AN EXACT ALGORITHM FOR THE GENERAL QUADRATIC ASSIGNMENT PROBLEM

by

B.K. Kaku & G.L. Thompson

December, 1983

DRC-70-N5-83

AN EXACT ALGORITHM FOR THE GENERAL QUADRATIC ASSIGNMENT PROBLEM

by

Bharat K. Kaku

and

Gerald L. Thompson

March 1983

This report was prepared as part of the activities of the Management Science Research Group, Carnegie-Mellon University, under Contract No, N00014-82-K-0392 NR 047-048 with the U. S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Science Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pa. 15213

AN EXACT ALGORITHM FOR THE GENERAL QUADRATIC ASSIGNMENT PROBLEM

ABSTRACT

We develop an algorithm that is based on the linearization and decomposition of a general Quadratic Assignment Problem of size n into n² Linear Assignment Problems of size (n-1). The solutions to 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[20], but permit us to calculate tighter bounds. Computational experience is given for solution to optimality of problems of size up to n = 10.

1. INTRODUCTION

A wide and diverse range of problems can be formulated as quadratic assignment problems: 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 location of indicators and controls on a control panel or in a control room, the backboard wiring problem in the design of computers and other electronic equipment, and the production sequencing problem with dependent setup times. These problems are similar in structure to the classical linear assignment problem (for which very efficient algorithms exist) of assigning indivisible facilities to discrete locations, but are more complicated because the objective function contains terms that are quadratic in the decision variables, which arise due to the interdependence of facilities.

1.1. MATHEMATICAL FORMULATION

Given n^4 cost coefficients c_{ijpq}, (i, j, p, q = 1, 2, ..., n), the quadratic assignment problem (QAP) is:

(1) $$\text{Minimize } Z = \sum_{i,j} \sum_{p,q} c_{ijpq} x_{ij} x_{pq}$$

subject to

(2) $$\sum_j x_{ij} = 1 \qquad \forall i$$

(3) $$\sum_i x_{ij} = 1 \qquad \forall j$$

(4) $$x_{ij} \in \{0,1\} \qquad \forall i, j$$

Koopmans and Beckmann[i8] were the first to formulate a quadratic assignment problem. They looked at the problem of locating plants with interplant flows and defined the following matrices:

A = | a_{ij} | is the profitability matrix, where a_{ij} is the profit expected from the operation of plant i at location j and is independent of other plant locations. This profit is gross revenue less costs of primary inputs, but before subtracting costs of inter-plant transportation.

• = | f_{ij} | is the flow matrix, where f_{ij} is the required commodity flow (in weight units) from plant i to plant j.

Δ = | d_{ij} | is the transportation cost matrix, where d_{ij} is the cost of transporting a unit flow from location i to location j.

The flow coefficients f_{ij} are assumed independent of the locations assigned, and the transportation costs d_{ij} are assumed independent of the plant assignments, and applicable to all amounts and compositions of flows. Finally, the transportation cost coefficients satisfy triangular inequalities. Given these definitions and assumptions, total net revenue for the agglomeration of plants is

(5) $$\sum_{i,j} a_{ij} x_{ij} - \sum_{i,j} \sum_{p,q} f_{ip} d_{jq} x_{ij} x_{pq}$$

The quadratic assignment problem is to maximize this expression by choice of a suitable permutation matrix 5 = | x_{ij} | subject to (2), (3) and

(6) $$x_{jj} = \begin{cases} 1 & \text{if plant i is assigned to location j} \\ 0 & \text{otherwise} \end{cases}$$

Alternatively, as suggested by Pierce and Crowston[26], the a_{ij}'s can be considered as the cost of establishing and operating plant i at location j, plus the cost of supplying some prespecified customer demand from location j. Again, these costs are independent of other plant locations. The objective function changes to

(7) Minimize $$\sum_{ij} a_{ij} x_{ij} + \sum_{ij} \sum_{p,q} f_{ip} d_{jq} x_{ij} x_{pq}$$

For the facilities layout problem we redefine the coefficients as follows:

$A = \| a_{ij} \|$ is the fixed cost matrix, where a_{ij} is the fixed linear cost of installing facility i at location j.

$\Phi = \| f_{ij} \|$ is the intensity matrix, where f_{ij} is the cost per unit distance of transporting the flow from facility i to facility j.

