ARAB REPUBLIC OF EGYPT

THE INSTITUTE OF
NATIONAL PLANNING

التخطيط القومى

Memo. No. 1256

An Exact Branch-And-Bound Procedure For The Quadratic Assignment Problem

by

Dr. Mokhtar S. Bazaraa

Dr. Alwalid N. Elshafei

September 1979

CAIRO
SALAH SALEM St.-NASR CITY

REPRINTED FROM

NAVAL RESEARCH LOGISTICS QUARTERLY

MARCH 1979
VOL. 26, NO. 1

OFFICE OF NAVAL RESEARCH

NAVSO P-1278

AN EXACT BRANCH-AND-BOUND PROCEDURE FOR THE QUADRATIC-ASSIGNMENT PROBLEM

M. S. Bazaraa*

School of Industrial and Systems Engineering
Georgia Institute of Techology
Atlanta, Georgia

A. N. Elshafei†

Institute of National Planning
Cairo, Egypt

ABSTRACT

The quadratic-assignment problem is a difficult combinatorial problem which still remains unsolved. In this study, an exact branch-and-bound procedure, which is able to produce optimal solutions for problems with twelve facilities or less, is developed. The method incorporates the concept of stepped fathoming to reduce the effort expended in searching the decision trees. Computational experience with the procedure is presented.

1. INTRODUCTION

The quadratic-assignment problem is a combinatorial problem that has been of great interest to many researchers. The problem can be stated as follows:

$$\text{Minimize } \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m} c_{ijkl}\, x_{ij}\, x_{kl} + \sum_{i=1}^{m}\sum_{j=1}^{m} f_{ij}\, x_{ij}$$

$$\text{subject to } \sum_{j=1}^{m} x_{ij} = 1, \qquad i = 1, \ldots, m,$$

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1, \ldots, m,$$

$$x_{ij} \text{ is 0 or 1}, \qquad i,j = 1, \ldots, m.$$

The problem can be interpreted as follows. We suppose that m facilities or objects are to be assigned to m locations. Here, x_{ij} is 1 if facility i is placed in location j and is 0 otherwise. The quantity c_{ijkl} is the cost of the mutual assigment of object i to location j and object k to location l, and it is usually determined as the number of interactions u_{ik} between objects i and k weighted by the distance from location j to location l, that is, $c_{ijkl} = u_{ik}d_{jl}$. Furthermore, f_{ij} is the fixed cost of assigning facility i to location j.

*This author's work was supported by NSF grant #GK-38337.
†Work done while visting North Carolina State University.

Since Koopmans and Beckman [10] introduced the quadratic-assignment problem in the context of locating indivisible objects, the problem has gained a great deal of popularity among researchers, due mostly to its wide range of applications. In [16], Whitehead and Elders discussed the use of the problem in the area of building layout. In [4], Elshafei described a quadratic-assignment algorithm that can be used in the context of hospital layout. The problem has also been used in the fields of urban planning, control-panel layout, and wiring design in the placement of electronic components in an assembly. For details on these applications, the reader may refer to Hopkins [9], Dorris [3], Breuer [2], Gaschutz and Ahrens [5], and Steinberg [15].

Various procedures for the solution of the problem have been suggested in the literature, including both exact and heuristic procedures. This study concerns itself with exact methods. Currently, the available exact procedures are all of the branch-and-bound type and can be classified into single-assignment algorithms, pair-assignment algorithms, and pair-exclusion algorithms. Single-assignment algorithms proceed by the assignment of one unassigned facility to a vacant location at any stage of the search process. The procedures of Gilmore [7], Graves and Whinston [8], and Lawler [12] fall in this class. Neither Gilmore nor Lawler reported any computational experience. Graves and Whinston compared their procedure with some existing heuristics and provided better-quality solutions, but they did not guarantee optimality. Pair-assignment methods proceed by simultaneous location of two facilities at two unoccupied locations. In [11], Land described a pair-assignment algorithm that first reduces the cost matrix so that it contains a zero in each row and a zero in each column. Gavett and Plyter [6] extended the method of Land by tightening the computation of the lower bounds. They reported that their algorithm took 14 min on an IBM 7044 machine to solve a problem of size $m = 7$ and 42 min for $m = 8$. Pair-exclusion algorithms proceed on the basis of a stage-by-stage exclusion of assignments from a solution to the problem. In [14], Pierce and Crowston reported the results for this procedure for a problem of size four facilities.

