Tree Elaboration Strategies
In
Branch and Bound Algorithms
For Solving the
Quadratic Assignment Problem

by

Peter M. Hahn, The University of Pennsylvania
and Sci-Tech Services, Inc.

William L. Hightower, Elon College

Terri Anne Johnson, Elon College

Monique Guignard-Spielberg,
The University of Pennsylvania

Catherine Roucairol,
The University of Versailles

ABSTRACT

This paper presents a new strategy for selecting nodes in a branch-and-bound algorithm for solving exactly the Quadratic Assignment Problem (QAP). It was developed when it was learned that older strategies failed on the larger size problems. It is a variation of the polytomic depth-first search strategy of Mautor and Roucairol which extends a node by all assignments of an unassigned facility to unassigned locations based upon the counting of 'forbidden' locations. A forbidden location is one where the addition of the corresponding leader element would increase the lower bound beyond the upper bound. We learned that this fortuitous situation never occurs near the root on Nugent problems larger than 15. One has to make better estimates of the bound if the strategy is to work. We have therefore designed and implemented an increasingly improved set of such bound calculations. The better of these bound calculations to be utilized near the root and the less accurate (poorer bounds) utilized further into the tree. The reason being that calculations of better bounds take longer and are needed at the root but not needed further into the tree. The result is an effective and powerful technique for shortening the run times of problem instances in the range of size 16 to 25. Run times were decreased generally by five- or six-to-one and the number of nodes evaluated was decreased as much as 10-to-one. For problem instances beyond size 25, we are convinced that even better and thus more computationally costly lower bound estimates will be required.

I. Introduction

The Quadratic Assignment Problem (QAP) is arguably the most difficult NP-hard combinatorial optimization problem. Solving general problems of size greater than 30 (i.e., with more than 900 (0-1) variables) is still computationally impractical. If among exact algorithms, branch-and-bound are the most successful ones, the lack of a sharp lower bound technique in these algorithms is one of the major difficulties.

Recently, Hahn and Grant [19] published a dual procedure (DP), which derives from a Lagrangean relaxation of the linearized QAP and yields strong bounds more efficiently than other known methods. Concurrently, Hahn, Grant and Hall [20] devised a branch and bound algorithm, which takes advantage of problem reformulation to make the bounds easier to calculate and provides for early pruning. These combined methods have made it possible to solve exactly heretofore-unsolved problems of size 30.

During attempts to solve QAPs of size greater than 20, it was determined that many of the techniques and strategies for branch-and-bound enumeration suggested in prior works [5, 9, 4, 25, 31, 34, and 35] were not helpful. Thus, it became necessary to work out new and better branching schemes. We report here on the strategies that work on problems ranging from size 16 to 30 and provide a roadmap for the solution of even larger problems. The lessons learned may apply to combinatorial optimization problems of similar difficulty.

II. The Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) is one in which N units have to be assigned to N sites in such a way that the cost of the assignment, depending on the distances between the sites and the flows between the units, is minimal. It can be formulated as follows:

Given two NxN matrices, $F = [f_{ij}]$ with f_{ij} the flow between units i and j, and $D = [d_{kl}]$ with $d_{kl\ the}$ distance between sites k and l, find a permutation p of the set S = {1,2,...,N} which minimizes the global cost function, $\text{Cost}(p) = \sum_{i=1,..,n} \sum_{j=1,..,n} f_{ij} d_{p(i)p(j)}$

The quadratic assignment problem is NP-hard. But, this theoretical complexity is not sufficient to explain the results described above, as we can now solve exactly very large instances of a great number of NP-hard problems. The homogeneity of the values of the solutions for most of the applications, due to the structure of the problem (scalar product of the two matrices) is a more convincing explanation. Indeed, first, we have a lot of solutions whose value is close to the optimum. So, even when the best solution is obtained, it is very hard to prove its optimality. Then, fixing one assignment has a low influence on the average value of the solutions. Even when going down in the branch-and-bound tree, the problem remains very hard. Moreover, it is difficult to prune important branches. Related to this aspect, the computation of the lower bound is one of the major difficulties. Indeed, the bound is either too loose (the number of nodes of the search tree becomes huge), or the time needed to compute the bound of a node is prohibitive.

The oldest and most frequently used bound, developed independently in 1962 by Gilmore [14] and in 1963 by Lawler [28], is quickly computed in O(n3) but the results are not very tight. Using Gilmore-Lawler bounds, sub-problem evaluation is more than 20% away from the best known solution for Nugent's problems of size greater than 20

Because of these results, many other lower bounds have been developed and tested. These

- either use an eigenvalue approach (Finke, Burkard and Rendl 1987 [12], Rendl and Wolkowicz 1992 [36]),
- or rely on equivalent dual formulations of the original QAP (Assad and Xu 1985 [3], Carraresi and Malucelli 1992 [7]),
- or propose a new class of lower bounds based on optimal reduction schemes (Li, Pardalos, Ramakrishnan and Resende 1994 [30]),
- or take advantage of regular grid properties but, obviously, can be used only when the sites lie on a regular grid (Chakrapani and Skorin-Kapov 1993 [8]).
- or are now the best of the known general lower bounds, the Interior Point Linear Programming (IPLP) bound, of Resende, Ramakrishnan and Drezner [37].

The more recent bounds in this list are closer to the best value than the Gilmore-Lawler bound. But, their usefulness remains limited when dealing with large problems. However, the recently published bounds of Hahn and Grant (denoted HGB) surpass all the above bounds in terms of both quality and computing speed as described in Hahn and Grant [19]. A recent comparison by Karisch, et al. [26] of the Hahn and Grant bounds to other linearization-based bounds confirms their superiority.

