<td>288.00</td>
<td>104.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>1,420.86</td>
<td>192.12</td>
<td>815.90</td>
<td>89.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>149.84</td>
<td>88.44</td>
<td>266.12</td>
<td>50.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>923.00</td>
<td>54.00</td>
<td>388.00</td>
<td>16.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>1,668.00</td>
<td>392.00</td>
<td>1,597.00</td>
<td>238.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>9,113.62</td>
<td>750.28</td>
<td>4,393.24</td>
<td>244.92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>1,070.17</td>
<td>305.55</td>
<td>1,260.06</td>
<td>161.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>6,150.00</td>
<td>212.00</td>
<td>1,398.00</td>
<td>54.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>11,318.00</td>
<td>1,544.00</td>
<td>7,686.00</td>
<td>800.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>65,452.16</td>
<td>3,113.33</td>
<td>25,662.36</td>
<td>748.44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>6,192.16</td>
<td>985.90</td>
<td>7,176.38</td>
<td>472.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>49,926.00</td>
<td>1,546.00</td>
<td>12,701.00</td>
<td>138.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>74,985.00</td>
<td>5,462.00</td>
<td>43,430.00</td>
<td>2,476.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>539,331.32</td>
<td>9,479.16</td>
<td>164,054.42</td>
<td>2,505.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>54,035.59</td>
<td>6,425.37</td>
<td>48,131.75</td>
<td>2,008.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>388,932.00</td>
<td>2,046.00</td>
<td>72,213.00</td>
<td>372.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>673,298.00</td>
<td>19,114.00</td>
<td>316,898.00</td>
<td>8,832.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>4,812,974.04</td>
<td>89,076.73</td>
<td>1,087,231.43</td>
<td>9,462.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>386,806.90</td>
<td>23,679.70</td>
<td>353,146.90</td>
<td>5,152.44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MN</td>
<td>3,938,265.00</td>
<td>29,684.00</td>
<td>637,761.00</td>
<td>1,744.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MX</td>
<td>5,694,867.00</td>
<td>121,276.00</td>
<td>1,936,571.00</td>
<td>19,972.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* N = number of nodes generated
Note that the Koopmann-Beckmann problem and the general CijkL case are referred to as "KBP" and "GQAP," respectively. In Table 1, $T_1$ values are compared. It is found that PM has the best average $T_1$ value and smaller standard deviation as the size level grows for both the GQAP and KBP. For the case of the GQAP, an interesting result obtained is that the KT cannot produce $T_1$ on average better than CE. This means that CE is also a good alternative to solve any QAP with $n \leq 10$. Nevertheless, the maximum $T_1$ of each size level obtained from the PM is less than that obtained from the CE. This implies that the PM outperforms the CE for $T_1$ values in every replication of each problem size. For larger $n$ ($n > 10$), it appears that computational times required by all methods become prohibitive to collect sufficient information. Another result obtained shows that the KBP requires less $T_1$ on average for comparable sized problems. This is true because of its restricted formulation compared to the more general case of GQAP.

In Table 2, a comparison of $T_2$ values demonstrates the same conclusion obtained from Table 1, except for $n = 5$, for which KT performs slightly better. From Table 3, the $N$ values of both methods are compared. The results
show that KT requires much less N on average. This state­
ment supports the fact that PM produces a weaker lower
bound and requires more nodes generated in the search tree.
More descriptive statistical results from data obtained in
Tables 1, 2 and 3 are illustrated in Figures 10 to 15 when
the mean values are plotted versus size levels for each
performance measurement. These figures illustrate the
trend of $T_1$, $T_2$ and N of each method as the size level
grows.

From Figure 10 to Figure 13, it appears that PM has a
longer transient state of the exponential time function as
n grows. In Figures 14 and 15, the result is reverse in
case of numbers of nodes generated in the search tree.

