

## Lower Bounds for the Quadratic Assignment Problem Based Upon a Dual Formulation

by Peter Hahn, Sci-Tech Services, Inc., 1416 Park Road, Elverson, PA 19520

and Thomas Grant, Bear Stearns and Co., 245 Park Ave., NY, NY 10167

### **Abstract**

A new bounding procedure for the Quadratic Assignment Problem (QAP) is described which extends the Hungarian method for the Linear Assignment Problem (LAP) to QAPs, operating on the four dimensional cost array of the QAP objective function. The QAP is iteratively transformed in a series of equivalent QAPs leading to an increasing sequence of lower bounds for the original problem. To this end, two classes of operations which transform the four dimensional cost array are defined. These have the property that the values of the transformed objective function  $Z'$  are the corresponding values of the "old" objective function  $Z$ , shifted by some amount  $C$ . In the case that all entries of the transformed cost array are non-negative, then  $C$  is a lower bound for the initial QAP. If, moreover, there exists a feasible solution  $U$  to the QAP, such that its value in the transformed problem is zero, then  $C$  is the optimal value of  $Z$  and  $U$  is an optimal solution for the original QAP. The transformations are iteratively applied until no significant increase in constant  $C$  as above is found, resulting in the so called Dual Procedure (DP).

Several strategies are listed for appropriately determining  $C$ , or equivalently, transforming the cost array. The goal is the modification of the elements in the cost array so as to obtain new equivalent problems which bring the QAP closer to solution. In some cases the QAP is actually solved, though solution is not guaranteed. The close relationship between the DP and the Linear Programming formulation of Adams and Johnson is presented. The DP attempts to solve Adams and Johnson's CLP, a continuous relaxation of a linearization of the QAP. This explains why the DP produces bounds close to the optimum values for CLP calculated by Johnson in her dissertation and by Resende, et al in their Interior Point Algorithm for Linear Programming.

The benefit of using DP within a branch-and-bound algorithm is described. Then, two versions of DP are tested on the Nugent test instances from size 5 to size 30, as well as several other test instances from QAPLIB. These compare favorably with earlier bounding methods.

## I. Introduction

The Quadratic Assignment Problem (QAP) is a discrete optimization problem which can be found in many fields of study, including economics, operations research, and engineering. In its basic interpretation, the problem seeks to assign  $N$  indivisible entities (or facilities) to  $N$  mutually exclusive locations, while minimizing a total quadratic interaction cost.

The QAP belongs to the class of NP-complete problems and is considered one of the most difficult. Exact solution strategies for the QAP have been largely unsuccessful for any but small problems (approximately  $N \leq 20$ ). As a result, a significant amount of efforts has been put forth by researchers in developing "inexact" or "heuristic" methods, which obtain good suboptimal solutions in reasonable CPU time (see Bazaraa and Kirca [4], Burkard and Bonniger [6] and Li et al.[25]).

The traditional algorithm used for solving QAPs exactly is branch-and bound. Until now the best bound functions for the QAP have been the 30 year old Gilmore-Lawler bound. Clausen and Perregaard [10] have chosen to use an improved version of those bounds in their groundbreaking solution of the Nugent 20 problem.

This paper describes a procedure for generating superior bounds for the QAP, which is based upon a mathematical dual of the problem and which has roots in the Hungarian algorithm (see Munkres) [27] for solving the linear assignment problem (LAP). This method was first discovered by one of us (Hahn) [17] in an attempt to solve the QAP via a resolvent sequence approach originally formulated by House et al.[19]. Some steps of the dual procedure (DP) resemble earlier lower bounding techniques, as described in: Assad and Xu [3], Burkard [5], Carraresi and Malucelli [9], Gilmore [14], Lawler [24], and Roucairol [30].

However, a unique and very important aspect of the DP is that at each stage, the QAP is restructured as fully equivalent to the original QAP in a manner that brings it closer to solution. In some cases the QAP is actually solved exactly. Unfortunately, solution is not guaranteed, as we explain in Section IV. At each stage of the DP, a high quality lower bound is economically calculated which is ideally suited to a branch-and-bound solution of the QAP.

As a result of a number of recently developed improvements, the DP produces lower bounds that are better than those of any methods of which we are aware, with the exception of methods based on eigenvalues of the flow and distance matrices or those which take advantage of symmetries in the flow or distance matrices. However, the DP handles problems more general than those of Koopmans-Beckmann [23] QAPs.

The measure of success of any efficient bounding technique is how it performs in a branch-and-bound algorithm. Such an algorithm is described in detail in a companion paper by Hahn, et al. [18]. This new algorithm is now the fastest known for solving the QAP exactly, which is demonstrated by solving in significantly less time the largest QAPLIB test instances that have been solved by others. With additional improvements, the DP can help to solve even larger problems and has the potential for being parallelized efficiently, thus shortening solution time significantly.

## II. QAP Formulation

The quadratic assignment problem (QAP) is defined as follows: Given  $N^4$  cost coefficients  $C_{ijkn} \geq 0$  ( $i, j, k, n = 1, 2, \dots, N$ ) determine an  $N \times N$  solution matrix

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

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

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

subject to the following constraints on  $\mathbf{U}$ :

$$u_{ij} = 0, 1 \quad (i, j=1, 2, \dots, N), \quad (3)$$

$$\sum_{i=1}^N u_{ij} = 1 \quad (j=1, 2, \dots, N), \quad (4)$$

and

$$\sum_{j=1}^N u_{ij} = 1 \quad (i=1, 2, \dots, N) \quad (5)$$

i.e.,  $\mathbf{U}$  is a permutation matrix.

Lawler [24] introduced the concept of an  $N^2$  by  $N^2$  solution (or assignment) matrix  $\mathbf{V}$  which is a Kronecker second power of the  $N \times N$  assignment matrix  $\mathbf{U}$ .

That is,

$$\mathbf{V} = \mathbf{U} \otimes \mathbf{U} = \begin{matrix} u_{11}\mathbf{U} & u_{12}\mathbf{U} & \dots & u_{1N}\mathbf{U} \\ u_{21}\mathbf{U} & u_{22}\mathbf{U} & \dots & u_{2N}\mathbf{U} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1}\mathbf{U} & u_{N2}\mathbf{U} & \dots & u_{NN}\mathbf{U} \end{matrix} = [v_{ijkn}] \quad (6a)$$

$$\text{where } v_{ijkn} = u_{ij} u_{kn} = u_{kn} u_{ij} = v_{knij} \quad (i = k \text{ and } j = n) \quad (6b)$$

$$v_{ijij} = u_{ij} \quad (6c)$$

$$v_{ibkb} = 0 \quad (i \neq k) \quad (6d)$$

$$\text{and } v_{bjbn} = 0 \quad (j \neq n) \quad (6-e)$$

The  $\mathbf{V}$  matrix is a composite matrix whose elements are comprised of the Null matrix (all elements zero) and matrix  $\mathbf{U}$ . Furthermore,  $\mathbf{V}$  exhibits a gross pattern identical to that of matrix  $\mathbf{U}$ .