These new bounds have been incorporated into a unique branch-and-bound algorithm which has been used by Hahn, et al. [20] to solve in record times many of the difficult QAP instances posed over 30 years ago by Nugent, et al. [33].

III. The Hahn-Grant Bound

The formulation of the quadratic assignment problem (QAP) may be stated as follows. Given N^4 cost coefficients C_{ikjn} (i,k,j,n=1,2,...,N) determine an N x N solution (i.e., permutation) matrix

$$\mathbf{U} = [u_{ab}] \qquad (1)$$

called an "assignment", so as to minimize a cost function,

$$R(\mathbf{U}) = \sum_{ijkn} C_{ijkn}\ u_{ij}\ u_{kn} \qquad (2)$$

Now, suppose we arrange the N^4 cost coefficients C_{ikjn} in an $N^2 x N^2$ matrix **C** as shown in Figure 1. The asterisks denote disallowed elements, i.e., elements that cannot be included in any feasible solution.

It can be shown that those elements of matrix **C**, which contribute to the cost R(U) of an assignment **U** are confined to one submatrix in each row and one submatrix in each column. Furthermore, within those submatrices, only one element from each submatrix row and one element from each submatrix column contribute to the cost R(**U**).

In each submatrix, the element occupying a position corresponding to the submatrix position in **C** is unique, because, if the submatrix contributes to an assignment, that element must be included. It is termed the 'leader' and lies at the intersection of the starred submatrix row and column.

$$
\mathbf{C} = \left(\begin{array}{c|c|c}
\mathbf{C}_{11} & \mathbf{C}_{12} & \mathbf{C}_{13} \\ \hline
\mathbf{C}_{21} & \mathbf{C}_{22} & \mathbf{C}_{23} \\ \hline
\mathbf{C}_{31} & \mathbf{C}_{32} & \mathbf{C}_{33}
\end{array}\right)
= \left(\begin{array}{ccc|ccc|ccc}
C_{1111} & & & & C_{1212} & & & & C_{1313} \\
& C_{1122} & C_{1123} & C_{1221} & & C_{1223} & C_{1321} & C_{1322} & \\
& C_{1132} & C_{1133} & C_{1231} & & C_{1233} & C_{1331} & C_{1332} & \\ \hline
* & C_{2112} & C_{2113} & C_{2211} & & C_{2213} & C_{2311} & C_{2312} & \\
C_{2121} & & & & C_{2222} & & & & C_{2323} \\
& C_{2132} & C_{2133} & C_{2231} & & C_{2233} & C_{2331} & C_{2332} & \\ \hline
& C_{3112} & C_{3113} & C_{3211} & & C_{3213} & C_{3311} & C_{3312} & \\
& C_{3122} & C_{3123} & C_{3221} & & C_{3223} & C_{3321} & C_{3322} & \\
C_{3131} & & & & C_{3232} & & & & C_{3333}
\end{array}\right)
$$

Figure 1. Matrix of costs for N = 3.

It is further shown in [17] that certain operations may be performed on $\mathbf{C} = (C_{ikjn})$ which will change the cost R(**U**) of assignments **U** in such a way that all assignment costs are shifted by an identical amount, thus preserving their order with respect to cost. These operations are divided into two classes (see Grant [15] for proofs):

Class 1: Addition (or subtraction) of a constant to all feasible elements of a submatrix ($\mathbf{C}_{ij}$) row or column and the corresponding subtraction (or addition) of this constant from either another row or column of the submatrix or from the submatrix leader element.

Class 2: Addition or subtraction of a constant to all feasible elements of any row or column in matrix **C**.

Class 1 operations maintain the cost of all assignments, but permit redistribution of element costs within a given submatrix. In contrast, Class 2 operations work on the matrix level, and change the cost of all feasible assignments by the amount added to or subtracted from the matrix row or column. A better understanding of these properties is provided in Hahn [17].

The discovery that Class 1 and 2 operations on matrix **C** serve to shift the cost R(**U**) of all assignments by an identical amount was provocative. If the operations on **C** decrease the cost by an amount R' and are performed in a way that keeps the elements of modified matrix **C'** non-negative, then no assignment cost can become negative and the following relationship holds:

$$R' \le R(\mathbf{U})_{min} \qquad (3)$$

Thus, R' constitutes a valid lower bound on the QAP. Finally, if R' can be increased until the equality holds, then the elements of the adjusted cost matrix involved in an optimum assignment would necessarily be zero. Thus a dual for the Quadratic Assignment problem can be stated: Maximize the sum of downward cost shifts R' permitted by Class 1 and 2 operations, such that no cost element in **C'** is driven negative.

In developing the HGB, it was recognized that this approach had roots in the Hungarian algorithm (Munkres [32]) for solving linear assignment problems (LAPs). The matrix reduction methods used in the Hungarian algorithm are essentially the Class 1 and Class 2 operations described above. Hahn adopted the Hungarian algorithm as the core set of operations for the HGB, applying the algorithm in a manner that extended its utility and made use of the underlying interaction between elements. The HGB is described more fully in [19].

The reductions that take place in matrix **C'** result in a reformulation of the original QAP; one that approaches solution of the QAP from the dual perspective. In some fortuitous cases the QAP is actually solved. This happens quite often during its use in a branch-and-bound context.

IV. The Dual Procedure Branch-and-Bound Algorithm (DPB&B)

The DPB&B algorithm is developed in a manner consistent with the HGB. We follow the conventional technique of selecting a single facility-location assignment as the first (highest) level as well as subsequent levels of partial assignment. In order to implement this selection, a linear cost is chosen to be involved in the assignment. For instance, we might choose the upper-leftmost linear cost. Referring to Figure 1, this would be element C_{1111} , implying facility 1 is assigned to location 1 (i.e., $v_{1111} = 1$).