To predict the performance of $T_1$, $T_2$ and N for larger
n (> 10), a simple linear regression (SLR) is used to fit
the following equation.

$$y = e^\alpha n + C$$

where

$y$ represented the predicted mean value of $T_1$, $T_2$ and N with
a given n; and

and $C$ are constant values oftained from the analysis of
SLR by letting $y_1 = LN(y) = \alpha n + C$. 
Figure 10. $T_1$ versus $n$ for GQAP.
Figure 12. $T_2$ versus n for GQAP.
Figure 14. N versus n for GQAP.
Figure 15. N versus n for KBP.
By applying the SLR to the data in Tables 1, 2 and 3, relevant results are collected in Table 4. From the results, it is expected that in the case of GQAP with $n = 11$, the PM should have $T_1$ and $T_2$ on average more than 3000 seconds and 1500 seconds, respectively, while the KT will have $T_1$ and $T_2$ on average of more than 22,000 seconds and 12,000 seconds, respectively. In the case of CE, it will take probably more than 13,000 seconds to verify an optimal solution for $n = 11$. Apparently, all methods become prohibitive when $n > 10$. Similarly for the KBP, KT becomes prohibitive when $n > 10$ while PM can be extended to solve up to $n = 12$. Nevertheless, more efficient branch and bound algorithms such as Burkard and Derigs (1980) and Bazaraa and Kirca (1983) showed that they can solve KBP up to size level $n = 15$ but these algorithms are not designed for the case of GQAP.

Another comparison between both methods is shown in Table 5, which compares the results among performance measurements replication by replication and shows the relative performance measured in each size level. To interpret this table, consider the size of 5 for $T_2$: there are 4 pairs for which KT gives better results while 89 pairs are tied. From the results in Table 5, we wish to determine whether PM is preferred to KT statistically. The sign test (see Conover, 1971) is used to test whether two
Table 4
Estimated Exponential Function for Each Performance Measurement*

<table>
<thead>
<tr>
<th></th>
<th>PT</th>
<th>MT</th>
<th>α</th>
<th>C</th>
<th>Y(11)</th>
<th>YLB(11)</th>
<th>YUB(11)</th>
<th>R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>GQAP</td>
<td>PM</td>
<td>1.705</td>
<td>-10.712</td>
<td>3,093.98</td>
<td>255.83</td>
<td>37,417.78</td>
<td>0.967</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KT</td>
<td>1.888</td>
<td>-10.737</td>
<td>22,660.68</td>
<td>13,160.15</td>
<td>39,019.79</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE</td>
<td>1.974</td>
<td>-12.197</td>
<td>13,493.68</td>
<td>3,761.20</td>
<td>48,410.01</td>
<td>0.993</td>
</tr>
<tr>
<td></td>
<td>KBP</td>
<td>PM</td>
<td>1.384</td>
<td>-9.075</td>
<td>471.53</td>
<td>46.93</td>
<td>4,738.24</td>
<td>0.958</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KT</td>
<td>1.653</td>
<td>-9.925</td>
<td>3,850.24</td>
<td>2,145.43</td>
<td>6,909.74</td>
<td>0.998</td>
</tr>
<tr>
<td>T2</td>
<td>GQAP</td>
<td>PM</td>
<td>1.655</td>
<td>-10.875</td>
<td>1,521.03</td>
<td>134.64</td>
<td>17,182.77</td>
<td>0.967</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KT</td>
<td>1.968</td>
<td>-12.230</td>
<td>12,317.33</td>
<td>8,179.07</td>
<td>18,549.38</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td>KBP</td>
<td>PM</td>
<td>1.372</td>
<td>-9.545</td>
<td>256.70</td>
<td>22.88</td>
<td>2,879.50</td>
<td>0.953</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KT</td>
<td>1.820</td>
<td>-12.316</td>
<td>2,212.45</td>
<td>1,302.80</td>
<td>3,757.23</td>
<td>0.998</td>
</tr>
<tr>
<td>N</td>
<td>GQAP</td>
<td>PM</td>
<td>1.978</td>
<td>-4.574</td>
<td>2.90 x 10⁷</td>
<td>1.50 x 10⁷</td>
<td>5.50 x 10⁷</td>
<td>0.998</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KT</td>
<td>1.437</td>
<td>-3.372</td>
<td>251,418.30</td>
<td>81,141.66</td>
<td>779,021.90</td>
<td>0.990</td>
</tr>
<tr>
<td></td>
<td>KBP</td>
<td>PM</td>
<td>1.755</td>
<td>-3.780</td>
<td>5,524,296.00</td>
<td>3,370,045.00</td>
<td>9,047,555</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KT</td>
<td>1.143</td>
<td>-2.406</td>
<td>25,861.00</td>
<td>16,902.40</td>
<td>39,567.70</td>
<td>0.999</td>
</tr>
</tbody>
</table>