In this study, we discuss an exact branch-and-bound scheme for solving the quadratic-assignment problem. The method is similar to that suggested by Gilmore [7], but it differs in the computation of the lower bounds and in the branching rules. It also incorporates the concept of stepped fathoming given in [1] for speeding the search of the decision tree. The reported algorithm was able to find the optimal solution of a problem of size 12 facilities but failed to produce exact solutions for problems of size $m \geqslant 15$.

2. AN EXACT-SOLUTION PROCEDURE

In this section we describe a branch-and-bound solution procedure which can be used to obtain optimal and qualified suboptimal solutions. The following notation will be used. The location to which object i is assigned is denoted by i^*. Hence, the mutual cost of assigning objects i and j is $u_{ij}d_{i^*j^*} + u_{ji}d_{j^*i^*}$ and the fixed cost of these assignments is $f_{ii^*} + f_{jj^*}$.

Decision Tree and General Framework

At each stage of the algorithm, we have a set of objects that has already been assigned to certain locations. These already assigned objects form a *partial solution* of the assignment problem. In order to obtain a feasible solution, that is, a complete assignment, we must find a *completion* of the partial solution. Rather than considering all the possible ways of completing the partial solution, we first investigate whether the partial solution might lead to a complete solution with an objective value smaller than the best solution that we already have. This is done by calculating a *lower bound* on the cost of completing this partial solution.

Let B be the lower bound and let C^* be the cost of the best available assignment. If $B \geqslant C^*$ then any completion of the partial solution can lead to no improvement. In this case, the partial solution is said to be *fathomed*, and it is abandoned. On the other hand, if $B < C^*$ it is worthwhile for us to pursue the partial solution by seeking to assign more objects.

Calculation of the Lower Bound

Suppose that a set of objects indexed by the set I has already been assigned to a set of locations indexed by the set J. In particular, suppose that object i is assigned to location i^*. A lower bound B on the cost of this partial solution and its completion is computed as $B = C_1 + C_2 + C_3$, where

C_1 = cost of the partial assignment;

C_2 = lower bound on the cost of interaction between assigned objects and unassigned objects plus the fixed cost of locating the unassigned objects;

C_3 = lower bound on the cost of interaction among the unassigned objects themselves.

Note that C_1 is given by

$$C_1 = \sum_{i \in I} f_{ii^*} + \sum_{i,j \in I} u_{ij}\, d_{i^*j^*}.$$

Here C_2 is the optimal cost of the following linear-assignment problem:

$$\begin{aligned}
&\text{Minimize } \sum_{i \notin I} \sum_{j \notin J} b_{ij} x_{ij} \\
&\text{subject to } \sum_{j \notin J} x_{ij} = 1 \qquad \text{for } i \notin I, \\
&\phantom{\text{subject to }} \sum_{i \notin I} x_{ij} = 1 \qquad \text{for } j \notin J, \\
&\phantom{\text{subject to }} x_{ij} \geqslant 0 \qquad \text{for } i \notin I,\ j \notin J,
\end{aligned}$$

where b_{ij} is a bound on the cost resulting from the assignment of object i to location j. For example, we can use

$$b_{ij} = f_{ij} + \sum_{t \in I} (u_{it}\, d_{jt^*} + u_{ti}\, d_{t^*j}).$$

Of course, a complete solution of the linear-assignment problem can be replaced by the simpler task of reduction of the cost matrix (b_{ij}) such that it has at least one zero in each row and each column by subtraction of the minima of the rows from the rows, and the minima of the columns from the resultant columns.

Two methods of computation of C_3 are available. The first method relies on the ranking of the interactions and distances as follows. Rank the interactions u_{ij} in a descending order for $i, j \notin I$, and rank the distances d_{ij} in an ascending order for $i, j \notin J$. This results in an ordered interaction vector and an ordered distance vector. Then C_3 is the inner product of these two vectors. In other words, we calculated C_3 by matching the largest interaction among unassigned elements to the smallest distance between unassigned locations, the second largest interaction to the second smallest distance, and so forth. Clearly, this procedure will give a lower bound on the cost among unassigned elements.