Based on the selection of linear cost C_{ijij}, submatrix $\mathbf{C}_{ij}$ is involved in the assignment. The submatrices remaining in the row and the column that contain submatrix $\mathbf{C}_{ij}$ disappear (as they cannot be involved in the assignment) and the problem is thus reduced to a QAP of size N-1. One consequence of this reduction is that any symmetry in the original problem disappears as well. It turns out that one row and one column likewise disappear from each submatrix of the original problem, with the exception of the original submatrix $\mathbf{C}_{ij}$, which remains N-1 in size. To complete the formulation of the newly formed N-1 problem, this submatrix is added (by simple matrix addition) to the now size N-1 matrix of linear costs.

It is the application of the HGB on the newly formed N-1 size problem that attempts to fathom a partial assignment postulated by the selection of linear cost C_{ijij}. By fathoming, one calculates a lower bound and tests it against the best-known upper bound. If the best-known upper bound is exceeded, the partial assignment is eliminated from the problem.

You may recall from above, the HGB moves costs out of the **C** matrix into a lower bound value, leaving a modified matrix **C'**. For subsequent branch-and-bound operations along a given partial assignment path, our strategy is to take advantage of this fact and to use this reduced cost matrix **C'** for setting up subsequent sub-problems deeper into the tree. Thus, lower bounds are calculated not from the original problem, but from the sub-problems that were already processed by the HGB at earlier (higher) levels of partial assignment. Using the modified matrix **C'** of each of these sub-problems has the additional benefit that the sub-problem is brought closer to dual solution, making it more likely that a sub-problem will be solved by the HGB and assuring that the tree will be pruned earlier in the branch-and-bound process. This innovative ‘reformulation technique’ is responsible for the impressive performance of the algorithm.

In implementing the HGB, the choice of m (number of dual iterations) at a given partial assignment became a dynamic decision, based on the closeness of the achieved lower bound after a fixed number of iterations. It had been determined experimentally that lower bounding performance early in the DP was an excellent (though not infallible) predictor of the lower bound that could eventually be reached. Thus, if after a small number of iterations at the current partial assignment, a threshold upper bound were not reached, the lower bounding attempt would stop and the algorithm would proceed to make an additional assignment. The choice of this threshold was determined experimentally.

V. Branching Strategy

In the DPB&B algorithm, the branch-and-bound tree is elaborated by extending assignments of a given facility to all locations or vice-versa. If fathoming fails to eliminate a partial assignment at a given level an additional partial assignment is made. Thus, the algorithm continues to reduce the problem size by one level at a time, clearly a depth-first search. In our experience with test instances of size 20 and less, it was never necessary to go beyond the 10th level into the tree, as fathoming always succeeded on or before this level or an attractive feasible solution had already been found. In fact, when solving the Nugent size 20 for the first time, it was necessary to go to the tenth level but once. This pattern of needing to search no more than N/2 levels has been more or less true for all problem instances. However, when the new branching strategies were applied on the largest test instances (N=25 and N=30), it often took several levels more than N/2 levels to locate the optimum feasible solution.

The DPB&B algorithm lends itself to depth-first search mainly because the bounding calculation involves reformulation of the QAP and its sub-problems. Each lower bounding calculation at a node of the tree (i.e., for a given partial assignment) is an attempt to solve a reduced size QAP (i.e., a sub-problem of the original QAP) from a dual perspective. Further branching from that node begins with the already reduced costs of that bounding calculation. Thus, the cost matrix of the reformulated problem must be stored prior to further branching. With depth-first branching, only one cost matrix per assignment level needs to be stored. If instead we were to use best-first branching, the memory required for a large number of QAP cost matrices would be prohibitively large. Fortunately, as explained in [10], depth-first search strategy for the QAP has the additional advantage that it is superior to best-first search strategy.

In branch-and-bound, if a node cannot be fathomed (i.e., eliminated), it is necessary to branch further (i.e., examine the children of that node). In the QAP, not all the children need be examined. One must examine only the children corresponding to either: (a) assignments of all unassigned facilities to given unassigned location or (b) assignments of a given unassigned facility to all unassigned locations. If then the children in (a) or (b) can be fathomed, the parent node is automatically eliminated.

We noticed early in our experimental work on branching strategies that the run-time of the DPB&B algorithm is very much dependent on the selection of the facility or location on which to branch. Selecting the facility or location on which to branch is important at all nodes that cannot be fathomed without further elaboration. This led us to conclude that careful facility(location) selection should be performed at all levels in the tree.

VI. Facility (Location) Selection

A number of measures have been devised in earlier investigations [5, 4, 9, 25, 31, 34, and 35] for making the choice of the best facility or location on which to branch. Many of these concentrated on the search for a good feasible solution to the QAP and thus were directed toward the avoidance of high cost assignments. These strategies are not designed for solving large QAPs in reasonable time and thus were summarily rejected.

What seemed to work for our purposes, was the polytomic depth-first search strategy of Mautor and Roucairol [31]. This strategy extends a node by all assignments of an unassigned facility to unassigned locations based upon the counting of 'forbidden' locations. A forbidden location is one where the addition of the corresponding leader element would increase the lower bound beyond the upper bound. We found, however, that this fortuitous situation never occurs near the root on Nugent problems larger than N=7. Nevertheless, the Mautor-Roucairol (M-R) strategy improved runtime for problem sizes up to N=15. Beyond that, the strategy failed to give an improvement.

We concluded that the selection of a facility (or location) on which to branch is an exercise in making a lower bound calculation for every child of the node. At the L-th level of the tree (i.e., where L partial assignments have been made) the number of children is M-squared; M being both the number of unassigned facilities and the number of unassigned locations. M=N-L where N is the original size of the QAP. These bound values are arranged in a square matrix and a row or column of this matrix is selected. The use of this matrix in guiding the tree elaboration is illustrated in Figure 2.