$\Delta = \| d_{ij} \|$ is the distance matrix, where d_{ij} is the distance from location i to location j.

This definition of the coefficients permits us to take into account different material handling methods used for transporting different materials, parts and sub/assemblies. The facilities layout problem can now be written as a QAP with objective function (7), subject to constraints (2), (3) and

$$x_{ij} = \begin{cases} 1 & \text{if facility i is assigned to location j} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

If the costs of installing a facility are either independent of locations or identical, the objective function reduces to

$$\text{Minimize} \sum_{i,j} \sum_{p,q} f_{ip} d_{jq} x_{ij} x_{pq} \tag{9}$$

Equations (7) and (9) constitute a special case of the QAP, known as the Koopmans-Beckmann Problem (KBP). Some authors [16, 26] show this correspondence by defining

$$c_{ijpq} = \begin{cases} f_{ip} d_{jq} & \text{if } i \neq p \text{ or } j \neq q \\ a_{ij} + f_{ii} d_{jj} & \text{if } i = p \text{ and } j = q \end{cases} \tag{10}$$

However, careful consideration reveals the following facts. The product $f_{ip} d_{jq}$ is the cost of the flow from facility i at location j to facility p at location q. In the first part of (10) if $i \neq p$, constraint (2) implies that $j \neq q$, Conversely, if $j \neq q$, constraint (3) implies that $i \neq p$. The redundancy in (10) can be avoided by using the condition $i \neq p$ *and* $j \neq q$. We show the equivalence between the general quadratic assignment problem and the Koopmans-Beckmann problem by defining

$$c_{ijpq} = \begin{cases} f_{ip} d_{jq} & \text{if } i \neq p \text{ and } j \neq q \\ a_{ij} + f_{ii} d_{jj} & \text{if } i = p \text{ and } j = q \\ \infty & \text{otherwise} \end{cases} \tag{11}$$

With this definition of the c_{ijpq} elements, (7) can be deduced from (1). We can

obtain (9) from (1) by removing the a_{ij} term from (11). Further, if the distance matrix is symmetric, we have a symmetric KBP.

If there are no inter-facility flows (facility locations are independent of each other), we set $f_{ij} = 0$, $\forall$ i, j, and the problem in (7) reduces to the linear assignment problem (LAP):

(12) Minimize $\sum_{i,j} a_{ij}x_{ij}$

subject to (2), (3) and (8)

Finally, if in addition, * is required to be a cyclic permutation matrix, we obtain a traveling salesman problem.

1.2. REVIEW OF EXISTING ALGORITHMS

Most of the methods proposed in the literature, exact or heuristic, are designed to solve KBPs. Optimal algorithms for the KBP have been presented by Gilmore[11], Lawler[20], Land[19] and Gavett and Plyter[10], For a review of exact algorithms, see Pierce and Crowston[26] and Burkard[7], Various heuristic procedures have been devised for the KBP and can be grouped under two categories, viz. Constructive procedures and Iterative-improvement procedures. Some of the better known ones are those proposed by Gilmore[11], Steinberg[31], Armour and Buffa[1], Hillier and Connors[16], Gaschutz and Ahrens[9], Nugent, Vollman and Ruml[25], Burkard and Oerigs[5], and Graves and Whinston[12]. For a review and experimental comparison of heuristic techniques, see Nugent et al.[25], Ritzman[27], Liggett[22, 23] and Burkard and Stratmann[6].

The approaches suggested for the general QAP use linearization techniques to reduce the QAP to a linear programming problem.

1.2.1. INTEGER PROGRAMMING ALGORITHM

Lawler[20] has shown that the quadratic assignment problem of equations (1H4) can be represented by an equivalent integer programming problem as follows:

The n^2 variables x_{ij} can be linearized by defining n^4 variables y_{ijpq} where

$$y_{ijpq} = x_{ij}x_{pq}$$

The problem can then be stated as

(13) Minimize $\sum_i \sum_j \sum_p \sum_q c_{ijpq} y_{ijpq}$

subject to

(14) $\sum_j x_{ij} = 1$ $\forall i$

(15) $\sum_i x_{ij} = 1$ $\forall j$

(16) $\sum_{p} \sum_{q} y_{ijpq} = n^2$ $\forall i,j,p,q$

(17) $x_{ij} + x_{pq} - 2y_{ijpq} \geq 0$ $\forall i,j,p,q$

(18) $x_{ij} \in \{0,1\}$ $\forall i,j$

(19) $y_{ijpq} \in \{0,1\}$ $\forall i,j,p,q$

Thus, if the original problem has n^2 x_{ij} variables, the equivalent integer programming problem has n^2 x_{ij} variables plus n^4 y_{ijpq} variables. The number of constraints increases from 2n to $2n^4 + 2n$. This approach is evidently impractical for all but the smallest problems.