**Example 1:** A  $\mathbf{V}$  matrix for  $N = 3$  and its corresponding  $\mathbf{U}$  matrix are shown in Figure 1.

$$\mathbf{V} = \begin{array}{ccc|ccc|ccc} \{0\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{1\} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \{1\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{0\} & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \{0\} & 0 & 0 & 0 & \{1\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 \end{array} \quad \mathbf{U} = \begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}$$

Figure 1. A  $\mathbf{V}$  Matrix for  $N = 3$  and its  $\mathbf{U}$  matrix

Equation 6b is very important to the effectiveness of the DP. It says that if an element  $v_{ijkn}$  ( $i = k$  and  $j = n$ ) is part of a solution (i.e., equal to 1) then it has a “complementary element”  $v_{knij}$  that is also in that solution. These complementary pairs have an interesting property; they are always in two different submatrices that never occupy the same submatrix row or submatrix column. Later,

we shall see that the equality between the pair creates a valuable communication between submatrices that can be exploited in improving bounds of Gilmore, Lawler and others.

The elements defined by Equations (6d) and (6e) are always zero. These elements are referred to as "disallowed" elements. Elements defined by Equation (6c) are termed "leaders" and are shown bracketed in the Figure 1. A leader has no complementary element. Observe that if there are any 1's in a submatrix  $u_{ij}U$  its leader is always unity and vice versa. Observe, too, that one and only one unity element may exist in each row and column of  $V$ . These constraints follow directly from the definition of  $V$  as the Kroenecker product of two permutation matrices.

Now, suppose we arrange the  $N^4$  cost coefficients  $C_{ijkl}$  ( $i,j,k,n=1,2,\dots,N$ ) in an  $N^2 \times N^2$  matrix  $C$ , similar to matrix  $V$  and indexed in precisely the same fashion. This is demonstrated in Figure 2, where the asterisks denote the disallowed elements, i.e., elements whose cost cannot be included in any feasible solution. The elements of matrix  $C$  which contribute to the cost  $R(U)$  of an assignment  $U$  are those and only those corresponding to the unity elements in matrix  $V$ . An assignment is said to "involve" a submatrix of  $C$ ,  $C_{ij}$ , if there are 1's in the elements of  $V$  that correspond to that submatrix. A submatrix  $C_{ab}$  of  $C$  is an  $N \times N$  matrix containing those elements of  $C$  whose first subscript is equal to  $a$  and whose second subscript is equal to  $b$ . Submatrices are clearly shown in Figure 2, separated from each other by dashed lines.

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

Figure 2. Matrix of Cost for N = 3.

The existence of complementary pairs opens the door to considerable flexibility in the  $\mathbf{C}$  matrix. If one element is part of a feasible solution, so is the other. Thus, for each complementary pair, cost can be shifted at will between the two elements. For instance, the cost of both elements can be added and placed at the first element while the cost of the second becomes zero or vice versa. The advantages of shifting cost from one element to another are discussed in Section III.

It can be shown that certain operations may be performed on  $\mathbf{C} = [C_{ijkn}]$  which will change the cost  $R(\mathbf{U})$  of assignments  $\mathbf{U}$  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:

Class 1: Addition (or subtraction) of a constant to all allowed 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 allowed elements of any row or column in matrix  $\mathbf{C}$ .

Class 1 operations maintain the cost of all assignments, but permit redistribution of element costs within a given submatrix. This follows because any submatrix involved in an assignment must itself have the form of an assignment matrix. Subsequently, the transfer of cost from one submatrix row or column to another will maintain the cost of all assignments. Furthermore, since the leader of an involved submatrix must also be involved in the assignment, the transfer of cost from a submatrix row or column to the leader, and vice versa, will also leave the cost of all assignments unchanged.

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. This is true because one and only one cost element in a row or column of  $\mathbf{C}$  can be included in the overall cost of an assignment.

### III. A Dual-Based Bounding Procedure for the QAP

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

$$R' \leq \min_{\mathbf{U}} R(\mathbf{U}) \quad (7)$$

If furthermore  $R'$  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 QAP can be stated: Maximize the sum of downward cost shifts  $R'$  permitted by Class 1 and 2 operations, under the constraint that no cost element in  $\mathbf{C}$  is driven negative.

In developing the DP, Hahn [17] recognized that this approach had roots in the Hungarian algorithm for solving linear assignment problems (LAPs) (see Munkres) [27]. The Hungarian algorithm can be viewed as a matrix reduction scheme which extracts cost from a linear cost matrix until the resulting pattern of zeros corresponds to a solution to the LAP. When this occurs, the reduction constant yields the optimum linear assignment cost. 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 DP, applying the algorithm in a manner which extended its utility and made use of the underlying interaction between elements. The motivation is the same as it is for the LAP, i.e., to extract costs from the cost matrix  $\mathbf{C}$  until the resulting pattern of zeros corresponds to a solution pattern to the QAP. (A solution pattern to the QAP is an  $N^2 \times N^2$  matrix  $\mathbf{V}$  corresponding to some  $N \times N$  permutation matrix  $\mathbf{U}$ .) This pattern of zeros is then the optimum solution. While some consideration was given to subtractions from rows and columns of  $\mathbf{C}$  directly, it was found that much more progress toward the solution is made when subtractions are made from within submatrices and added to leaders and then from leader rows and columns to  $R'$ . The DP approach is summarized as follows:

1) The Hungarian algorithm is first applied to the  $N^2$  submatrices  $\{\mathbf{C}_{ij}\}$  of  $\mathbf{C}$ . That is, for each submatrix  $\mathbf{C}_{ij}$  ( $i, j = 1, 2, \dots, N$ ), the following  $(N-1 \times N-1)$  LAP is solved:

$$\text{Minimize } \sum_{k,n} C_{ijkn} x_{kn} \quad (k = i \text{ and } n = j),$$

subject to the constraint that  $\mathbf{X} = [x_{kn}]$  is a permutation matrix. As each submatrix is about to be solved by the Hungarian algorithm, the complementary costs are collected in its elements. The resulting solution cost for each LAP is then added to the corresponding leader cost. The remaining cost elements in the matrix are left as they are found at the end of applying the Hungarian algorithm. Consequently, the costs of all assignments which involve the submatrix are maintained. (Each revised leader cost represents a lower bound on the cost of involving its associated submatrix in an assignment.) The prior collecting of complementary costs in a single submatrix results in a greater transfer of costs to leaders than would otherwise occur.