The four steps shown in Figure 2 are generally, but not always, sequential and the entire process is not described. They are, however, the major steps in tree elaboration discussed earlier in Section V. Not mentioned in Figure 2 is the situation where a feasible solution is discovered. In such case, the solution value must be equal to or lower than the upper bound. Whether the solution value is equal to or lower than the upper bound, the solution is recorded, the upper bound improved (if applicable) and the elaboration continued along the same facility (location).

VII. Single Extension Bound Choice.

By Single Extension Bound Choice (SEBC) we mean the method of calculating bounds for the purpose of making facility (location) tree elaboration decisions. As mentioned in Section VI, what appeared to work for this purpose on the smaller problem instances ($N \leq 16$) was the concept of counting 'forbidden' locations, which uses the leader of the facility/location pair as a bound estimate. We worked to extend the usefulness of the concept by increasing the accuracy of the bound estimate. The steps we took were based on calculations already implemented for the DP bounding algorithm described in [19].

Though not described explicitly in [19] nor in [20], the DPB&B algorithm is calculated in a way that those components to the bound calculation that contribute the largest values and require the shortest computational steps are done first. These are followed by the more tedious calculations (i.e., those comprising the LAPs in the NxN cost submatrices.)

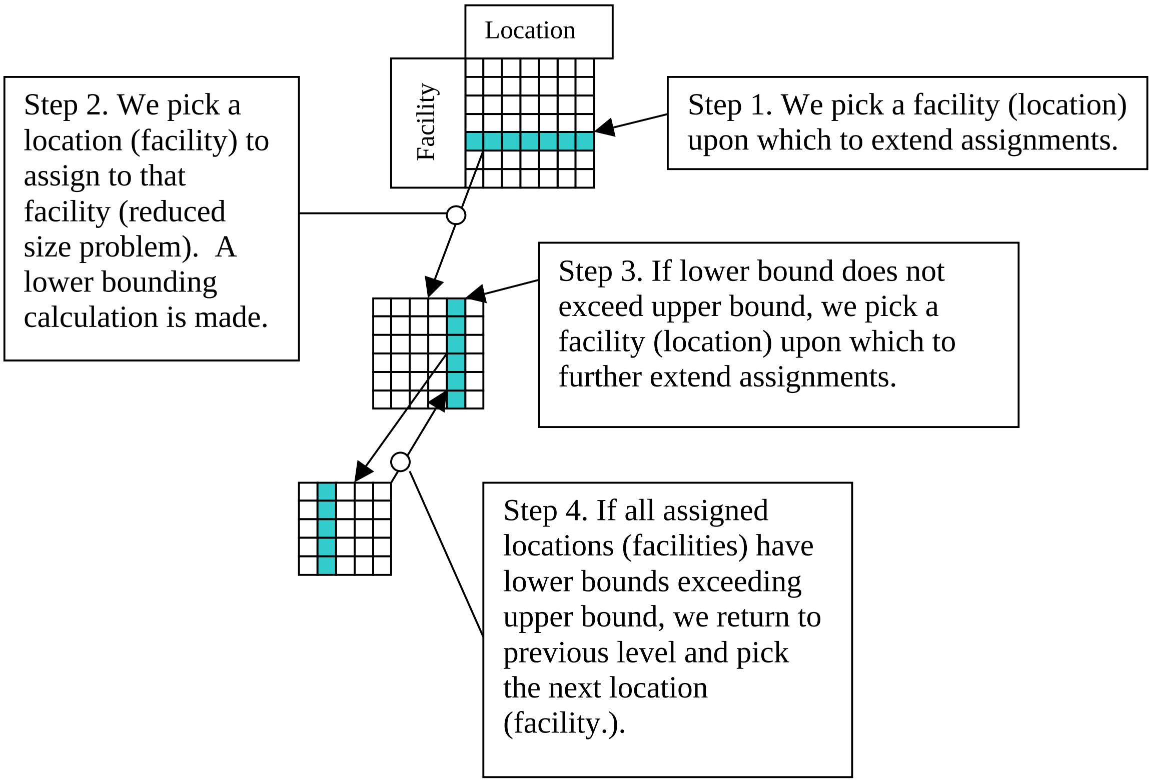

Figure 2 - Tree Elaboration Strategy

The first improvement we made on the 'forbidden' location concept was the following. Recall that we extend the assignment by one. Now, for each facility/location pair we generate a new (reduced-by-one) leader matrix and perform a Linear Assignment Problem (LAP) on this matrix. The amount drawn out of the reduced size leader matrix (i.e., its solution value) is added to the old leader producing an improved bound. This new bound was not appreciably better than the original 'forbidden' location concept and was not pursued any further.

An even better bound was produced by adding the submatrix corresponding to the extended assignment to the new leader matrix, then performing the LAP, and adding this solution value to the bound. We call this new bound M-R PLUS 2. It was dramatically effective for tree elaboration purposes in the Hadley 16. However, when we tried this technique on the Hadley 18 and the Nugent 20, the runtime and node evaluation improvements were disappointing. We learned that M-R PLUS 2 bounds calculated for the larger instances were inadequate for predicting which facility (location) was the best choice for elaboration at and near the root.

For problem sizes beyond size 20, it was necessary therefore to utilize the HGB bounds themselves. Not only are HGB bounds required, but also several iterations are necessary to get sufficiently good information for branching decisions at or near the root of the tree. Fortunately, once we get several levels into the tree, we can very effectively use the M-R-PLUS-2 bounds that take much less time to calculate.