Lawler shows that the n^4 variables y_{ijpq} can be rearranged in an $n^2 \times n^2$ matrix (say Q) in such a way that the solution matrix(Y) to the LAP represented by Q is a feasible and hence optimal solution to the QAP if Y is a Kronecker second power of the implied n x n solution matrix(X) to the QAP. He suggests a way to enforce this condition by partitioning Q into n^2 minors of n^2 elements each and solving them as LAPs. These minors have a form similar to the subproblems defined for the method presented in this paper but provide comparatively inferior bounds.

1.2.2. MIXED INTEGER PROGRAMMING ALGORITHMS

1. Bazaraa and Sherali[3] suggested a linearization that results in a formulation with n^2 integer variables, $n^2(n-1)^2/2$ continuous variables and $2n^2 + 2n$ constraints. The authors applied Benders' Decomposition to this formulation but were unable to solve problems larger than n = 6 optimally within reasonable time.

2. Kaufman and Broeckx[17] have proposed the use of a linearization method suggested by Glover. To define this method we introduce the following two types of variables :

(20) $$w_{ij} = x_{ij} \sum_p \sum_q c_{ijpq} x_{pq}$$

(21) $$u_{ij} = \max \left[\sum_p \sum_q c_{ijpq} , 0 \right]$$

The QAP can then be reduced to the mixed integer problem of equations (22)-(27).

(22) $$\text{Min} \sum_i \sum_j w_{ij}$$

subject to

(23) $$\sum_j x_{ij} = 1 \qquad \forall\ i$$

(24) $$\sum_i x_{ij} = 1 \qquad \forall\ j$$

(25) $$u_{ij} x_{ij} + \sum_p \sum_q c_{ijpq} x_{pq} - w_{ij} \leq u_{ij} \qquad \forall\ i, j$$

(26) $$x_{ij} \in \{0,1\} \qquad \forall\ i, j$$

(27) $$w_{ij} \geq 0 \qquad \forall\ i, j$$

This formulation adds n^2 new continuous variables and n^2 new constraints. This gives a total of n^2 {0,1} variables plus n^2 continuous variables, and $n^2 + 2n$ constraints. The authors found that for a problem of size n = 8, the core requirements became too large for the mixed integer code they used and optimality could not be proved.

2. A BRANCH AND BOUND ALGORITHM FOR THE GENERAL QAP

In our algorithm, given that certain variables have been fixed equal to one, we obtain bounds by solving a relaxed problem in which costs are calculated correctly for the interaction between these fixed variables, but costs for the interaction between fixed and non-fixed variables as well as the interaction between pairs of free variables are estimated from the solution of LAPs.

We define cost elements of the form c_{hkij} and c_{ijhk}, for all i and j, as fixed-pair costs associated with the assignment h→k where we have fixed or set the variable x_{hk}=1. We make a distinction between fixing and setting a variable equal to one as follows. At level i of the branch and bound tree we have a partial assignment consisting of i variables - these variables are considered as *fixed*. The n-i unassigned indices can be assigned to any of the n-i free indices, permitting $(n-i)^2$ pairs, each

corresponding to a free variable. One of these free variables is *set* equal to one to define each of the subproblems described below, for the purpose of calculating lower bounds.

Suppose we want to calculate an initial lower bound, when no variables have been fixed. We define 11^ to be the subproblem obtained by setting x ^ i and considering only costs incurred by the pairing of X_{hk} and other x_{ij} variables, for all i and j, but *ignoring costs incurred by pairs of variables not involving the (h,k) pair.* This has the effect of linearizing the problem as well as decomposing it into n^2 linear assignment subproblems. In other words, within any subproblem, we consider only fixed-pair cost elements c_{WeIJ} and a_{IJnK} associated with x_{rHC}. We want to minimize the total cost of all assignments. Then subproblem FL^{nk} is

(28) $\text{Min } \{ Z_i Z_j (o_m \ll\bullet c_{jjhk}) x_{jj} \} - c^{hkhk}$
subject to (2), (3), (4) and

(29) $x_{hk} = 1$

Proposition 1: No cost element of the c_{IJIQ} form, q # j, or the c_{UPJ} form, p # i can ever be part of the cost of any solution.