2) The Hungarian algorithm is next applied to the revised leader costs, which themselves can be arranged as an  $N \times N$  matrix. That is, the following LAP is solved:

$$\text{Minimize } \sum_{i,j} C_{ijj} x_{ij} ,$$

subject to the constraint that  $\mathbf{X} = [x_{ij}]$  is a permutation matrix. The solution cost for this leader LAP is accumulated in a reduction constant,  $R'$ , which acts as a "superleader" and has an initial value of zero. When this process is completed, all assignment costs have been reduced by the amount  $R'$ , and three possibilities exist: 1) the zeros in the reduced cost matrix  $C'$  satisfy a QAP assignment pattern (as is the case for the '1's in Figure 1), in which case  $R'$  is the cost of the optimum assignment and the problem is solved; or 2) the QAP pattern is not satisfied by the zeros, indicating that a solution is not available. However, on this iteration the constant  $R'$  has experienced an increase. In this case, the procedure continues to step 3; or 3) the QAP pattern is not satisfied by the zeros but  $R'$  remains unchanged, in which case the procedure terminates with no solution.  $R'$  then represents a lower bound. Testing whether or not the zeros in the reduced cost matrix satisfy a QAP assignment is simple. One looks at the solution zeros from the leader LAP and sees if the corresponding elements in those submatrices are also zero. If they are, the problem is solved, since the cost of the assignment is minimized.

3) The reduced leader matrix is investigated for non-zero (positive) elements. If no such elements are found, the procedure terminates with a lower bound,  $R'$ , albeit no solution. If one or more non-zero leaders are found, as there usually are, each is distributed among the elements of its submatrix (on a row or column basis), and the procedure continues to step 4. The most effective distribution of a leader element is uniform (i.e.,  $1/(N-1)$ th of the leader is distributed to each submatrix row). The motivation here is that by replacing costs into submatrices, there is a new opportunity to make those leaders which had low or zero values after step 2 larger, permitting even more cost to be eventually moved to  $R'$ . While one might expect that giving leader costs back to the submatrices would not produce a useful result, this is not the case. By the shifting of costs between complementary elements during Step 1, there is sufficient change in the entire cost distribution throughout  $C$  before the Hungarian algorithm is reapplied.

4) The Hungarian algorithm is reapplied to each submatrix, beginning with those whose leaders were zero prior to step 3. Again, as each submatrix is about to be solved by the Hungarian algorithm, the complementary costs are collected in its elements. By first processing those submatrices whose leaders had been zero prior to Step 3, greater opportunity is given for those leaders to no longer be zero, thereby increasing the amount that is moved to  $R'$ . The procedure then recycles from step 2 until a termination criterion is met. These steps are further clarified by the following example, in which the DP terminates with an optimal solution for the input QAP:

**Example 2:** A matrix  $C$  of costs for  $N=4$  is shown in Figure 3. In this representation, an asterisk is shown in the elements corresponding to disallowed indices. As a first step, all complementary element pairs are added to each other and their sums entered in the uppermost indexed position for each pair. The results of this step are shown in Figure 4. The costs in the lowermost indexed position for each pair necessarily becomes zero

In Figure 5: Transfers are made from the elements of all top row submatrices to their leaders in accordance with step 1 of the algorithm (i.e., the LAPs for those top row submatrices are solved and the solution values are added to the respective leaders). Prior to this step, no shifting of costs between complementary pairs of elements was required. However, in preparation for

solving the LAPs in the second row of submatrices, costs were moved from row 2 of each top row submatrix to their complementary positions in the second row of submatrices. The completion of step 1 of the algorithm is shown in Figure 6.

In Figure 7: Transfers are made from leader elements to the superleader (written in the upper left hand corner), in accordance with step 2.

In Figure 8 Step 3 is applied. The value of each leader is divided approximately in thirds and each portion is added to a different row of the submatrix. Since complementary cost pairs are collected in their upper locations, each resulting non-leader element reflects the contributions of two separate leaders.

The Hungarian algorithm is then applied to those submatrices whose leaders were zero as a result of step 2; this is the first part of step 4. The Hungarian algorithm is now applied to the remaining submatrices; step 4 is thus completed. This is shown in Figure 9

A second round takes place, repeating steps 2 through 4. The results are shown in Figure 10. On the application of step 2 during the third round of the algorithm, the algorithm terminates. The result is shown in Figure 11.

Superleader = 0

105	60 120 30	108 54 24	30 36	48 48 0 90	105
	70 140 35	126 63 28	35 42	56 56 0 105	
	20 40 10	36 18 8	10 12	16 16 0 30	
	60 120 30	108 54 24	30 36	48 48 0 90	
90	50 100 25	90 45 20	25 30	40 40 0 75	90
	60 120 30	108 54 24	30 36	48 48 0 90	
$C =$	70 140 35	126 63 28	35 42	56 56 0 105	$Leaders =$
	50 100 25	90 45 20	25 30	40 40 0 75	105 105 105 105
	105	105	105	105	90 90 90 90
	10 20 5	18 9 4	5 6	8 8 0 15	105 105 105 105
	20 40 10	36 18 8	10 12	16 16 0 30	90 90 90 90
	60 120 30	108 54 24	30 36	48 48 0 90	
	10 20 5	18 9 4	5 6	8 8 0 15	
90		90	90	90	

Figure 3. Cost Matrix for N = 4

Superleader = 0

105	105	105	105	
168 150 78	168 90 24	150 90 138	78 24 138	
196 175 91	196 105 28	175 105 161	91 28 161	
56 50 26	56 30 8	50 30 46	26 8 46	
<hr/>				
0 0 0	0 0 0	0 0 0	0 0 0	
90	90	90	90	
140 125 65	140 75 20	125 75 115	65 20 115	105 105 105 105
168 150 78	168 90 24	150 90 138	78 24 138	90 90 90 90
<hr/>				Leaders =
0 0 0	0 0 0	0 0 0	0 0 0	105 105 105 105
0 0 0	0 0 0	0 0 0	0 0 0	90 90 90 90
105	105	105	105	
28 25 13	28 15 4	25 15 23	13 4 23	
<hr/>				
0 0 0	0 0 0	0 0 0	0 0 0	
0 0 0	0 0 0	0 0 0	0 0 0	
0 0 0	0 0 0	0 0 0	0 0 0	
90	90	90	90	