An alternative method for finding a suitable bound C_3 is the solution of a linear assignment problem whose cost matrix is constructed as follows. For each unlocated element i, rank the interactions between it and all other unlocated elements in descending order. Similarly, for each vacant location j, rank the distances between it and all other vacant locations in ascending order. Then a lower bound e_{ij} on the cost of locating facility i in location j is the inner product of the above two vectors. Thus, we find C_3 by solving the following linear-assignment problem:

$$\begin{aligned} &\text{Minimize } \sum_{i \notin I} \sum_{j \notin J} e_{ij} x_{ij} \\ &\text{subject to } \sum_{j \notin J} x_{ij} = 1 && \text{for } i \notin I, \\ &\qquad \sum_{i \notin I} x_{ij} = 1 && \text{for } j \notin J, \\ &\qquad x_{ij} \geqslant 0 && \text{for } i \notin I,\ j \notin J. \end{aligned}$$

Needless to say, the above assignment problem can be combined with the assignment problem in the C_2 calculation to give $C_2 + C_3$. The overall lower bound $B = C_1 + C_2 + C_3$ is now available.

Continuation of the Search: Fathoming (Backward Move)

Suppose that k objects indexed by the set I have already been assigned to k locations indexed by the set J. The *level* of the search tree is called k. A bound on the cost that results from all completions of the current partial solution B is calculated as discussed above. If $B \geqslant C^*$, where C^* is the best known cost of a complete assignment, then the partial solution is *fathomed*. The last assignment, that is, the kth assignment, is *banned* or *prohibited* in the hope that this will lead to an improved completion. For example, if the kth assignment involves placing object i_k in location i_k^* then $x_{i_k i_k^*}$ is forced to be zero. Here, i_k is placed in the list of unassigned objects, that is, i_k is removed from I, and similarly i_k^* is removed from the list of unassigned locations, that is, i_k^* is removed from J. We calculate a new bound B' in exactly the same manner as explained above, except, of course, that the assignment $x_{i_k i_k^*} = 1$ is prohibited, by forcing $b_{i_k i_k^*} = \infty$ while we solve the linear-assignment problem. If B' is still $\geqslant C^*$, then the partial solution of the first $k - 1$ assignments, while banning the assignment i_k to i_k^*, can still lead to no improved solutions. Since the first $k - 1$ assignments with $x_{i_k i_k^*} = 1$ and $x_{i_k i_k^*} = 0$ lead to no improvement, then all the possibilities at level k have been exhausted, and prohibition of the assignment at level $k - 1$ is now possible. This condition is called *strong fathoming*. The level of the tree is thus reduced by one unit, and the assignment at level $k - 1$ is prohibited. If, on the other hand, the bound B' is less than C^*, a condition referred to as *weak fathoming*, then object i_k is assigned to some other unassigned location. This is discussed in more detail in the forward move of the search. The cases of strong and weak fathoming are depicted in Figures 1 and 2.

Progress of the Search (Forward Move)

If $B < C^*$ then we must choose an object i_{k+1} for assignment. For example, we may choose i_{k+1} to be an unassigned object with maximum interactions with already assigned objects, or choose i_{k+1} to be an unassigned object with maximum interactions with the most recently assigned object i_k. This object is assigned to an unassigned location i_{k+1}^* which is not prohibited. This location i_{k+1}^* can be chosen in such a way that the total weighted interaction with assigned objects in minimal. For example, choose i_{k+1}^* which minimizes $\sum_{j=1}^{k} u_{i_{k+1} i_j} d_{t i_j^*} + \sum_{j=1}^{k} u_{i_{k+1} i_j} d_{i_j^* t}$ over $t \notin J$ such that $x_{i_{k+1} t} = 1$ is not prohibited.