When experimenting with larger problem instances, we found that we could save considerable computational time (for guidance of tree elaboration) by calculating bound values only for certain facility/location pairs. In order to understand this, arrange facility/location pairs in a square matrix. Then, calculate bounds for elements in rows and columns in a way that gives an almost equal number of bounds in every row and every column. Along each row (facility choice) and column (location choice), a bound average is calculated giving a measure of the difficulty of fathoming branches along that row or column. Figure 3 gives two examples of the selection of elements in a square matrix that would satisfy this rule.

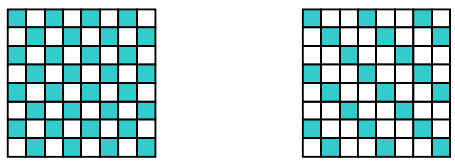

Figure 3 - Sampling the facility/location pair matrix.

We call this bounding technique HGBSMPL-2 or HGBSMPL-3 depending on whether every second or every third element in the facility/location matrix is included in the bound calculations. We call it HGBSMPL-i when the i-th element is sampled.

VII. Experimental Results

For the purposes of testing branching strategies the following rules were implemented for selecting the best facility (or location) upon which to extend the tree:

1. Uppermost facility (i.e., facility 1).
2. Facility chosen at random.
3. Facility (location) with the most locations (facilities) with bounds above the median.
4. Highest total bound for a given facility (location).
5. Highest average bound for a given facility (location).

Rule 1 is the way that the DPB&B algorithm had been originally written, as described in [20]. Rule 2 was implemented for the purposes of confirming that without a good branching strategy, the runtime and the number of nodes would be much higher. Rules 3 and 4 made sense as measures of the difficulty of elaborating the corresponding portion of the tree. Rule 5 was added to deal with the situation when some of the locations along a row or facilities along a column may have previously been disallowed or excluded. Such exclusions happen, for example, in the case of Nugent problem instances with symmetries that can be exploited. Rule 5 is required for the HGBSMPL-i bounds.

We tested these rules on the Hadley 16, Hadley 18, Nugent 20, Nugent 22 and Nugent 24 instances. The results of these tests are given in Tables 1 through 5 below. As our computational resources were limited, where a rule was clearly ineffective on the smaller instances we did not implement it on the larger ones. Note that each table begins with the choice of the uppermost facility (i.e., facility 1) as the facility/location selection method. This corresponds to the exact results published in [20] for the original DPB&B version of our algorithm. The tabulated results are essentially in the order in which the tests were performed.

Table 1. Hadley 16 Experiments

Facility Location Selection Method	Single Extension Bound Computation	Nodes Evaluated	Runtime in Seconds
Uppermost facility	None	13,549	879
Random facility or location	None	17,199	955
Max sum of bounds	M-R	13,481	614
Max # above median	M-R	12,892	1,119
Max sum of bounds	M-R-PLUS-2	3,069	206
Max sum of bounds	HGB (1 iteration)	2,502	1,623
Max # above median	HGBSMPL-2 (1 iteration)	4,045	1,397
Max # above median	HGBSMPL-2 (2 or 3 iterations)	4,021	1,570

Table 2. Hadley 18 Experiments

Facility Location Selection Method	SEBC near root	SEBC after level L, where root = level 0	Number of Nodes Evaluated	Runtime in Seconds
Uppermost facility	None	None	197,487	33,666
Max sum	M-R	SAME	359,962	22,967
Max sum	M-R-PLUS-2	SAME	53,224	4,362
Max # above median	HGB 1 iteration	SAME	42,607	42,076
Max # above median	HGBSMPL-6 1 iteration	SAME	67,719	12,859
Max # above median	HGBSMPL-3 1 iteration	SAME	66,288	20,498
Max # above median	HGBSMPL-2 1 iteration	M-R-PLUS-2 after root	66,103	5,914

SAME = Same as at root

Table 3. Nugent 20 Experiments				
Facility Location Selection Method	SEBC near root	SEBC after level L, where root = level 0	Number of Nodes Evaluated	Runtime in Seconds
Uppermost facility	None	None	724,289	48,578
Max average	HGB at root HGBSMPL-3 (3 iterations)	M-R-PLUS-2 after level 1	608,258	29,764
Max average	HGB at root HGBSMPL-3 (3 iterations)	M-R-PLUS-2 after level 2	239,449	23,645
Max average	HGB at root HGBSMPL-3 (3-iterations)	M-R-PLUS-2 after level 3	207,157	27,054

Table 4. Nugent 22 Experiments				
Facility Location Selection Method	SEBC near root	SEBC after level L, where root = level 0	Number of Nodes Evaluated	Runtime in Seconds
Uppermost facility	None	None	10,768,366	1,812,100
Max average	HGB at root HGBSMPL-2 (3 iterations)	M-R-PLUS-2 after level 3	988,302	293,427

Table 5. Nugent 24 Experiments				
Facility Location Selection Method	SEBC near root	SEBC after level L, where root = level 0	Number of Nodes Evaluated	Runtime in Seconds
Uppermost facility	None	None	49,542,338	4,859,940
Max average	HGB at root HGBSMPL-2 (3 iterations)	M-R-PLUS-2 after level 3	11,674,955	1,135,610 (13.14 days)
Max average	HGB at root HGBSMPL-2 (3 iterations)	M-R-PLUS-2 after level 5	5,629,849	2,791,380

In Table 1 for the Hadley 16 test instance, we see that the choice of random facility requires a larger number of nodes and longer runtime than for the original algorithm. This would be expected. In trying the Mautor-Roucairol technique, a small runtime advantage was observed for the maximum sum of bounds rule but not for the maximum number of bounds above median rule. At this point in time the M-R-PLUS-2 bound was developed and gave clearly outstanding results. The remaining experiments in Table 6 were performed after the experiments with the larger test instances. These results make clear that the more powerful elaboration guidance techniques required for the larger test instances are overkill for smaller problems. Note that the number of nodes evaluated in the last three cases is impressively small whereas the runtime is almost doubled.