*In all cases, the observed significance levels are less than 0.001

MT = method used
P = performance measurement
PT = problem types
R² = coefficient of determination
Y(11) = predicted value at n = 11
YLB(11) = lower bound of Y(11) at 95% confident interval
YUB(11) = upper bound of Y(11) at 95% confident interval
### Table 5
Comparisons Replication by Replication Between PM and KT

<table>
<thead>
<tr>
<th>Size</th>
<th>GQAP</th>
<th></th>
<th></th>
<th>KBP</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PM</td>
<td>KT</td>
<td>PM</td>
<td>KT</td>
<td>PM</td>
<td>KT</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>33</td>
<td>16</td>
<td>22</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>46</td>
<td>1</td>
<td>12</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>50</td>
<td>0</td>
<td>44</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>34</td>
<td>14</td>
<td>21</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>33</td>
<td>12</td>
<td>24</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>48</td>
<td>2</td>
<td>28</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>40</td>
<td>10</td>
<td>33</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>41</td>
<td>9</td>
<td>34</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
populations have the same medians, where the observations come in pairs with one element of each pair being from each population. The sign test was conducted and the results shown in Table 6 and Table 7 for the following hypothesis for GQAP and KBP, respectively.

\( H_0: \ E(X_i) \geq E(Y_i) \quad \text{for all } i \)

\( H_1: \ \text{otherwise,} \)

where \( E(X_i) \) and \( E(Y_i) \) represented the expected value of \( T_1 \), \( T_2 \) and \( N \) for PM and KT, respectively.

In Table 6 and Table 7, the results from the sign test are shown at the level of significance (\( \alpha \)) 0.05. Let the test statistic \( T \) equal the number of points which a performance measurement of PM is better (less) than that of KT. Define \( n^* \) as the total number of pairs excluding all tied pairs, and obtain

\[ t = 0.5(n^* - 1.645 \sqrt{n^*}) \]

(round to positive integer). The decision rule is to reject \( H_0 \) if \( T \geq n^* - t \). An example of applying the sign test to the results in Table 6 is as follows.

Consider the size of 6 for \( T \), value of GQAP in Table 6. There are 43 pairs for which PM gives better results. Therefore,

\[ T = 43. \]
Table 6
Results of GQAP from the Sign Test
(R = Reject $H_0$; A = Accept $H_0$; OSL = Observed Significance Level*)

<table>
<thead>
<tr>
<th>Size</th>
<th>T</th>
<th>n*</th>
<th>t</th>
<th>n*-t</th>
<th>Conclusion</th>
<th>OSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>T1</td>
<td>33</td>
<td>49</td>
<td>19</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>46</td>
<td>47</td>
<td>18</td>
<td>28</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>T1</td>
<td>43</td>
<td>43</td>
<td>16</td>
<td>27</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>34</td>
<td>48</td>
<td>18</td>
<td>16</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>33</td>
<td>45</td>
<td>17</td>
<td>28</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>48</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>9</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>40</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>41</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
</tbody>
</table>

*OSL is defined as the smallest significance level at which one can reject $H_0$ (observing a more extreme test statistic value; $H_0$ is true).
Table 7
Results of KBP from the Sign Test
(R = Reject H0; A = Accept H0;
OSL = Observed Significance Level)