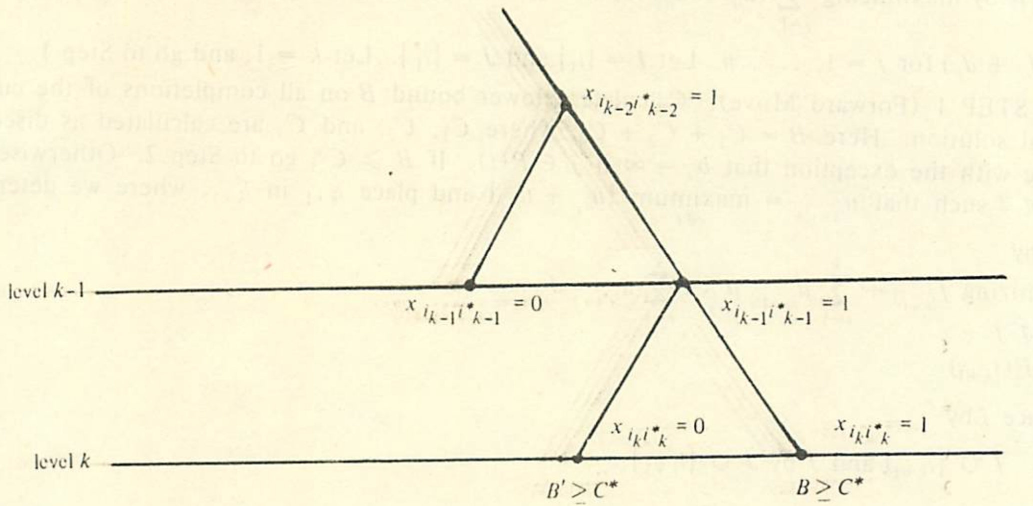

FIGURE 1. Illustration of strong fathoming.

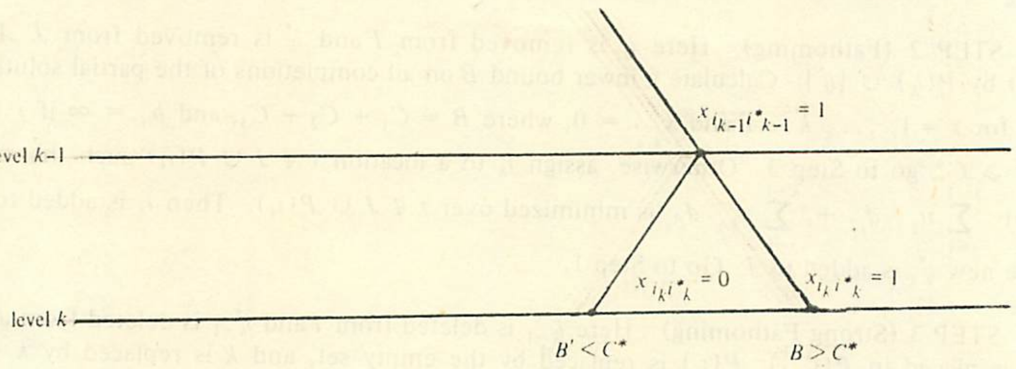

FIGURE 2. Illustration of weak fathoming.

Needless to say, when the level of the tree is m, if the cost is less than C^*, then C^* is updated and the corresponding assignment is stored.

Termination

We have described forward and backward progress of the search tree. If the level of the tree ever reaches value zero, then we stop. This would mean that we are currently at level one, and are trying to backtrack. This means that all possible assignments under $x_{i_1 i^*_1} = 1$ and $x_{i_1 i^*_1} = 0$ have already been enumerated so that all possible ways of assignment of the m objects are enumerated, and we stop. The stored assignment and corresponding C^* give the optimal solution.

Summary of the Algorithm

We have discussed above all the details required to describe the following solution procedure of the quadratic assignment problem:

INITIALIZATION STEP: Let the prohibited locations for object i be $P(i) = \emptyset$ for $i = 1, \ldots, m$, and let $C^* = \infty$. Choose an object i_1 and place it in location i_1^*. We may determine i_1 by maximizing $\sum_{j=1}^{m} (u_{ij} + u_{ji})$ for $i = 1, \ldots, m$, and we can determine i_1^* by minimizing $\sum_{i=1}^{n} (d_{ij} + d_{ji})$ for $j = 1, \ldots, n$. Let $I = \{i_1\}$ and $J = \{i_1^*\}$. Let $k = 1$, and go to Step 1.

STEP 1 (Forward Move): Calculate a lower bound B on all completions of the current partial solution. Here $B = C_1 + C_2 + C_3$, where C_1, C_2, and C_3 are calculated as discussed above with the exception that $b_{ij} = \infty$ if $j \in P(i)$. If $B \geqslant C^*$, go to Step 2. Otherwise pick $i_{k+1} \notin I$ such that $u_{i_{k+1}i_k} = \underset{i \notin I}{\text{maximum}}\ [u_{ii_k} + u_{i_k i}]$ and place i_{k+1} in i_{k+1}^*, where we determine i_{k+1}^* by

$$\underset{\substack{j \notin J \\ j \notin P(i_{k+1})}}{\text{minimizing}}\ f_{i_{k+1}j} + \sum_{t=1}^{k} u_{i_{k+1}i_t} d_{ji_t^*} + \sum_{t=1}^{k} u_{i_t i_{k+1}} d_{i_t^* j}.$$

Replace I by

$I \cup \{i_{k+1}\}$ and J by $J \cup \{i_{k+1}^*\}$.