In the Hadley 18 experiments (Table 2) again the original M-R technique gave a small improvement. The M-R-PLUS-2 bound again gave outstanding results. The remaining four experiments gave increasingly better performance as we learned how to use the better bounds. Clearly, the use of HGB gave an impressive (almost 5:1) saving in the number of nodes evaluated, but the run still took too long. When we used the bound sampling techniques, results improved. Node count went up but runtime savings went down even more. Finally, we learned that the better bounds needed near the root are not useful further into the tree. The combination of HGBSMPL-2 and M-R-PLUS-2 became the standard for further experimentation.

While the experiments in Table 2 used predominantly the maximum number of bounds above the median, it was quickly determined that this rule was inferior to the maximum average bound rule. This was especially true for the Nugent problem instances. This was confirmed experimentally by partially enumerating the tree. Complete enumeration would have wasted our limited computational resources. Thus, in Tables 3, 4 and 5 only the maximum average rule was implemented. The experiments in these tables comprise a refinement of the branching strategy which is basically a combination of HGB at the root, HGBSMPL-2 up to and including level L (depending on problem size) and M-R-PLUS-2 beyond level L. From the tables L=1 for size 18, L=2 for size 20 and L=3 for size 20-24. Later experiments indicate that L=5 worked well for the Nugent 25 and L= 3 works better than L=5 for the Krarup 30, both of which were solved using these parameter settings. Early experimentation on the Nugent 30 indicates that L=5 will be a good setting.

These results were used to solve two heretofore-unsolvable problem instances, the Nugent 25 and the Krarup 30. During the running of the Krarup 30a instance, it was noted that the runtimes for each of the major branches were not consistent with our expectation that runtime would be predicted by enumeration guidance bounds. That is, higher bounds along that branch would result in shorter runtime and lower bounds would produce longer runtime. Figure 4 plots the major branch runtimes versus the value of the bounds for taking those branches. While there is some adherence to the expectation that runtime would increase with decreasing bound value, there is sufficient departure from this expectation that one would conclude that better bounds are needed for tree elaboration guidance purposes near the root if we are to solve larger problems in reasonable time.

Table 6 summarizes the progress made through our branching strategy efforts and contains the runtimes and number of nodes evaluated for the Hadley 18, the Nugent 20, 22, 24 and 25 and the Krarup 30a instances. Runtimes in the Table are normalized to the speed of an UltraSPARC 360 MHz processor.

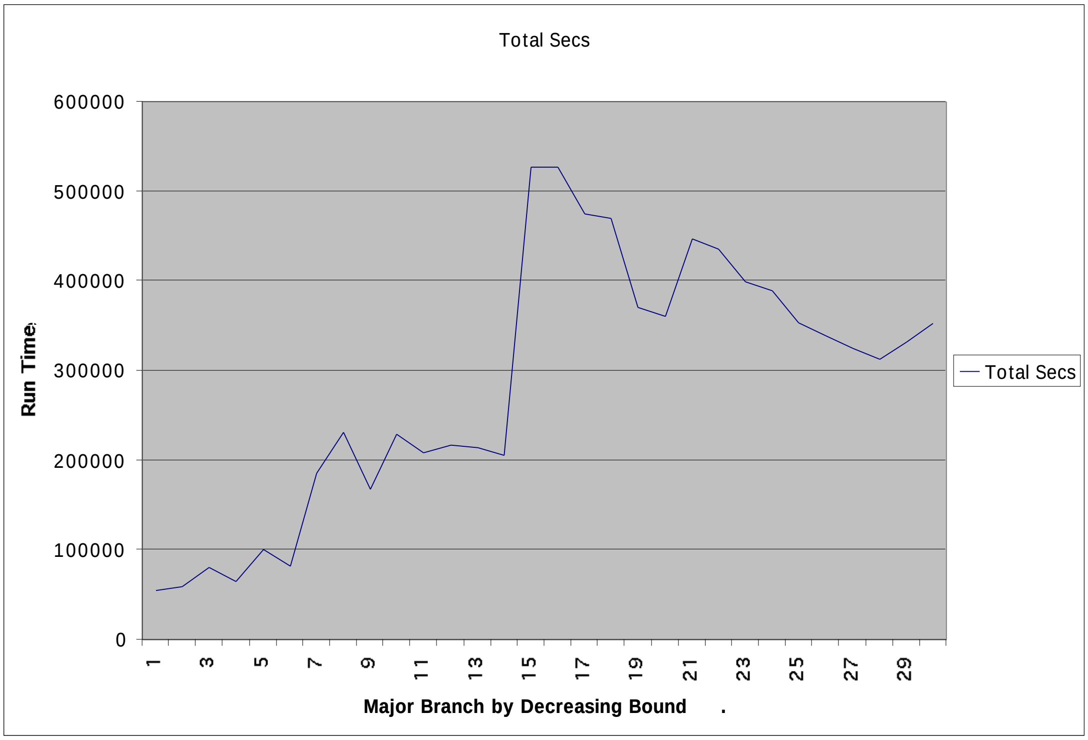

Figure 4. Krarup 30a first level branch runtime versus decreasing bound

Table 6. Tree Elaboration Experimental Results				
QAP Instance	Facility (location) choice by numerical order		Facility (location) choice by highest average bound	
	Number of Nodes	Runtime in minutes	Number of Nodes	Runtime in minutes
Hadley 18	197,487	140.3	53,224	18.2
Nugent 20	724,289	550.8	239,449	268.1
Nugent 22	10,768,366	20,545.4	988,302	3,326.8
Nugent 24	49,542,338	55,101.4	11,674,955	12,875.4
Nugent 25	N/A	N/A	108,738,131	94,980.3
Krarup 30a	N/A	N/A	29,764,589	141,958.4* (98.6 days)

N/A = Not Available

* Latest version of program (~ 1.5 to 4 times faster than previous versions).