<table>
<thead>
<tr>
<th>Size</th>
<th>T</th>
<th>n*</th>
<th>t</th>
<th>n*-t</th>
<th>Conclusion</th>
<th>OSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>T1</td>
<td>22</td>
<td>35</td>
<td>13</td>
<td>22</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>12</td>
<td>38</td>
<td>15</td>
<td>13</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>T1</td>
<td>44</td>
<td>48</td>
<td>19</td>
<td>29</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>21</td>
<td>42</td>
<td>16</td>
<td>26</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>7</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>24</td>
<td>34</td>
<td>12</td>
<td>22</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>28</td>
<td>46</td>
<td>18</td>
<td>28</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>9</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>33</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>T1</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>34</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>0</td>
<td>50</td>
<td>20</td>
<td>30</td>
<td>A</td>
</tr>
</tbody>
</table>
There are zero pairs for which KT gives better results, so
\[ n^* = 43 \]

(excluding all tied pairs) for \( \alpha = 0.05, t = 0.5(43 - 1.645/43) - 16, n^* - t = 27. \)

In this case, \( T > n^* - t \), so \( H_0 \) is rejected, which means that PM is preferred to KT for this size of \( n = 6 \) of GQAP at \( \alpha = 0.05 \). From Tables 6 and 7, we can conclude that PM produces \( T_1, T_2 \) values on average at least as good as or better than KT while KT produces a better average \( N \) value, which supports the descriptive results from Figure 10 to Figure 15.

Another statistical test referred to as comparison mean test with unknown variances (Montgomery, 1976) was conducted under normality assumption. The conclusion from this test is also the same as obtained from the sign test.

It should be pointed out that the run time values of \( T_1 \) and \( T_2 \) obtained from this experiment are dependent on the efficiency of the computer code. Similar programming techniques, data structures as described in Appendix A and a common compiler (QuickBasic) are used for each algorithm in order to make these relative comparisons as reasonable as possible. Finally, we can summarize the results obtained from this chapter as follows:
For $n < 10$, the proposed method can solve a general QAP in a more reasonable time on average than the method of Kaku and Thompson. In case of $n > 10$, there is still no exact method which can solve any general QAP excluding the Koopmans-Beckmann formulation in a reasonable time.
A branch and bound method has been proposed for solving a general QAP. The method utilizes a new lower bound, which requires much less time to obtain than any others yet proposed in the literature. This lower bound is usually looser, so trade-offs between the strength of lower bound and number of nodes generated have been studied. Discussion of results in Chapter 5 showed that the proposed method performs better (less computational time) than the best available method of Kaku and Thompson for size level \( n \leq 10 \). In the case of \( n \geq 11 \), it appeared that all exact methods become prohibitive because of excessive computational time required. Nevertheless, all results obtained in this work have been run on a COMPAC 386/20 microcomputer which has a slower CPU speed than most mainframe computers. More computational results for larger sized problems by a more powerful computer can be investigated in the future.