If $k = m - 1$, then C_1 is the cost of the complete assignment. If $C_1 < C^*$, replace C^* by C_1, store (i_t, i_t^*) for $t = 1, \ldots, m$, replace k by m, and go to Step 2. If $C_1 \geqslant C^*$, then replace k by m and go to Step 2. If $k < m - 1$, then replace k by $k + 1$ and repeat Step 1.

STEP 2 (Fathoming): Here i_k is removed from I and i_k^* is removed from J. Replace $P(i_k)$ by $P(i_k) \cup \{i_k^*\}$. Calculate a lower bound B on all completions of the partial solution $x_{i_t i_t^*} = 1$ for $t = 1, \ldots, k - 1$ and $x_{i_k i_k^*} = 0$, where $B = C_1 + C_2 + C_3$, and $b_{ij} = \infty$ if $j \in P(i)$. If $B \geqslant C^*$, go to Step 3. Otherwise, assign i_k to a location $t \notin J \cup P(i_k)$ such that the cost $f_{i_k t} + \sum_{j=1}^{k-1} u_{i_k i_j} d_{ti_j^*} + \sum_{j=1}^{k-1} u_{i_j i_k} d_{i_j^* t}$ is minimized over $t \notin J \cup P(i_k)$. Then i_k is added to I, and t, the new i_k^*, is added to J. Go to Step 1.

STEP 3 (Strong Fathoming): Here i_{k-1} is deleted from I and i_{k-1}^* is deleted from J. Then i_{k-1}^* is placed in $P(i_{k-1})$, $P(i_k)$ is replaced by the empty set, and k is replaced by $k - 1$. If $k = 0$ go to Step 4, otherwise go to Step 2.

STEP 4 (Termination): The search of the decision tree has been completed. The optimal cost is C^*, and its corresponding assignment (i_k, i_k^*), $k = 1, \ldots, m$, is the optimal assignment. Stop.

Note that during the initialization step an upper bound $C^* = \infty$ is used. As the search progresses, C^* denotes the objective value of the best available complete assignment. Further, $P(i)$ represents the locations that object i cannot be assigned to. These are initialized by the empty sets. Step 1 represents a forward step, where the level of the tree increases by one unit. In this case the bound is less than C^*, hence a complete solution with an objective better than C^* is possible. Step 2 is a fathoming step, where $B \geqslant C^*$. In this case the last assignment is prohibited. Immediately, a new bound is calculated. If the new bound is less than C^*, then a forward move is made. But if the bound is still greater than or equal to C^*, then a strong fathoming is made at Step 3, and the level of the tree is reduced. Of course, strong fathoming is most desirable, since it avoids the expensive task of trying to locate object i_k in a free location other than i^*_k.

Adding New Facilities to an Existing Layout

In many applications a large number of facilities are already preassigned, and only some new facilities are to be placed in such a way that the overall cost is minimized. In this case, the above algorithm can be applied with a few obvious modifications in the calculations. Since the preassigned objects and their locations will remain unchanged, these objects will always be in the set I and their locations will always be in the set J. In the search tree, if γ objects are already assigned, we start the search by assigning more objects, that is, the level of the tree starts at $\gamma + 1$. If the level of the tree ever becomes γ, then we stop.

3. SUBOPTIMAL AND OPTIMAL SOLUTIONS BY STEPPED FATHOMING

Due to the highly combinatorial nature of the problem, the task of finding an optimal solution and then verifying its optimality within a reasonable computational time is almost impossible in the case of large problems. Here we must resort to suboptimal solutions. The branch-and-bound procedure itself can be used to obtain qualified suboptimal solutions. In [1], Bazaraa and Elshafei proposed two stepped fathoming methods for obtaining controlled suboptimal solutions in the context of branch and bound. The application of these methods for the quadratic-assignment problem is discussed in this section.