VIII. Conclusions

Our work on tree elaboration (branching) strategies for the Quadratic Assignment Problem has extended the concept of 'forbidden' locations as originally discussed by Lawler, et al, [29] in connection with the travelling salesman problem and later implemented by Mautor and Roucairol [31] for the QAP. In 'forbidden' locations, the leader value of the next possible branching decision is used as a bound estimate for deciding whether or not to take that branch. We have developed for purposes of tree elaboration, a series of bound estimates ranging from rather poor estimates that are computationally trivial to much, much better bound estimates that are computationally expensive. We find that the better estimates are required for speeding up the enumeration of larger QAP instances ($N>16$) while poorer estimates are good only for smaller problem instances ($N\leq 16$).

Two issues are important in this development. The first is the rule by which bound estimates influence branching. The second is the choice of bound utilized for the tree elaboration decision with respect to the level in the tree.

Regarding the first issue, we explain that for the QAP, it is necessary to select at each level of the tree a facility or location upon which to extend the elaboration. The bounds associated with all possible (as yet unassigned) facility/location pairs come into play here. If these bounds are arranged in a square matrix, where the rows represent facilities and the columns represent locations, the idea is to select the row or column where bounds are generally large. Two rules were considered. The first is to take the number of bounds in the row or column that exceed the median value of all N-square bounds. The second is the sum (or more generally the average) of bounds on each row and each column.

Our experimental results indicate that the most effective rule is to choose the row or column whose average bound value is the greatest. By effective, we mean that both the runtime and the number of nodes evaluated are minimized. An average is preferable to a sum since the some of the locations along a row or facilities along a column may have previously been disallowed or excluded. Such exclusions happen, for example, in the case of Nugent problem instances with symmetries that can be exploited. When there are no exclusions, the sum and the average are the same.

Regarding the second issue, we determined that for large problem instances our most powerful bounding technique, the HGB or a variation on the HGB, must be utilized for guiding tree elaboration purposes near the root. A newly developed bound, designated M-R-PLUS-2, that comprises some of the operations in the HGB is useful for smaller problem instances and for tree elaboration farther from the root in the larger problems.

We have, therefore, designed and implemented an increasingly improved set of bound calculations. The better of these bound calculations to be utilized near the root and the less accurate (poorer bounds) utilized further into the tree. The result is an effective and powerful technique for shortening the run times of problem instances in the range of size 16 to 25. Run times were decreased generally by five- or six-to-one and the number of nodes evaluated was decreased as much as 10-to-one. Later improvements in our strategy produced a better than 3-to-1 reduction in runtime so that overall improvement was as high as 20-to-1 as compared to our earlier results [20].

What remains eminently clear from our recent results is that the QAP continues to be very difficult to solve exactly. This is especially true in the case of the largest of the Nugent instances, i.e., the Nugent 30. Tests we have made on the Nugent 30 indicate that with our current algorithm it would take between 2 and 7 years to solve exactly on a single UltraSPARC 360 MHz CPU.

While the Nugent instances have symmetries that are unlikely to exist in real-life problems and are especially difficult to solve, they are nevertheless good measures of algorithm performance. The discovery of newer and better lower bounds would certainly help. However, for a bound to be effective in a branch-and-bound context would require that it also give the additional advantage of reformulation that results in objective function cost reductions throughout the tree, as does the HGB.

As we observed in the solution of the Krarup 30a, the HGB (as currently applied) does not provide sufficient tree elaboration guidance to assure that the branching decisions near the root are the best available. Thus, it would certainly pay to consider the use of other tighter bounds that are not computationally exorbitant for this purpose. One such possibility is the new bound of Anstreicher and Brixius as described in their as yet unpublished paper, "A New Bound for the Quadratic Assignment Problem Based on Convex Quadratic Programming", dated May 24 1999. Their bound dominates the well-known projected eigenvalue bound and appears to be competitive with existing bounds in the tradeoff between bound quality and computational effort.

References

[1] Adams, W.P and Johnson, T, “Improved Linear Programming-Based Lower Bounds for the Quadratic Assignment Problem”, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 16 (1994): 43-76.

[2] Adams, W.P and Sherali, H.D, “A Tight Linearization and an Algorithm for Zero-One Quadratic Programming Problems”, *Management Science*, vol. 32, No. 10 (1986): 1274-1290.

[3] Assad, A.A, and Xu, W, “On Lower Bounds for a Class of Quadratic 0,1 Programs,” *Operations Research Letters*, vol. 4 (1985): 175-180.

[4] Bazaraa, M.S, and Kirca, O, “A Branch-and-Bound-Based Heuristic for Solving the Quadratic Assignment Problem”, *Naval Research Quarterly,* Vol. 30, (1983): 287-304.

[5] Burkard, R.E, “Locations with Spatial Interactions: The Quadratic Assignment Problem,” in *Discrete Location Theory*, P.B. Mirchandani and R.L. Francis, eds., John Wiley & Sons (1991): 387-437.

[6] Burkard, R.E, Karisch, S.E. and Rendl, F., “QAPLIB - A Quadratic Assignment Problem Library,” *European Journal of Operational Research,* vol. 55, No. 99 (1991): 115-119.

[7] Carraresi, P, and Malucelli, F, “A New Lower Bound for the Quadratic Assignment Problem,” *Operations Research*, vol. 40 (1992) Supple. No. 1: S22-S27.