Extensions of this research could be conducted in the following ways. First, due to the highly combinatorial nature of the problem, the task of finding an optimal
solution and then verifying its optimality within a reasonable time is almost impossible in the case of large problems. Here, we must resort to suboptimal solutions. Bazaraa and Elshafei (1979) proposed two stepped fathoming methods for obtaining controlled suboptimal solutions in the context of branch and bound. The first method utilizes the relation $\text{LB} \geq a\text{UB}$ where $a \in (0,1]$ to fathom a partial assignment solution with the lower bound LB and the upper bound UB. The second method defines the relation $a \cdot \text{LB} + (1-a)\text{UB} = K$ where $a \in (0,1]$. Since LB < UB, then $K \leq \text{UB}$. Two cases are possible. In the first case, we will be able to find a complete solution with an objective function value less than K. This new objective value becomes the new UB. In the second case, there is no solution with objective value less than K. K becomes the new LB. The process repeats in the search tree until the difference between the lower and upper bound is smaller than a prescribed tolerance. In both cases, if $a$ is close to 1, the search will not speed up considerably. On the other hand, if $a$ is close to zero, then fathoming will be fast but it is likely that a good feasible solution will not be obtained. Bazaraa and Elshafei (1979) recommended $a = 0.7$ as a starting value; then as the search proceeded, $a$ is increased until $a = 1$. 
The proposed method may be modified to solve a more complicated model of the QAP such as dynamic QAP, multi-objectives QAP. Nevertheless, the modified method may be applied to only small problems. Some branch and bound based heuristics should be developed for the case of large problems. Finally, the computer programs developed for the proposed method and the method of Kaku and Thompson are not particularly efficient codes; certainly there are improvements which can be made in this area to decrease execution time. For instance, there are many methods to solve the linear assignment problem. The particular procedure used in both methods may not necessarily be the most efficient. Another factor is the efficiency of the compiler. The QuickBASIC compiler may not be as efficient as a C-compiler or a PASCAL compiler. Using Pascal or C would also allow the use of data structures and dynamic memory allocation to store the search tree.
REFERENCES


APPENDIX

A DESIGN STRATEGY FOR A BRANCH AND
BOUND CODE FOR THE QAP

The purpose of this appendix is to describe a branch and bound scheme suitable for permutation problems such as the QAP. The scheme is based on a backtracking strategy described by Baker (1974). This choice should make things a little bit easier for the computer programmer because the storage demands of a backtracking code are not severe. Therefore, the programmer need not be overly concerned about organizing his data in sophisticated ways in situations where core memory is a scarce resource.

Backtracking is the strategy of always branching from the node corresponding to the most nearly solved subproblem. Stated more explicitly, this means that backtracking attempts to find a complete solution in the most direct way, and then to retrace its steps only so far as necessary to find the next closest solution in the tree. We have utilized this algorithm for the QAP. In this case, the single assignment tree is used. Consider the following efficient means of representing the tree internally.
Define TREE(I,J) to be an $n \times n$ matrix. Row i of this matrix corresponds to level $i$ (object $i$) of the search tree and contains the indices of locations that correspond to newly generated nodes each time branching is executed. To illustrate how this matrix is manipulated, consider the tree in Figure 16.

The corresponding TREE matrix takes the following form.

\[
\begin{array}{cccc}
1 & 3 & 2 & 4 \\
3 & 2 & 4 & 22 \\
2 & 4 & 23 & 24 \\
4 & 19 & 21 & 23 \\
\end{array}
\]

The indices of row 1 in the upper half of this matrix correspond to the nodes at level 1 of the tree ranked by lower bound. Similarly, the indices of row $i$ correspond to nodes at level $i$, ranked by their lower bound. Notice that the lower bounds corresponding to all active nodes are stored in the lower half of the TREE matrix. No other large commitment of storage space is required, and relative little additional information is necessary to carry out the algorithm.

Upon backtracking, the active nodes remaining at levels 2 and 3 are fathomed since the complete solution has
Figure 16. The illustrated tree.
a total cost of 21. When the algorithm next encounters a complete assignment, the tree matrix takes the form

\[
\text{TREE} = \begin{bmatrix}
3 & 2 & 4 & -1 \\
1 & 2 & 4 & \text{LB}_{32} \\
2 & 4 & \text{LB}_{22} & \text{LB}_{23} \\
4 & 21 & 23 & -1
\end{bmatrix}
\]

where

- \( \text{LB}_{ij} \) is the lower bound corresponding to the node at element \( \text{TREE}_{ij} \) of the TREE matrix and

- -1 represents the part of solutions having been enumerated.

If all \( \text{TREE}_{ij} = 1 \), the process stops and an optimal solution is verified.

In the case of the binary search tree, the same approach can be adapted by defining TREE as an \( n \times 2 \) matrix. Similarly, this approach can be applied to a variety of combinatorial optimization problems such as the traveling salesman problem (TSP), flow shop scheduling, single machine scheduling with minimum total tardiness, etc.