Method 1

Recall that a partial solution is fathomed if the lower bound on all its completions is at least as big as $C^* > 0$, the best known objective value. Suppose that a partial solution is fathomed if $B \geqslant \alpha C^*$, where $\alpha \in (0,1]$. In this case, the partial solution is abandoned if there is no hope that it will lead to an objective which is better than αC^*. The purpose of this simple strategy is clear. We want to fathom the partial solution quickly even if it might lead to a slight improvement. Of course, as a new C^* is found, then we fathom whenever the bound is greater than or equal to α times the new C^*. The procedure continues until we cannot find a feasible solution with an objective less than αC^*. Thus, we have a feasible assignment with objective C^* coupled with the statement that the optimal objective is greater than or equal to αC^*.

Choice of α: Of course, if α is small, then fathoming will speed up considerably, resulting in a small computational effort. But on the other hand, the quality of the best feasible solution is not satisfactory. We recommend values of $\alpha \geqslant 0.9$, depending on the accuracy required.

Method 2

At each stage of the algorithm, we have an upper bound C^*. A lower bound on the overall problem L can be devised. Rather than fathoming on C^*, suppose we fathom on $K = \alpha C^* + (1 - \alpha) L$, where $\alpha \in (0,1]$. Since $L < C^*$, then $K \leqslant C^*$. Two cases are possible. In the first case, we will be able to find a complete solution with objective less than K. The objective value of this new solution becomes the new upper bound C^* and the process is repeated. In the second case, we will not be able to find such a solution. This automatically implies that there are no solutions with objective less than K and hence K itself is the new lower bound. The process is repeated. From this we keep narrowing the gap between the lower and upper bounds, either by lowering the upper bound when an improved feasible assignment is found or by raising the lower bound when no feasible solution with objective less than K is found. When the difference between the lower and upper bounds is smaller than a prescribed tolerance we stop.

Choice of α: Here α is any number in the interval $(0,1]$. Of course, if α is close to 1, then we are in effect fathoming on a number very close to C^*, and the search will not speed up considerably. On the other hand, if α is close to zero, then improvement is achieved only if we obtain a feasible solution very close to the overall lower bound. In this case, fathoming will be fast, but it is likely not to obtain feasible solutions with an objective value that is less than K. If $\alpha = 0.5$ then the interval of uncertainty in which the optimal objective value lies will be halved at each stage.

Calculation of the Initial Overall Lower and Upper Bounds

Initial lower and upper bounds are needed to implement the above fathoming scheme. To calculate the lower bound, first calculate a lower bound b_{ij} on the cost of locating object i to location j, as discussed in Section 2. Then a linear-assignment problem is solved to find L. More precisely, let L be the optimal objective value of the following problem:

$$\text{Minimize} \sum_{i=1}^{m} \sum_{j=1}^{m} b_{ij} x_{ij}$$

$$\text{subject to} \sum_{j=1}^{m} x_{ij} = 1 \qquad \text{for } i = 1, \ldots, m,$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for } j = 1, \ldots, m,$$

$$x_{ij} \geqslant 0 \qquad \text{for } i, j = 1, \ldots, m.$$

We can obtain an upper bound C^* immediately by calculating the quadratic cost of the optimal assignment resulting from the above problem. Now Method 2 can be initiated.

Exact Solution by Stepped Fathoming

Either of the above two methods could be slightly modified to provide optimal solutions and still reduce the portion of the decision tree explicitly enumerated. Suppose that with any given α the search terminates with the conclusion that there exsits no feasible solution with a quadratic objective value less than K, where $K = \alpha C^*$ for the first method and $K = \alpha C^* + (1 - \alpha) L$ for the second method. The search can then be repeated from the complete stored solution whose objective value is C^* with a larger value of α.

Several increasing values of α, with the last value equal to one, could be used. Obviously, for $\alpha = 1$ we would fathom if the objective value is at least equal to C^*, and the method will produce an optimal solution. Even though portions of the decision tree may be repeated, the quick fathoming would usually result in a reduction of the overall computational effort. For further details, the reader is referred to [1].

4. COMPUTATIONAL EXPERIENCE

The experience gained with the solution procedure was in relation to the problems reported by Nugent, Vollman, and Ruml [13]. First, we will discuss some details pertinent to the purely computational aspects of the procedure.