Proof: For $c_{...}$ to be part of the cost of a solution we must have $x_{..} = 1$ *and* $x_{iq} = 1$, which is impossible under constraint (2). Similarly, c_{ijpj} can be part of the cost of a solution only if $x_{ij} = 1$ *and* $x_{pj} = 1$, which is ruled out by constraint (3). •

Discarding inadmissible cost elements, II_{hk} can now be simplified to

(30) $\text{Min } \{ Z_{j \# h} Z_{jft|c} (c^{\wedge}_{ij} + c_{jjhk}) x_{ij} \} + c_{hkhk}$

Proposition 2: If a cost element c_{ijpq} is part of the cost of a given solution, then c_{pqij} must also be part of the cost of that solution.

Proof: For c_{ijpq} to be part of the cost of a solution, we must have $x_{ij} = x_{pq} = 1$. This forces c_{pqij} to be part of the cost. •

Thus we can consolidate elements c_{ijnk} and $c^{\wedge}_{MKIJ}$ into one cost element

(31) $b^{hk}_{ij} = c_{....} \bullet c_{...}$ for i#h, j≠k

Ignoring the $c_{...}$ term, (31) can be written as the subproblem $II^*_{...}$

(32) $\text{Min} \sum_{j \neq h} \sum_{j \neq R} b_{j_j} x_{ij}$

subject to

(33) $\sum_{j \neq k} x_{ij} = 1 \qquad i \neq h$

(34) $\sum_{i \neq h} x_{ij} = 1 \qquad j \neq k$

(35) $x_{ij} \in \{0,1\} \qquad i \neq h,\ j \neq k$

We use an n = 4 example to illustrate the concept. The subproblems generated for the example problem would be

$$\Pi^*_{11}: \quad \begin{matrix} b^{11}_{22} & b^{11}_{23} & b^{11}_{24} \\ b^{11}_{32} & b^{11}_{33} & b^{11}_{34} \\ b^{11}_{42} & b^{11}_{43} & b^{11}_{44} \end{matrix}$$

$$\Pi^*_{21}: \quad \begin{matrix} b^{21}_{12} & b^{21}_{13} & b^{21}_{14} \\ b^{21}_{32} & b^{21}_{33} & b^{21}_{34} \\ b^{21}_{42} & b^{21}_{43} & b^{21}_{44} \end{matrix}$$

$$\Pi^*_{44}: \quad \begin{matrix} b^{44}_{11} & b^{44}_{12} & b^{44}_{13} \\ b^{44}_{21} & b^{44}_{22} & b^{44}_{23} \\ b^{44}_{31} & b^{44}_{32} & b^{44}_{33} \end{matrix}$$

Consider subproblem $11^$ It can be interpreted as stating the following problem. Given that we have set $x_{21} = 1$, the remaining indices (1, 3 and 4) can be assigned to indices *2.* 3 and 4. The rows of the matrix represent the n-1 indices remaining to be

assigned, the columns the n-1 free indices to which they can be assigned and the elements of the matrix are the costs of the corresponding assignments, ie., b^{*1}_{pq} is the cost incurred when x_{21} is *set* equal to one and if, in addition, x_{pq} is assigned equal to one. We wish to minimize the total cost of a complete assignment. It is obvious that II^*_2 Represents a linear assignment problem. The cost of the optimal solution r_{21} can be interpreted as a lower bound on the costs incurred by pairs of variables of the form (x_{21},x_{ij}), ie. costs of the form c_{21j_j} and c_{ij21}. In addition, this solution will imply assignments for indices 1, 3 and 4 *relative* to the linear assignment subproblem U^*_{21} Similarly, we obtain implied assignments in the solutions of the other subproblems. These assignments are obviously not necessarily consistent across subproblems.