[8] Chakrapani, J. and J. Skorin-Kapov, “Massively parallel tabu search for the quadratic assignment problem", Annals of Operations Research. 41, 1993, 327-341.

[9] Clausen, J., and Perregaard, M. "Solving Large Quadratic Assignment Problems in Parallel", June 30, 1995, accepted for publication in *Computational Optimization and Applications*, also DIKU Report 1994/24, 19 pages.

[10] Clausen, J., and Perregaard, M., "On the Best Search Strategy in Parallel Branch-and-Bound – Best-First-Search vs. Lazy Depth-First-Search", Baltzer Journals, October 21, 1996

[11] Clausen, J, T. Espersen, S.E. Karisch, M. Perregaard, N. Sensen and S. Tschöke, "Benchmark testing for quadratic assignment problems on a portable parallel branch-and-bound library," Work in progress, 1996.

[12] Finke, G., Burkard, R.E. and Rendl, F., "Quadratic Assignment Problems", *Annals of Discrete Mathematics*, Vol. 31 (1987): 61-82.

[13] Frieze, A.M., and Yadegar, J.: "On the Quadratic Assignment Problem", *Discrete Applied Mathematics*", vol. 5, no. 1 (Jan. 1983): 89-98.

[14] Gilmore, P.C.: "Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem", *Journal of the Society of Industrial and Applied Mathematics*, vol. 10, no. 2 (1962): 305-313.

[15] Grant, T.L., "An Evaluation and Analysis of the Resolvent Sequence Method for Solving the Quadratic Assignment Problem", Master's Thesis, University of Pennsylvania, 1989.

[16] Hadley, S., Rendl, F., and Wolkowicz, H., "Bounds for the Quadratic Assignment Problem Using Continuous Optimization Techniques", in *Integer Programming and Combinatorial Optimization*, University of Waterloo Press (1990): 237-248.

[17] Hahn, P.M., "Minimization of Cost in Assignment of Codes to Data Transmission", Ph.D. Dissertation, University of Pennsylvania, 1968.

[18] Hahn, P.M., "Practical Aspects of Calculation of Dual Procedure Bounds for the Quadratic Assignment Problem," (invited) *CO98 International Symposium on Combinatorial Optimization*, Brussels, April 15 1998.

[19] Hahn, P.M., and Grant, T.L, "Lower Bounds for the Quadratic Assignment Problem Based Upon a Dual Formulation", *Operations Research*, Vol. 46, No. 6, Nov-Dec 1998.

[20] Hahn, P.M., Grant, T.L. and Hall, N., "A Branch-and-Bound Algorithm for the Quadratic Assignment Problem Based on the Hungarian Method," *European Journal of Operational Research*, August, 1998.

[21] Hahn, P.M., Hightower, W. and Johnson, T., "Branching Strategies for a Quadratic Assignment Problem Branch and Bound Algorithm," (invited) *INFORMS'98*, Montreal, April 29, 1998.

[22] Hahn, P.M., Hightower, W. and Johnson, T., "Tree Elaboration Strategies In Branch and Bound Algorithms For Solving the Quadratic Assignment Problem," *SIAM*, Atlanta, GA, May 12-15, 1999.

[23] Hahn, P.M., Mahajan, S. and Ornatsky, A., "Theoretical Limitations to the Dual Procedure for Calculating Lower Bounds on the Quadratic Assignment Problem," *INFORMS*, San Diego, May 1997.

[24] Johnson, T.A, "New Linear Programming-Based Solution Procedures for the Quadratic Assignment Problem", *Ph.D. Dissertation,* Clemson University, Clemson, SC, (1992).

[25] Kaku, B.K, and Thompson, G.L, “An exact algorithm for the general assignment problem”, *European Journal of Operational Research,* Vol. 23, (1986): 382-390.

[26] Karisch, S.E, Cela, E, Clausen, J, and Espersen, T, “A dual framework for lower bounds of the quadratic assignment problem based on linearization”, SFB Report 120, Department of Mathematics, Technical University, Graz and as Technical Report IMM-REP-1998-02, Department of Mathematical Modeling, Technical University of Denmark.

[27] Karisch, S.E and Rendl, F, “Lower Bounds for the Quadratic Assignment Problem via Triangle Decomposition, *Mathematical Programming,* Vol. 71, No. 2 (1995): 137-152.

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

[29] Lawler, E.L., Lenstra, J., Rinnoy Kan, A., and Shmoys, D., *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley. New York, 1985.

[30] Li, Y, Pardalos, P, Ramakrishnan, K.G, and Resende, M.G, “Lower Bounds for the Quadratic Assignment Problem”, *Annals of Operations Research*, vol. 50, (1994): 387j-411.

[31] Mautor, T. and Roucairol, C., “A New Exact Algorithm for the solution of Quadratic Assignment Problems”, *Discrete Applied Mathematics,* Vol. 55, (1994): 281-293

[32] Munkres, M., "Algorithms for the Assignment and Transportation Problems”, *Journal of the Society of Industrial and Applied Mathematics*, vol. 5, No. 1 (March 1957): 32-38.

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

[34] Pardalos, P.M., Crouse, J.V., "A Parallel Algorithm for the Quadratic Assignment Problem”, *Proceedings of the 1989 Supercomputing Conference* , ACM Press, (1989): 351-360.

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

[36] Rendl, F. and Wolkowicz, H, “Applications of Parametric Programming and Eigenvalue Maximization to the Quadratic Assignment Problem”, *Mathematical Programming,* vol. 53, (1992):63-78.

[37] Resende, M.G.C, Ramakrishnan, K.G and Drezner, Z, “Computing Lower Bounds for the Quadratic Assignment Problem with an Interior Point Algorithm for Linear Programming”, *Operations Research,* Vol. 43 (1995): 781-79