Figure 4. Complementary Costs Added and Entered in Uppermost Position

Superleader = 0

402	279	398	257	
0 0 0	0 0 0	0 0 0	0 0 0	
27 12 0	76 11 0	18 0 8	9 0 59	
0 0 48	0 0 44	0 32 0	0 36 0	
<hr/>				
52 8 0	12 0 0	0 0 40	0 0 0	
90	90	90	90	
140 125 65	140 75 20	125 75 115	65 20 115	402 279 398 257
168 150 78	168 90 24	150 90 138	78 24 138	90 90 90 90
<hr/>				Leaders =
0 0 0	0 0 0	0 0 0	0 0 0	105 105 105 105
0 0 0	0 0 0	0 0 0	0 0 0	90 90 90 90
105	105	105	105	
28 25 13	28 15 4	25 15 23	13 4 23	
<hr/>				
0 0 0	0 0 0	0 0 0	0 0 0	
0 0 0	0 0 0	0 0 0	0 0 0	
0 0 0	0 0 0	0 0 0	0 0 0	
90	90	90	90	

Figure 5. Top Row Submatrix Elements Reduced, Second Row Prepared

Superleader = 0

402		279		398		257			
0	0	0	0	0	45	0	0	0	23 55 0
27	0	0	52	0	0	0	0	8	0 0 40
0	0	48	0	0	44	0	32	0	0 36 0
<hr/>									
0	0	0	0	0	0	0	0	0	0 0 0
316		201		295		179			
0	28	0	38	0	0	41	0	0	0 0 42
15	30	0	68	0	0	51	0	8	0 0 58
<hr/>									
0	0	0	0	0	0	0	0	0	0 0 0
0	0	0	0	0	0	0	0	0	0 0 0
151		109		140		109			
0	0	0	15	8	0	0	0	0	0 0 16
<hr/>									
0	0	0	0	0	0	0	0	0	0 0 0
0	0	0	0	0	0	0	0	0	0 0 0
0	0	0	0	0	0	0	0	0	0 0 0
108		90		104		90			

  

C =	<table border="1"> <tr> <td>402</td> <td>279</td> <td>398</td> <td>257</td> </tr> <tr> <td>316</td> <td>201</td> <td>295</td> <td>179</td> </tr> <tr> <td>151</td> <td>109</td> <td>140</td> <td>109</td> </tr> <tr> <td>108</td> <td>90</td> <td>104</td> <td>90</td> </tr> </table>	402	279	398	257	316	201	295	179	151	109	140	109	108	90	104	90	Leaders =	<table border="1"> <tr> <td>402</td> <td>279</td> <td>398</td> <td>257</td> </tr> <tr> <td>316</td> <td>201</td> <td>295</td> <td>179</td> </tr> <tr> <td>151</td> <td>109</td> <td>140</td> <td>109</td> </tr> <tr> <td>108</td> <td>90</td> <td>104</td> <td>90</td> </tr> </table>	402	279	398	257	316	201	295	179	151	109	140	109	108	90	104	90
402	279	398	257																																
316	201	295	179																																
151	109	140	109																																
108	90	104	90																																
402	279	398	257																																
316	201	295	179																																
151	109	140	109																																
108	90	104	90																																

Figure 6. Step 1 - Submatrix Elements Reduced.

Superleader = 706

88		0		88		0			
0	0	0	0	0	45	0	0	0	23 55 0
27	0	0	52	0	0	0	0	8	0 0 40
0	0	48	0	0	44	0	32	0	0 36 0
<hr/>									
0	0	0	0	0	0	0	0	0	0 0 0
80		0		63		0			
0	28	0	38	0	0	41	0	0	0 0 42
15	30	0	68	0	0	51	0	8	0 0 58
<hr/>									
0	0	0	0	0	0	0	0	0	0 0 0
0	0	0	0	0	0	0	0	0	0 0 0
7		0		0		0			22
0	0	0	15	8	0	0	0	0	0 0 16
<hr/>									
0	0	0	0	0	0	0	0	0	0 0 0
0	0	0	0	*	0	0	0	0	0 0 0
0	0	0	0	0	0	0	0	0	0 0 0
0		17		0		0			39

  

C =	<table border="1"> <tr> <td>88</td> <td>0</td> <td>88</td> <td>0</td> </tr> <tr> <td>80</td> <td>0</td> <td>63</td> <td>0</td> </tr> <tr> <td>7</td> <td>0</td> <td>0</td> <td>22</td> </tr> <tr> <td>0</td> <td>17</td> <td>0</td> <td>39</td> </tr> </table>	88	0	88	0	80	0	63	0	7	0	0	22	0	17	0	39	Leaders =	<table border="1"> <tr> <td>88</td> <td>0</td> <td>88</td> <td>0</td> </tr> <tr> <td>80</td> <td>0</td> <td>63</td> <td>0</td> </tr> <tr> <td>7</td> <td>0</td> <td>0</td> <td>22</td> </tr> <tr> <td>0</td> <td>17</td> <td>0</td> <td>39</td> </tr> </table>	88	0	88	0	80	0	63	0	7	0	0	22	0	17	0	39
88	0	88	0																																
80	0	63	0																																
7	0	0	22																																
0	17	0	39																																
88	0	88	0																																
80	0	63	0																																
7	0	0	22																																
0	17	0	39																																

Figure 7 Step 2 - Leader Elements Reduced

Superleader = 706

	0	0	0	0	
	30 51 30	27 21 45	57 30 30	50 55 21	
	56 29 37	55 0 8	32 29 45	3 0 40	
	35 29 90	0 0 57	29 67 42	0 42 0	
	0 0 0	0 0 0	0 0 0	0 0 0	
<b>C =</b>	0	0	0	0	
	27 55 34	40 0 7	64 21 28	2 0 42	0 0 0 0
	47 56 39	68 0 13	72 27 42	0 6 58	0 0 0 0
	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0 0
	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0 0
	0 0 0	0 0 0	0 0 0	0 0 0	
	7 2 15	15 8 13	0 5 13	7 12 23	
	0 0 0	0 0 0	0 0 0	0 0 0	
	0 0 0	0 0 0	0 0 0	0 0 0	
	0 0 0	0 0 0	0 0 0	0 0 0	
	0	0	0	0	

Figure 8 Step 3 - Leader Elements Redistributed

Superleader = 706

	63	29	38	21	
	0 14 0	0 0 0	13 3 0	0 0 0	
	7 0 0	29 0 0	4 0 19	0 0 12	
	0 0 9	0 0 11	0 38 0	0 36 0	
	0 0 0	0 0 0	0 0 0	0 0 0	
<b>C =</b>	83	37	73	30	
	0 2 10	12 0 0	0 0 10	0 0 0	63 29 38 21
	31 0 0	65 0 0	0 0 34	0 0 44	83 37 73 30
	0 0 0	0 0 0	0 0 0	0 0 0	28 35 28 24
	0 0 0	0 0 0	0 0 0	0 0 0	0 6 2 38
	0 0 0	0 0 0	0 0 0	0 0 0	
	28	35	28	24	
	0 0 12	0 0 0	0 20 0	0 0 0	
	0 0 0	0 0 0	0 0 0	0 0 0	
	0 0 0	0 0 0	0 0 0	0 0 0	
	0 0 0	0 0 0	0 0 0	0 0 0	
	0	6	2	38	