Choice of α

If a small value of α is chosen, the fictitious upper bounds αC^* and $\alpha C^* + (1 - \alpha) L$ tend to be tighter, and hence the search becomes faster. However, we may increase the number of times we restart the search from the current best complete solution with a smaller value of α. On the other hand, if α is large then fathoming becomes weaker, and the search tends to be lengthier but to have fewer restarts. The tradeoff is only computational and is data dependent.

During the course of our study, we noticed that due to the features of the search procedure many successive good solutions are obtained very early in the search, and the optimum follows suit. As a result, we have adopted the strategy of choosing a small value of α at the beginning of the search and at a certain stage switching to $\alpha = 1$. We accomplish this by specifying an initial value of α and also specifying a difference between the actual upper and lower bounds at the achievement of which α is switched to 1. If the difference is appropriately chosen, we will get to the stage where the upper bound is tight enough to speed up the search, and also we will not have to restart the search once the tree is enumerated, as there is no interval of uncertainty in this case. We found that the choice of $\alpha = 0.7$ as a starting value was adequate for all the problems solved. The difference between the two bounds at which we switched to $\alpha = 1$ varied from one problem to another.

Solving the Linear Assignment Problem

As was mentioned in Section 2, the lower bound can be calculated by various methods. One procedure involves the use of a linear-assignment problem to obtain a tighter lower bound. There is no need, however, to solve a fresh assignment problem each time a lower bound is to be calculated. It is possible to take any previous solution and update it according to the new cost matrix.

Tables 1 and 2 summarize the experience with the following two codes:

QAP3: A code for an algorithm based on calculation of the lower bound $C_1 + C_2 + C_3$ by the matching of the ordered interaction and distance vectors as discussed in Section 2.

QAP7: A code for an algorithm based on calculation of the lower bound $C_1 + C_2 + C_3$ by the solution of a linear assignment problem. Here the cost matrix is first reduced so that it has a zero in every row and every column. If C^* is greater than or equal to the bound, we fathom. Otherwise, the complete linear-assignment problem is solved in the hope that the lower bound can be tightened.

Total Number of Nodes: the number of nodes, both intermediate and terminal, generated during the search.

Total Number of Moves: the number of forward and backward moves conducted during the search.

Number of times it was necessary to solve an assignment problem: whenever the lower bound calculated at any particular node by the reduction method was less than the current upper bound, it was necessary to solve a linear assignment problem to improve the value of this lower bound. Naturally, this strategy is applicable only to QAP7.

Fathoming Efficiency: the ratio between the number of times the search was not pursued, as a result of the lower-bound test, to the total number of times the lower-bound test was applied.

Comparison of QAP3 and QAP7: We recall that the only difference between QAP3 and QAP7 is the method of calculation of the lower bound as shown in Section 2. In Table 1, we notice that the solution times when QAP7 was used were always less than those obtained when QAP3 was used.

TABLE 1. *Summary of the Computational Experience with QAP3 and QAP7*

Problem Number	Size	Algorithm	Total Number of Nodes	Total Number of Moves	No. of Times the LB was Calculated	No. of Times it was Necessary to Solve an AP	Fathoming Efficiency %	Value of Best Solution Obtained	Solutions Time (seconds)*
4001	5×5	QAP3	15	38	43	—	46.12	25√	0.26
		QAP7	9	20	20	14	45.00	25√	0.15
4002	6×6	QAP3	27	99	129	—	54.26	43√	1.01
		QAP7	18	56	67	36	52.24	43√	0.63
4003	7×7	QAP3	55	177	235	—	51.06	74√	3.9
		QAP7	22	62	73	40	58.90	74√	2.7
4004	8×8	QAP3	231	739	1005	—	50.25	107√	15.3
		QAP7	52	179	235	141	52.77	107√	9.6
4005	12×12	QAP3	6877	29282†	—	—	—	289	221.8
		QAP7	5724	24496	37531	26368	50.02	289√	490.4
4006	15×15	QAP3	10071	40502‡	—	—	—	618	—

*On an IBM 370/165
†No restarting was allowed
‡The search was forced to termination
√Optimality verified

TABLE 2. *Number of Nodes Generated at the Tree Levels*

Problem Number	Algorithm	Number of Moves	Total Number of Nodes	Number of Nodes at Different Levels											
				1	2	3	4	5	6	7	8	9	10	11	12
4001	QAP3	38	15	0	4	5	3	3							
	QAP7	20	9	0	2	3	2	2							
4002	QAP3	99	27	0	6	13	4	2	2						
	QAP7	56	18	0	6	2	4	3	3						
4003	QAP3	177	55	0	7	21	21	3	1	2					
	QAP7	62	22	0	6	8	2	2	2	2					
4004	QAP3	739	231	0	8	40	109	58	10	3	3				
	QAP7	179	52	0	8	17	10	8	3	3	3				
4005	QAP3	29282*	6877	0	12	125	866	3367	1774	645	73	6	3	3	3
	QAP7	24496	5724	0	12	132	884	2539	1492	585	53	18	3	3	3

*Decision tree has not been exhausted.