Having shown that the subproblems can be reduced to ordinary (n-1)x(n-1) LAPs, we show next how the solutions to these subproblems can be used to construct the master problem whose solution provides a lower bound for the QAP. For the moment, we continue to ignore the $c_{u...}$ terms. Let r_{menu} be the cost of the optimal solution to Tl^*_{nk} The master probfem is

(36) Min $Z = \sum_{i,j} r_{ij} x_{ij}$

subject to (2K (3) and (4)

Proposition 3: The cost of the optimal solution to (35), say R*, is equal to twice the value of the lower bound for the QAP.

Proof: First, the solution to (36) implies assignments for all the indices. The relaxation obtained here is due to the fact that the assignments implied by the solutions to the subproblems nj^do not correspond in all cases to the assignments implied by the solution of the master problem (36). Second, consider a pair of assignments in the solution of (36), say *a->y* and *fi^r.* This means that R* contains r_{ay} and r_{fir} lfl tUm, r_{ay} contains term $b_{)Js}$ z_{ayfi} \ * °fi\ay' '*?' Where P WaS assigned to j in the solution of subproblem nib_{ar} Also r^*_{pr} contains term \s $b\pounds^{\wedge} = {}^{c}Braa + {}^{c}aQj3r' \wedge *^{Tt}$ *where a* is assigned to q in the solution of subproblem II^Thus the interaction between indices *a* and *J3* has been considered twice, as is the case for all pairs of indices. We thus have a lower bound whose value is equal to half R*. •

We next show how the same subproblems nj^ can be used after one or more assignment has been fixed so that these assignments are reflected in a consistent manner in the solutions of all the subproblems. As we descend a branch of the decision tree, a new assignment is decided at each level, and the procedure enforces this assignment in all the subproblems that need be considered. Referring again to our example problem, let us say that we have decided to fix the variable $*_{23}=1$ at level 1. We need consider only the subproblem that corresponds to the setting of index 2 to index 3 and those corresponding to the setting of indices 1, 3 and 4 to indices 1, 2 and 4. Subproblem II^*_{23} will have the same solution as before since no other variables have been fixed. However, in the 9 other subproblems that need to be solved we force $*_{23}=1$. This is done by barring all elements except $b'^{\wedge}_3$ in the appropriate row and column by putting them equal to a large number and solving II^*_{ij} again.

We can now formally state the method for calculating lower bounds. Reintroducing the $c_{...u}$ terms (these are the fixed costs of making assignment h-*k), we define $z_{hk} = c_{Nchk} * {}^{1/\wedge 2}{}_{hk*}$ Suppose we have n indices and at any stage of the procedure the indices already fixed are included in the set S and the indices to which they are fixed are in the set T. Further, index i€S is assigned to index $j^{s}{}^{\wedge}(i)$, j€T, ie., variable x_{ij} has been fixed equal to one. Then we can get a lower bound (LB) for all completions as follows :

$$LB - Z_{€\ s}\ z_{ijMi)} + r \tag{37}$$

Here the first part provides a lower bound on the costs incurred by variables already fixed. If S contains more than one index, the variables to be fixed in addition to $x_{hk}{}^{s}1$ are taken into consideration by forcing the corresponding element in the LAP matrix of nj^ into the solution. For example, if in addition, $x_{ij}=1$ has been fixed, we force element $b^{!*}_{ij}$ into the solution.

The second part, Z*, gives a lower bound on the costs incurred by the free variables where Z* is the solution of the LAP represented by matrix

$$Z = \{\ z_{..}\ |\ i \not\in s,\ j \not\in T\ \} \tag{38}$$

Here again any variables that have been fixed are forced into the solution while solving for each $z_{...}$

We use these bounds in a branch and bound procedure of the depth first type. The solution of the LAP represented by Z enables us to use a branching rule based on the regrets or alternate costs of an assignment. This not only provides a decision rule for choosing the next assignment but also helps in pruning the decision tree.

The calculation of the lower bound is done in two steps. First, we solve the necessary subproblems to get $z_{i\neq l(i)}$ for i G S and z_{ij} for i $\notin$ S, j $\notin$ T. Then we use the solutions to these subproblems to calculate a lower bound for the master problem.