Figure 9 Step 4 Completed - All Submatrices Now Reduced

Superleader = 792

14										1			15			1				
0	10	14	0	0	0	13	2	0	1	0	0	0	0	0	13	2	0	1	0	0
21	0	0	30	0	0	4	0	5	0	0	11	0	0	0	4	0	5	0	0	11
0	0	22	0	0	11	0	5	0	0	16	0	0	0	11	0	5	0	0	16	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48										13			28			0				
3	5	0	0	0	0	0	4	0	1	0	0	0	0	0	0	4	0	1	0	0
38	0	0	52	0	0	8	0	46	0	0	44	0	0	0	8	0	46	0	0	44
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0										2			0			13				
0	0	12	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	6	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0										33			0			40				

  

Leaders =	14	1	15	1
	48	13	28	0
	0	2	0	13
	0	33	0	40

Figure 10 Second Round - Step 4 (Completed)

Superleader = 793

13										{0}			14			0				
0	10	14	0	0	{0}	13	2	0	1	0	0	0	0	{0}	13	2	0	1	0	0
21	0	0	30	{0}	0	4	0	5	0	0	11	0	0	0	4	0	5	0	0	11
0	0	22	{0}	0	11	0	5	0	0	16	0	0	0	11	0	5	0	0	16	0
0	0	0	0	0	0	0	0	0	0	{0}	0	0	0	{0}	0	0	0	0	{0}	0
48										13			28			{0}				
3	5	0	0	0	0	0	4	0	1	0	{0}	0	0	0	0	4	0	1	0	{0}
38	0	0	52	0	0	8	0	46	{0}	0	44	0	0	0	8	0	46	{0}	0	44
0	0	0	0	0	0	0	{0}	0	0	0	0	0	0	0	0	{0}	0	0	0	0
0	0	0	0	0	0	0	0	{0}	0	0	0	0	0	0	0	0	{0}	0	0	0
0										2			{0}			13				
0	0	12	0	0	0	{0}	0	0	0	6	0	0	0	0	{0}	0	0	0	6	0
{0}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	{0}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	{0}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{0}										33			0			40				

  

Leaders =	13	{0}	14	0
	48	13	28	{0}
	0	2	{0}	13
	{0}	33	0	40

Figure 11. Termination of Dual Procedure.

The minimum quadratic assignment cost is  $R_{\min} = 793$ . The minimum cost assignment for this is found from the zero elements that exhibit the pattern for an assignment. These elements are bracketed in Figure 11. The optimum assignment is therefore

$$\mathbf{U}_{\text{opt}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

At this point it is useful to point out the similarities of the DP with previous approaches. The Lagrangian dual ascent procedure of Assad and Xu [3] has the essential elements of the DP. It includes operations which move costs from within the submatrices of the  $\mathbf{C}$  matrix to leaders, leaving the QAP unchanged. Leader costs are transferred to a bound which is increased iteratively. The LAP is used effectively in these cost moving operations. Together, this results in bounds that are an improvement over those of Gilmore and Lawler. Our impression is that their bound is weaker because they don't take advantage of complementary costs as we do. This conclusion is based on their formulation of the QAP (page 179 of [3]) and the Lagrangean problem that motivates their bound. Their formulation does not have the  $y_{ijkl} = y_{klij}$  constraints. The constraints that they place into the objective function, i.e.,  $y_{ijkl} = N \cdot x_{kl}$ , only allows for the movement of costs between leaders and non-leaders. Recognition of a broader class of operations would have likely resulted in improved bounds.

Carraresi and Malucelli [9] base their bounds on the concept of reformulation of the QAP as an entirely equivalent problem, as is done in the DP. In equation (5) of [9] they introduce a transformation which moves costs from submatrix elements of  $\mathbf{C}$  (referred to as the quadratic costs) to leaders (the linear costs) and from leaders to the bound. This transformation is more general than the one assumed by Assad and Xu in equation (17) of [3], which explains why the bounds achieved in [9] are tighter. The Gilmore Lawler bound is a subclass of the transformations permitted and in fact is the first stage of the iterative process of generating monotonically increasing bounds. One cannot ascertain from [9] all of the operations that were permitted by (5). Thus, it is difficult to say just how closely this procedure approaches the DP.

#### IV. Relationships to Linear Programming

It turns out that the DP procedure can be viewed in the context of linear programming. In fact, the approach is a dual-ascent procedure applied to a linear formulation of the QAP. The approach implicitly uses a decomposition of the constraints in an effort to exploit the block diagonal structure. It bears resemblance to work of Johnson [20] and Adams and Johnson [1], but provides moderately stronger computational results.

In [2], Adams and Sherali devised a general strategy for linearizing 0-1 quadratic programming problems. When applied to the QAP, the following formulation emerges.

LP : Minimize :

$$\sum_{\substack{i,j,k,n \\ i < k \\ j < n}} C_{ijkn} v_{ijkn} + \sum_{i,j} C_{ijij} v_{ijij}$$

Subject to :

$$v_{ijkn} = v_{ijij} = 0,1 \quad (i,j,n), j < n \quad (8)$$

$$v_{ijkn} = v_{ijij} = 0,1 \quad (i,j,k), i < k \quad (9)$$

$$v_{knij} = v_{ijkn} \quad (i,j,k,n), k < i, j < n \quad (10)$$

$$v_{ijij} = 1 \quad (j) \quad (11)$$

$$v_{ijij} = 1 \quad (i) \quad (12)$$

$$v_{ijkn} \geq 0 \quad (i,j,k,n), k < i, j < n \quad (13)$$

In Johnson [20] and Adams and Johnson [1], this linearization (denoted Problem LP) was shown to theoretically dominate all other linear formulations of the QAP (in terms of the strength of the continuous relaxation), and to dominate most of the published bounding schemes. Included in Adams and Johnson's work was a dual-ascent strategy for obtaining a near-optimal dual solution to the continuous relaxation of LP (denoted Problem CLP), and problem lower bound. Resende et al. [29] later solved CLP to optimality using an interior point algorithm.

Our DP gives a bound very close to the CLP optimum, improving upon the computational results of Johnson [20] and Adams and Johnson [1], and requiring only a small fraction of the time of Resende et al. [29]. This latter point is illustrated in Table III below, where the DP bounds after 500, 1000 and 2000 iterations very closely approach the IPLP bounds of Resende et al.