Our observation about QAP3 is that it can face severe difficulties when the problem size increases. For example, problem 4005 was rerun with a starting upper bound equal to the true optimal objective value of $\alpha = 1$ in the hope that this would speed up the search. However, we had to terminate the problem after 50,400 moves, as we noticed that 11,539 nodes were generated but the number of nodes at various levels were:

0 4 41 358 2411 4101 3715 1000 184 8 3 3.

Thus a substantial part of the tree was still to be searched, and the estimated time for the completion of the search was about 15 minutes on the IBM 370/165. Also, the experience with 4006 was not more encouraging. Note that both QAP3 and QAP7 found the optimal solutions of problems 4001 through 4004 and verified optimality. QAP3 and QAP7 found the optimal solution of problem 4005, but only QAP7 verified optimality.

We might also add that the concept of stepped fathoming was essential in the procedure. QAP3 and QAP7 were not able to find optimal solutions to some of the reported problems when $\alpha = 1$ was used from the start of the search.

ACKNOWLEDGMENTS

The authors would like to thank Dr. L. F. McGinnis of the Georgia Institute of Technology for providing them with a code for the solution of the linear-assignment problem.

REFERENCES

[1] Bazaraa, M. S., and A. N. Elshafei, "On the Use of Fictitious Bounds in Tree Search Algorithms," Management Science *23*, 904-908 (1977).

[2] Breuer, M. A., "The Formulation of Some Allocation and Connection Problems as Integer Programs," Naval Research Logistics Quarterly *13*, 83-95 (1966).

[3] Dorris, A. L., "The Utility of Optimization Techniques in the Design of Man-Machine Systems," Masters Thesis, Georgia Institute of Technology, Atlanta, Georgia (1971).

[4] Elshafei, A. N., "Hospital Layout as a Quadratic Assignment Problem," Operational Research Quarterly *28*, 167-179 (1977).

[5] Gaschutz, G. K., and J. H. Ahrens, "Suboptimal Algorithm for the Quadratic Assignment Problem," Naval Research Logistics Quarterly *15*, 49-62 (1968).

[6] Gavett, J. W., and N. V. Plyter, "The Optimal Assignment of Facilities to Locations by Branch and Bound," Operations Research *14*, 210-232 (1966).

[7] Gilmore, P. C., "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem," SIAM Journal on Applied Mathematics *10*, 305-313 (1962).

[8] Graves, G. W., and A. B. Whinston, "An Algorithm for the Quadratic Assignment Problem," Management Science *17*, 453-471 (1970).

[9] Hopkins, L. D., "Land-Use Plan Design—Quadratic Assignment and Central Facility Models," Environment and Planning *9*, 625-642 (1977).

[10] Koopmans, T. C., and M. Beckman, "Assignment Problems and the Location of Economic Activities," Econometrica *25*, 53-76 (1957).

[11] Land, A. H., "A Problem of Assignment with Interrelated Costs," Operational Research Quarterly *14*, 185-198 (1963).

[12] Lawler, E. L., "The Quadratic Assignment Problem," Management Science *9*, 586-599 (1963).

[13] Nugent, C. E., T. E. Vollman, and J. Ruml, "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," Operations Research *16*, 150-173 (1968).

[14] Pierce, J. F., and W. B. Crowston, "Tree Search Algorithms for Quadratic Assignment Problems," Naval Research Logistics Quarterly *18*, 1-36 (1971).

[15] Steinberg, L., "The Backboard Wiring Problem: A Placement Algorithm," SIAM Journal on Applied Mathematics *3*, 37-50 (1961).

[16] Whitehead, B., and M. Z. Elders, "An Approach to the Optimum Layout of Single Story Buildings," Architect's Journal *139*, 1373-1380 (1964).