2.1. BRANCHING RULE

The branching rule uses the concept of alternate costs as suggested by Lenstra[21] and is essentially the same as that used in Burkard's method. Let us say that we have solved the LAP in the master problem and found an optimal cost of Z* with facility i $\notin$ S assigned to location /'(i). We now use the dual variables of the optimal solution to reduce the matrix Z, ie. replace z_{ij} by $z_{ij}-u_i-v_j$. For every {i,/'(i)} element, which is now zero, we find the next smallest element in that row and column, and take their sum. This amount is the *regret* or minimum *additional* cost if assignment *ii,p{\)}* is not made. Also, this regret plus LB gives us the *alternate cost* of this assignment. At the next level of the decision tree we make that assignment which has the maximum regret, or, equivalently, the maximum alternate cost.

While backtracking, if the alternate cost is greater than the best known solution (present upper bound), the node can be fathomed and we backtrack to the next higher level. If not, we develop the next node at that level. Suppose the first assignment made at this level on the present branch is $x_{pq} = 1$. The objective is to cover either all possible assignments of p -> j, j $\notin$ T, or all possible assignments i -» q, i $\notin$ S. This is the most efficient way of fathoming the branch (possibly after improving the upper bound). This is achieved in two steps. First, the element z_{PQ} in Z is blocked out and the LAP is re-solved. The cost of this solution, if greater than the alternate cost, provides a second possibility of fathoming the node. Otherwise, we compute the regrets for the two new assignments in the row and column containing the blocked element and choose the one with the larger regret as the next assignment. If the next choice is from the row (column), then the third and all subsequent assignments are chosen from the same row (column), until we can backtrack to a higher level.

2.2. THE BRANCH AND BOUND PROCEDURE

We start the algorithm with an upper bound equal to a very large number. Proceeding as described above we descend to level (n-2) of the decision tree. At this stage, only two completions are possible and we compute the exact cost of these solutions. If the better one of these has a cost lower than the upper bound, the upper bound is set to this value and the corresponding assignments are stored in array MIN. When the tree has been completely fathomed and we return to level 0, all possible solutions have been enumerated implicitly, and the final upper bound along with the assignments stored in MIN give us the optimal solution. In case of alternate optima, the program stores only the first one but can be modified easily to list all alternate optimal solutions.

The form of the subproblems provides an easy method for tightening bounds while backtracking. Say we are developing the next node of the search tree at the current level along any branch. We can block out all cost elements of the form b_{ij}^{hk} in the subproblem Π_{hk}^{*} where the pair $i \rightarrow j$ is any assignment that has already been evaluated at that level, and solve the subproblems again. Computational experience with this rule showed that although the number of nodes evaluated did decrease, the overall solution time increased. This is due to the fact that the time required to solve the subproblems is so large that the value of the possibly better bounds it provides is negated. The final design of the algorithm, therefore, uses the initial solutions to the subproblems at a particular level along any given branch, and the subproblems are re-solved only when we branch again.

It is obvious that an assignment tried at level one (say $i \rightarrow j$) will never again be tried at any level of the tree. Thus when we return to level 0, we can scan the matrices of the subproblems and set each element of the form b_{ij}^{hk} to a large number before solving the subproblems again. (The subproblem Π_{ij}^{*} will, of course, not be considered again.) This modification did prove to be effective and has been incorporated in the algorithm.

Finally, since the largest part of solution time was the time taken to solve subproblems, we attempted to reduce the latter. As noted above, when we branch, we solve the subproblems again but force the new assignment in each solution. By storing the assignments for each subproblem, and checking whether the new assignment was already included, we found that it was often unnecessary to solve some of the subproblems again at the next level. This resulted in a saving of 15 to 20% in CPU time.

2.2.1. THE ALGORITHM

Step 0. Initialize: Set up subproblems II_{ij}. K = 0, UB = oo.

Step 7. Solve the subproblems to obtain the values z_{ij}. Solve the master problem to get initial lower bound. Reduce f z_{ij} I and calculate regrets. Choose an assignment with greatest regret, say P to Q. Alternate cost = Z·*«• regret.

Step 2. Assign x_{pQ} = 1. Replace S by SIMP}. T by TU{Q}, ^(P) = Q, K = K+1. Calculate the lower bound (LB) as in equation (37). If LB £ UB go to step 5. If K = N-2, go to step 4.

Step 3. Use dual variables to reduce Z and choose next assignment, say P to Q. If backtracking, choose next assignment from same row and/or column of Z as previous assignment. Alternate cost = LB • regret. Go to step 2.

Step 4. Calculate exact cost of two remaining completions. If smaller cost < UB, update UB and store assignments.