In order to see how the DP attempts to solve CLP it is necessary to obtain a smaller reformulation via the substitution of variables suggested by constraints (10), thereby reducing the number of variables and constraints each by  $N^2(N-1)^2/2$ , in addition to halving the number of non-negativity restrictions in (13). Finally, the binary restrictions in (8) and (9) are relaxed, resulting in

CLP1: Minimize :

$$\sum_{\substack{i,j,k,n \\ i>k \\ j \leq n}} \tilde{C}_{ijkn} v_{ijkn} + \sum_{i,j} C_{ijij} v_{ijij}$$

where  $\tilde{C} = C_{ijkn} + C_{knij}$

Subject to :

$$0 \leq \sum_{\substack{k \\ k<i}} v_{ijkn} + \sum_{\substack{k \\ k>i}} v_{knij} = v_{ijij} \leq 1 \quad (i, j, n), j \leq n \quad (14)$$

$$0 \leq \sum_{\substack{n \\ n \leq j}} v_{ijkn} = v_{ijij} \leq 1 \quad (i, j, k), i > k \quad (15)$$

$$0 \leq \sum_{\substack{n \\ n \leq j}} v_{knij} = v_{ijij} \leq 1 \quad (i, j, k), i < k \quad (15a)$$

$$v_{ijij} = 1 \quad (j) \quad (16)$$

$$v_{ijij} = 1 \quad (i) \quad (17)$$

$$v_{ijkn} \geq 0 \quad (i, j, k, n), i > k, j \leq n \quad (18)$$

Now consider the following Lagrangian dual, whereby constrains (14), (15) and (15a) are placed into the objective function using multipliers a, b and b' respectively.

LD1: Maximize  $(a,b,b')$  where

$$(a,b,b') = \min_{\substack{i,j,k,n \\ i>k \\ j \leq n}} (\tilde{C}_{ijkn} - a_{ijn} - a_{knj} - b_{ijk} - b'_{kni})v_{ijkn} + \sum_{i,j} (C_{ijij} + a_{ijn} + b_{ijk} + b'_{ijk})v_{ijij} \quad (19)$$

such that  $0 \leq v_{ijij} \leq 1$  and subject to (16), (17) and (18).

The operations implied in LD1 are just the Class 1 operations defined earlier. i.e., constant amounts subtracted from (or added to) submatrix rows (or columns) are added to (or subtracted from) their corresponding leader. Now, in Steps 1 and 4 of the DP, the solution of submatrix LAPs are purely Class 1 operations. Thus, LD1 explains that part of the DP. The collecting of complimentary costs prior to solving each submatrix LAP is explained by the substitution of variables which eliminated constraints (10) in CLP. Thus, Steps 1 and 4 of the DP are fully explained by this formulation.

Since Step 3 of the DP consists only of Class 1 operations, Step 3 is also explained.

To explain Step 2, consider a further relaxation of the above Lagrangean dual, whereby constraints (16) and (17) are placed into the objective function using multipliers  $c$  and  $d$  respectively.

LD2: Maximize  $(a,b,b',c,d)$  where

$$(a,b,b',c,d) = \min_{\substack{i,j,k,n \\ i>k \\ j \leq n}} (\tilde{C}_{ijkn} - a_{ijn} - a_{knj} - b_{ijk} - b'_{kni})v_{ijkn} + \sum_{i,j} (C_{ijij} + a_{ijn} + b_{ijk} + b'_{ijk} - c_i - d_j)v_{ijij} \quad (20)$$

$$+ \sum_i c_i + \sum_j d_j$$

such that  $0 \leq v_{ijij} \leq 1$  and subject to (18).

The operations implied by LD2 constitute both Class 1 and Class 2 operations defined earlier. i.e., they include the subtraction (or addition) of constant amounts from leader rows (or columns) and the addition (or subtraction) of these amounts to (from) the DP bound. Thus, Step 2 is explained in terms of LD1, as are all four steps of the DP.

The difference between the work of Adams and Johnson and that presented here lies mainly in the fact that during Step 1 of the DP, complimentary costs are always collected in the submatrix about to be solved. This assures that there is greater transfer of costs to the leader than would otherwise be possible. Consequently, in the DP, the lower bound is greater at each iteration resulting in a faster algorithm and a higher final bound.

Adams and Johnson [1] explain that the following bounds can all be characterized in terms of progressively, more restricted subsets of the dual region encompassed by the CLP: 1. the reduction scheme of Li et al. [26]; 2. the row and column reduction strategies of Burkard and Stratmann [8], Roucairol [30] and Edwards [11]; and 3. the celebrated Gilmore-Lawler bound [14]. Thus, the linear programming solution to the CLP (and consequently the DP bounds which approach this solution very closely) must be tighter than the bounds in these three categories.

## **V. Testing and Evaluation of the Dual Procedure**

### **A. Original experiments on the DP**

In 1989 Grant [15] carried out experiments on the DP in order to evaluate the quality of the resulting lower bounds with those available from procedures used in conventional branch-and-bound algorithms. In Grant's evaluation of the DP, the algorithm was operated for 100 iterations. Two versions of the DP were tested. The first of these was the original procedure, described in Section III and referred to here as the ODP (for "original DP"). In the ODP, non-leader complementary cost pairs are combined and located in their upper-indexed submatrices first, before beginning lower bound calculations on the individual submatrices. Because of this combination, LAPs must be solved on the upper submatrices first, after which residual costs are relocated to their lower-indexed submatrices, and the next set of LAPs is solved. The potential weakness of this method is that it gives unequal weighting to the upper and lower positions for a given complementary cost pair, yielding subsequent imbalances in the leader lower bounds.

The second version of the DP tested, referred to here as the NDP (for "new DP"), utilized a different distribution scheme for complementary cost pairs. In this version, approximately half the value of each combined cost was placed in each of the two locations where the complementary pair

resided. This was expected to yield a more balanced set of submatrix lower bounds, and, in turn, possibly improve the overall lower bound calculation. The results obtained for both methods are summarized in Table I, along with experimental results published by Frieze and Yadegar [13] from their work in evaluating various lower bounding strategies.

The methods tested by Frieze and Yadegar included those of Gilmore [14], Roucairol [30], Burkard and Stratmann [8], and Edwards [11], as well as their own 'subgradient' method, based on a Lagrangian relaxation of a mixed-integer formulation for the QAP. In the Table, LB0 refers to the maximum of the 5 lower bounds of Gilmore, Roucairol, Burkard and Stratmann and Edwards. LB1 is the largest value using the subgradient method. LB2 is the largest value obtained in a simplification of that method.