Step 5. If K = 0, STOP - optimal solution found. K * K-1. If alternate cost £ UB, repeat step 5. If K = 0, set all $b_{ij\cdot pQ}$ = oo and solve relevant subproblems. Set $^{Z}PQ = °°^{\#}$ Solve Z to get a tighter LB. If LB £ UB, repeat step 5. Otherwise, go to step 3. xs

3. COMPUTATIONAL EXPERIENCE

Bazaraa and Sherali solved the Nugent, Vollman and Ruml(NVR)[25] 5 facility and 6 facility problems in 6.9 and 154.4 sees, respectively on a CDC Cyber 70, using a Fortran IV code. The method was unable to prove optimality for larger problems in the cut off time of 770 sees.

Kaufman and Broeckx also mention computational experience with two of the NVR problems, using IBM's mixed integer code MPSX on an IBM 370/168. They solved the 6 facility problem in 157.8 sees, and the 8 facility problem in 240 sees, but in the latter case the solution was not proved to be optimal as the core requirements were too large for MPSX. The authors also attempted an application of Bender's Decomposition to their mixed integer programming formulation but report that the results were disappointing.

For the purpose of comparison we solved to optimality the NVR 5, 6, 7 and 8 facility problems after converting the flow and distance data into the general QAP format. The times required on a DEC-20, using a Fortran program, were respectively 0.32, 1.83, 3.87 and 28.13 sees. The remaining computational testing was done on randomly generated problems. The elements c_{ijpq} were chosen from a uniform distribution between 0 and and a maximum value called MAX that was varied. AJl computational results mentioned in this paper were obtained on a DEC-20 using Fortran, and average results are shown in Table I.

Size(n)	Values of MAX	Problems solved	Average solution < time (sees)
6	10,40.70,100	40	1.28
7	10,40,70,100	40	5.49
8	1,2,3,5,10. 15.20,30,40, 60,80,100	120	31.43
9	30.70	20	220.91
10	50	3	1305.65

Table I. Computational experience with the code written for the algorithm in Section 2.2.1. All the problems were randomly generated as described in section 3. All problems were solved to optimality.

First, we studied the effect of the range of cost coefficients. With n = 8, we solved sets of 10 problems each with MAX set at 1, 2, 3, 5, 10, 15, 20, 30, 40, 60, 80 and 100 resp. The average times for the sets varied between 26.24 and 35.12 sees, with an average for the 120 problems of 31.43 sees. There was no discernible trend in the solution time as the cost range was varied.

Next, we developed a branch-and-bound procedure based on Lawler's definition of subproblems, using the same method to approach feasibility and the same branching rule as described above. We compared the solution times with our method for 10 problems each with n = 6, MAX = 30; n = 7, MAX = 50; and n = 8, MAX = 70. On

the average our method required 63, 45 and 36% resp. of the time required by the code we wrote based on Lawler's method. We observed that the comparative advantage of the algorithm given in Section 2.2.1 increases with increasing size of problem.

4. CONCLUSIONS

In this paper an exact branch and bound algorithm for solving general QAPs based on LAP relaxations has been described and compared to other methods of solving the general QAP. Computational experience was given that showed that the method was capable of solving to optimality problems of sizes up to n = 10 in reasonable computation times. This size of problem is larger than those previously reported in the literature. Nevertheless, it is still relatively small and further work is needed on this problem.

REFERENCES

1. Armour G.C. and Buffa E.S., "A Heuristic Algorithm and Simulation Approach to Relative Location of Facilities", *Management Science,* Vol.9, No.2, January 1963.

2. Bazaraa M.S. and Elshafei A.N., "An Exact Branch-and-Bound Procedure for the Quadratic Assignment Problem", *Naval Res. Logistics Quart.,* Vol.26, 1979.

3. Bazaraa M.S. and Sherali H.D., "Benders' Partitioning Scheme Applied to a New Formulation of the Quadratic Assignment Problem", *Naval Res. Logistics Quart..* Vol.27, 1980.

4. Buffa E.S., Armour G.C. and Vollmann T.E., "Allocating Facilities with CRAFT", *Harvard Business Review.* March-April 1964.

5. Burkard R.E. and Derigs U., *Assignment and Matching Problems: Solution Methods with Fortran Programs,* Lecture Notes in Economics and Mathematical Systems, ,VOI.184, Springer-Verlag, Berlin, 1980.

6. Burkard R.E. and Stratmann K.H., "Numerical Investigations on Quadratic Assignment Problems", *Naval Res. Logistics Quart.,* Vol.25, 1978.

7. Burkard R.E., "Some Recent Advances in Quadratic Assignment Problems", Presented at the *International Congress on Mathematical Programming,* Rio de Janeiro, April 6-8, 1981.

8. Elshafei A.N., "Hospital Layout as a Quadratic Assignment Problem", *Operational Research Quart.,* Vol.28, 1977.

9. Gaschutz G.K. and Ahrens J.H., "Suboptimal Algorithms for the Quadratic Assignment Problem", *Naval Res. Logistics Quart.,* Vol.15, 1968.

10. Gavett J.W. and Plyter N.V., "The Optimal Assignment of Facilities to Locations by Branch and Bound", *Operations Research,* Vol.14, 1966.

11. Gilmore P.C., "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem", *Journal of SIAM.* Vol.10, 1962.

12. Graves G.W. and Whinston A.B., "An Algorithm for the Quadratic Assignment Problem", *Management Science,* Vol.17, No.7, 1970.

13. Hanan M. and Kurtzberg J.M., "A Review of the Placement and Quadratic Assignment Problems", *SIAM Review.* Vol.14, No.2, 1972.

14. Heider C.H., "An n-Step, 2-Variable Search Algorithm for the Component Placement Problem", *Naval Res. Logistics Quart..* Vol.20, 1973.

15. Hillier F.S., "Quantitative Tools for Plant Layout Analysis", *Journal of Ind. Eng..* Vol.14, 1963.

16. Hillier F.S. and Cpnnors M.M., "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities", *Management Science.* Vol.13, No.1, 1966.

17. Kaufman L and Broeckx F., "An Algorithm for the Quadratic Assignment Problem Using Benders' Decomposition", *European Journal of Operational Research.* Vol.2, 1978.

18. Koopmans T.C. and Beckmann M., "Assignment Problems and the Location of Economic Activities", *Econometrica,* Vol.25, Jan. 1957.

19. Land A.H., "A Problem of Assignment with Inter-related Costs", *Operational Research Quart.,* Vol.14, June 1963.

20. Lawler E.L, "The Quadratic Assignment Problem", *Management Science.* Vol.9, July 1963.

21. Lenstra, *Sequencing by Enumerative Methods.* Mathematisch Centrum, Amsterdam, 1977, Pages 97-100.

22. Liggett R.S., "The Quadratic Assignment Problem: an Analysis of Applications and Solution Strategies", *Environment and Planning,* Vol.7, 1980.

23. Liggett R.S., "The Quadratic Assignment Problem: an Experimental Evaluation of Solution Strategies", *Management Science.* Vol.27, No.4, 1981.

24. Little J.D.C., Murty K.G., Sweeney D.W., and Karel C, "An Algorithm for the Traveling Salesman Problem", *Operations Research.* Vol.11, Nov-Dec 1963.

25. Nugent C.E., Vollmann T.E. and Ruml J., "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations", *Operations Research,* Vol.16, Jan-Feb 1968.

26. Pierce J.F. and Crowston W.B., "Tree Search Algorithms for Quadratic Assignment Problems", *Naval Research Logistics Quart.,* Vol.18, 1971.

27. Ritzman L.P., "The Efficiency of Computer Algorithms for Plant Layout", *Management Science,* Vol.18, No.5, 1972.

28. Sahni S. and Gonzales T., "P-complete Approximation Problems", *Journal of the Assn. for Computing Machinery,* Vol.23, No.3, 1976.

29. Scriabin M. and Vergin R.C., "Comparison of Computer Algorithms and Visual Based Methods for Plant Layout", *Management Science,* Vol.22, No.2, 1975.

30. Smith T.H.C, Unpublished Ph.D. thesis, G.S.I.A., Carnegie-Mellon University, 1976.

31. Steinberg L, "The Backboard Wiring Problem: A Placement Algorithm", *SIAM Review.* Vol.3, No.1, 1961.

32. Vollmann T.E. and Buffa E.S., "The Facilities Layout Problem in Perspective", *Management Science.* Vol.12, No.10, 1966.