As Table I reveals, both versions of the DP yielded equal or better results than those achieved by the methods of Gilmore, Roucairol, Burkard and Stratmann, or Edwards. This could

<i>TABLE I : COMPUTATIONAL RESULTS FOR THE DUAL PROCEDURE</i>							
<i>MAXIMUM LOWER BOUND VALUES</i>							
<i>PROBLEM</i>	<i>SIZE (N)</i>	<i>COST OF OPTIMAL SOLUTION</i>	<i>DUAL PROCEDURE (100 ROUNDS)</i>		<i>FRIEZE &amp; YADEGAR lb0</i>	<i>FRIEZE &amp; YADEGAR lb1</i>	<i>FRIEZE &amp; YADEGAR lb2</i>
			<i>ODP</i>	<i>NDP</i>			
Nugent, et al.	5	50	50	50	50	50	50
Nugent, et al.	6	86	83.55	83.13	82	86	82
Nugent, et al.	7	148	141.57	143.69	137	148	138
Nugent, et al.	8	214	191.27	197.24	186	194	187
Nugent, et al.	12	578	499.91	511.12	493	--	494
Lawler	7	559	559	559	505	559	511

be expected, since their approaches made no attempt to improve upon their first-cut lower bound calculation. Furthermore, the NDP results showed improvements over those for the ODP for problems of size N 7. This supports the argument for using a balanced distribution of the

combined cost of complementary pairs. With regard to Frieze and Yadegar's subgradient method, their best results exceeded all others, including those for the DP, for the Nugent, et al.[28] problems of size 7 and less. However, for the 8 and 12 facility problems of Nugent, et al., the NDP results exceeded those of Frieze and Yadegar. This suggests that the DP might yield good results when incorporated in a formal branch-and-bound algorithm.

### **B. Use of the Dual Procedure in Branch and Bound.**

Within a branch-and-bound algorithm, the DP can be utilized as the auxiliary process for computing lower bounds. In this context, the convergence limitations of the DP are not critical. Furthermore, information communicated between the main algorithm and the bounding process serve to improve the performance of both processes. When the branch-and-bound algorithm accumulates sufficient fathoming information so that permanent (or even temporary) decisions can be made on the solution matrix, these decisions are communicated to the DP and serve to improve the lower bound calculations. Such decisions involve deciding that certain elements of the solution matrix are zeros, which will be recorded in the DP cost matrix by setting the corresponding element costs to a very large value. This effectively bars the decided 0 elements from inclusion in a DP solution and focuses the reduction efforts on those elements still eligible for consideration. If instead a decide 1 element is determined, that decision is recorded in the DP cost matrix by adjusting the costs of all elements subsequently restricted from a solution (i.e., decided 0) as a consequence of this decide 1 to the large value. In either case, the communication of decisions in this manner to the DP generally permits additional cost to be extracted from the matrix, resulting in an improved lower bound and sometimes in an exact solution to the QAP. A branch-and-bound algorithm employing the DP is described in Hahn et. al. [18].

An advantage of the DP is that in a branch-and-bound algorithm it produces exact feasible solutions without having to progress through all the branches of the tree to their outermost tips. When the branch-and-bound procedure attempts to fathom branches (i.e., eliminate elements from the solution) in the submatrix that contains the optimum, it always terminates with a pattern of zeros that constitute a feasible solution. As the upper bound in the experiment is the optimum

solution, the feasible solution is indeed this optimum. The feasible solution is generated, in fact, many times during the process of fathoming within the solution submatrix, once for each element that would be part of one of the optimum solutions. This happens at various levels of branching but usually between the level at which  $N/4$  assignments have been made and the level at which  $N/3$  assignments have been made. It is hardly ever necessary to calculate bounds at a level at which  $N/2$  assignments have been made, in order to achieve a feasible solution this way.

### C. Recent improvements in the DP

While the bounds achieved by the NDP are considerably better than those found in other general bounding techniques, several improvements were made in the algorithm. One notable improvement came from the judicious return of a part of the superleader to the matrix of leaders prior to step 3, in distributing this return equally among the leader matrix elements. This was done using simulated annealing, whereby random percentages of the superleader were returned on each round; the random percentages following an exponentially decreasing annealing schedule. Some experimentation was done to optimize the selection of the exponential rate for the schedule, which affected algorithm speed as well as the bound achieved. The performance of the algorithm, however, was not very sensitive to this rate. The results of those improvements are shown in Table II for three Nugent problems and a hospital facilities problem of Elshafei [12].

Problem	BKV	DP	LB1/MCCR	EVB3	TDB	Dual Ascent
Nugent 12	578	523	493	495	n/a	513
Nugent 15	1150	1039	963	989	1083	1010
Nugent 20	2570	2179	2057	2229	2394	2145
Elshafei 19	17,212,548	16,771,926	n/a	n/a	n/a	n/a

n/a = not available

Here, we see that the lower bounds achieved by the DP exceed those of the LB1 and MCCR bounds of Li, et al.[26] as well as the Dual Ascent bounds of Adams and Johnson [1]. These are improvements on the famous Gilmore-Lawler bounds and are entirely general in their

construction. The DP bound is however exceeded by the EVB3 bound of Hadley et al. [16] for the Nugent 20 problem and the Triangle Decomposition Bound (TDB) of Karisch and Rendl [22] for all Nugent problems. It should be noted, however, that the EVB3 bounds are not appropriate for intermediate bounding during the branch-and-bound process as they depend upon a Koopmans-Beckmann formulation and the TDBs depend on the distance matrix being of a rectangular grid, which is usually not the case.

An even more interesting comparison can be made with the linear programming-based bounds of Resende, et al. [29]. Their lower bounds are much better than those of Li, et al.[26] and are not limited to Koopmans-Beckmann problems nor to rectangular grid distance matrices. Furthermore, they published run-times for their results. Table III compares the DP bounds and runtimes with those in [29].

Table III Runtime Comparison of Roughly Equivalent Bounds								
Problem	RR&D IPLP		DP 100 Iterations		DP 500 Iterations		DP 2000 Iterations	
	bound	secs	bound	secs	bound	secs	bound	secs
Nug08	203.5	9.5	201.8	0.8	203.3	4.0	203.3	14.8
Nug12	522.9	754.1	516.2	4.7	521.2	22.2	522.4	82.5
Nug15	1041.0	5203.8	1028.3	17.8	1037.6	85.3	1039.7	325.3
Nug20	2181.4	3611.4	2158.5	80.6	2174.7	382.7	2179.1	1457.1
Rou20	643346	4427.6	636036	62.7	640459	290.2	641663	1091.8
Kra30a	76002.3	24679.0	74933.6	452.6	75678.1	2120.0	75853.4	7382.8
Kra30b	76751.8	30481.8	75634.6	453.2	76374.6	2136.7	76562.2	7466.5
Nug30	4804.4	28819.6	4744.7	411.9	4779.4	1897.1	4788.9	6893.9

Run-times for the DP are listed for 100, 500 and 2000 iterations. The problems are taken from QAPLIB (see [7]). The run times for the IPLP were logged on a Silicon Graphics Challenge computer with a 150 MHz IP19 processor. The DP results were logged on a SPARC 10 workstation with a 75 MHz SuperSparc processor. For each problem in the Table, all the bounds

for a given problem size are within 1.5 percent of each other. As explained earlier, the DP at 2000 iterations produces bounds almost equal to those of the IPLP. However, the difference in actual run times is large. To demonstrate the advantage of the DP over the IPLP in a branch-and-bound algorithm, compare the IPLP time of 4427.6 seconds for the Roucairol size 20 problem with the 100 iteration DP time of 62.7 seconds. If the two machines were equivalent, the ratio of runtimes would be over 70:1. The machine differences could increase that advantage to over 100:1. As for growth in runtime with problem size, the DP appears to have a slight edge over the IPLP.

In Table IV we present a comparison of the bounds produced by the DP with bounds of additional test instances from QAPLIB. In the last five instances, the DP produced significantly

Table IV Dual Procedure (DP) Comparison with QAPLIB Bounds			
Instance	BKV	QAPLIB	DP
Had16	3,720	3,648.0	3,556.9
Had18	5,358	5,226.0	5,082.6
Had20	6,922	6,713.0	6,571.5
Nug21	2438	n/a	2018.6
Nug22	3596	n/a	2843.7
Nug24	3488	3251	2866.6
Tai17a	491,812	412,722	**441,560
Tai20a	703,482	580,674	**617,206
Tai20b	122,455,319	14,857,089	**93,563,896
Tai25a	1,667,256	962,417	**1,006,749
Tai25b	344,355,646	51,401,950	**151,115,405

n/a not available      \*\*Improved bound

better bounds than were listed in QAPLIB. The QAPLIB result for the Nugent 24 instance is a TDB bound which you may recall is dependent upon the structure of the distance matrix. The other results represent general bounding techniques and thus provide a better comparison.

## **VI. Potential for Speeding up the Dual Procedure**

In the development of the DP, a number of variations on the Class 1 and 2 operations were tried (see Section II). One variation called 'pivoting' has considerable potential. Pivoting is usually introduced after the Hungarian algorithm has been applied to a submatrix. At that point, as much as possible has been drawn into the leader and there is at least one zero in every row and column of the submatrix. A given zero element is the pivot. A column (or row) containing the pivot is scanned for its minimum value (exclusive of the pivot). That amount is subtracted from the column (or row) and added to the row (or column) containing the pivot. Since every element in a submatrix is a member of another submatrix, significant movement of costs are thus made throughout the cost matrix  $C$ . Given that during a branch-and-bound process, we come across what we suspect to be the optimum solution to the QAP, we conjecture that it is possible to develop a heuristic that allows the movement of costs in such a manner that the cost elements for that solution can be decreased while increasing other costs so that they can contribute to leaders and the superleader thus increasing the bound. The goal is to reduce the number of iterations required to reach a useful bound.

### **Acknowledgments**

The authors thank Charles Clarke of Chalfont, PA for his helpful suggestions and encouragement during the very earliest development of the Dual Procedure described herein. We also thank Terri Johnson and Warren Adams who made helpful suggestions and pointed out various errors and whose encouragement is much appreciated. Special thanks go to Monique Guignard-Spielberg for helping us make the theoretical connection between the DP and the CLP of Adams and Johnson. We are grateful to the referees whose comments were invaluable to improving the readability of this paper.

## 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 Comp. 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 Logistics Quarterly*, vol. 30, no. 1 (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 and Bonniger, T, "A Heuristic for Quadratic Boolean Programs with Applications to Quadratic Assignment Problems," *European Journal of Operational Research*, vol. 13, (1983): 374-386.
- [7] 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.
- [8] Burkard, R.E. and Stratmann, K.H, "Numerical Investigations on Quadratic Assignment Problems," *Naval Research Logistical Quarterly*, vol. 25, no. 1 (1978): 129-148.
- [9] Carraresi, P, and Malucelli, F, "A New Lower Bound for the Quadratic Assignment Problem," *Operations Research*, vol. 40 (1992) Supple. no. 1: S22-S27.

- [10] Clausen, J, and Perregaard, M, "Solving Large Quadratic Assignment Problems in Parallel," DIKU, Dept. of Computer Science, University of Copenhagen, June 30, 1995: accepted for publication in *Computational Optimization and Applications*.
- [11] Edwards, C.S, "A Branch-and-Bound Algorithm for the Koopmans-Beckmann Quadratic Assignment Problem," *Mathematical Programming Study*, vol. 13 (1980): 35-52.
- [12] Elshafei, A, "Hospital Layout as a Quadratic Assignment Problem," *Operations Research Quarterly*, vol. 28, no. 1 (1977): 167-179.
- [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 for 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, Grant, T.L. and Hall, N, " Solution of the Quadratic Assignment Problem Using the Hungarian Method," accepted for publication in the *European Journal of Operational Research*.
- [19] House, R.W, Nelson, L.D, and Rado, T, "Computer Studies of a Certain Class of Linear Integer Programs," *Recent Advances in Optimization Techniques*, edited by A. Lavi and T.P. Vogl, New York: John Wiley and Sons, 1965.

- [20] Johnson, T.A, "New Linear Programming-Based Solution Procedures for the Quadratic Assignment Problem," Ph.D. Dissertation, Clemson University, Clemson, SC, (1992).
- [21] Kaku, B.K, Thompson, G.L, "An Exact Algorithm for the General Quadratic Assignment Problem," *European Journal of Operational Research*, vol. 23, no. 3 (Mar. 1986): 382-390.
- [22] 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.
- [23] Koopmans, T.C, and Beckmann, M.J, "Assignment Problems and the Location of Economic Activities," *Econometrica*, vol. 25 (1957): 53-76.
- [24] Lawler, E.L, "The Quadratic Assignment Problem," *Management Science*, vol. 9 (1963): 586-599.
- [25] Li, Y, Pardalos, P, and Resende, M., "A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem," *DIMACS Series in Discrete Mathematics and Theoretical Comp. Science*, vol. 16 (1994): 237-263.
- [26] Li, Y, Pardalos, P, Ramakrishnam, K.G, Resende, M.G, "Lower Bounds for the Quadratic Assignment Problem," *Annals of Operations Research*, vol. 50, (1994): 387-410.
- [27] Munkres, M., "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1 (March 1957): 32-38.
- [28] Nugent, C.E, Vollman, T.E, and Ruml, J, "An Experimental Comparison of Techniques for the Assignment of Facilities to Locations," *Operations Research*, vol. 16 (Jan.-Feb. 1968): 150-173.
- [29] 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): 781-791.
- [30] Roucairol, C, "A Reduction Method for Quadratic Assignment Problems, *Operations Research Verfahren*, vol. 32 (1979): 183-187.