

A Lower Bound for the Quadratic Assignment Problem
Based on a Level-2 Reformulation-Linearization
Technique

by

Peter M. Hahn*, The University of Pennsylvania

William L. Hightower, Highpoint University

Terri Anne Johnson, Clemson University

Monique Guignard-Spielberg,
The University of Pennsylvania

Catherine Roucairol,
The University of Versailles

Statement of Scope and Purpose

This paper should be of interest to the combinatorial optimization community and especially to those interested in the Quadratic Assignment Problem (QAP). The QAP has application in the assignment of facilities to locations (to minimize the cost of intra-facility transportation), the placement of electronic components (to minimize the length of interconnecting wire), the placement of blades in a turbine (to minimize rotor imbalance), and many other analogous problems. The advances described herein are mainly computational. The authors demonstrate the efficacy of their new exact solution algorithm by timing it on series of especially difficult test cases and comparing the results with those of the two competing algorithms for this purpose. For the largest case tested, the new algorithm is an order of magnitude faster than the only other algorithm to have solved this problem. More importantly, the increase in computation time appears to be much slower with problem size than in the case of the two competing algorithms. Sufficient theory is presented to give an understanding of the underlying principles of the algorithm.

* Corresponding author.

ABSTRACT

This paper presents a new lower bound for the Quadratic Assignment Problem (QAP) based on a level-2 reformulation-linearization technique (RLT). This technique provides tight bounds that can be calculated efficiently and are suitable for use in a branch-and-bound algorithm for solving the QAP exactly. Our calculation of the bound depends on a formulation of the QAP similar to the one developed by Ramachandran and Pekny, which is based on the lifting of the problem into a higher dimensional space. However, the technique has its primary roots in the level-2 RLT of Adams and Sherali. Our implementation of the bound calculation is a generalization of the dual ascent procedure we developed for the level-1 RLT. In this paper, we compare the tightness and speed of computation of this new bound with that of competing bounds. We also demonstrate the usefulness of this bound in a branch-and-bound algorithm specifically designed to exploit the RLT properties of the calculation.

KEYWORDS

Suggested keywords to describe research content: Quadratic Assignment Problem, Lower Bounds, Reformulation-Linearization Technique (RLT), Integer Programming

Suggested keywords to describe computational methods used: Dual Ascent Method

I. Introduction

The Quadratic Assignment Problem (QAP) is among the most difficult combinatorial optimization problems today. Solving general problems of size greater than 30 (i.e., with more than 900 (0-1) variables) is still computationally impractical. Among exact algorithms, branch-and-bound are the most successful, but the lack of sharp lower bounds in these algorithms has been one of the major difficulties.

The quadratic assignment problem is NP-hard. But, this theoretical complexity is not sufficient to explain why it is so difficult, 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, is a more convincing explanation. Indeed, 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. 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. In addition, the computation of the lower bound is another major difficulty. 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. We refer you to our paper on branching strategy [12] for a discussion of the progress in QAP lower bounds from 1962 to the present.

Two recent developments have resulted in a large improvement in the ability to solve QAPs exactly. The first is the dual procedure (DP) bound of Hahn and Grant [10], which derives from a level-1 reformulation linearization technique (RLT) formulation of the QAP. Not only does the DP yield strong bounds efficiently, but it affords the removal of costs from branch sub-problems bringing them closer to dual solution and making the calculation of bounds within the tree even more efficient. The second is the quadratic programming (QP) bound of Anstreicher and Brixius [3]. It is the speed of calculation of this QP bound and its parallel implementation that makes it effective in solving the most difficult problems to date. These two methods have made it possible to solve exactly heretofore-unsolved problems of size 30 [5 and 13].

In this paper, we present a new lower bound for the Quadratic Assignment Problem (QAP) based on a level-2 reformulation-linearization technique (RLT). This new bound was originally conceived by William Hightower and was implemented and refined by the authors of this paper. Section II of this paper is devoted to a review of the QAP. For the necessary background, Section III presents the level-1 RLT formulation of the Quadratic Assignment Problem of Adams and Johnson [1]. Section IV presents the formulation of the QAP by Ramachandran and Pekny [19]. Their formulation is based on the lifting of the problem into a higher dimensional space. Our new level-2 lower bound is based on this formulation. The lifting procedure of Ramachandran and Pekny is in fact the level-2 RLT of Sherali and Adams [22-23]. Section V describes our implementation of the level-2 RLT lower bound calculation. It is a generalization of the dual ascent procedure developed by Hahn [9] for the level-1 RLT. In Section VI, we compare the tightness and speed of computation of this new bounding technique with that of competing bounds. In Section VII, we demonstrate the effectiveness of this new bound in a branch-and-bound algorithm specifically designed to exploit the reformulation properties of the calculation. We compare these results to those of Anstreicher and Brixius [4 and 5].

II. The Quadratic Assignment Problem

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

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

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

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

The permutation constraints implied in (1) are then

$$\sum_i u_{ij} = 1 \quad (3)$$

$$\sum_j u_{ij} = 1 \quad (4)$$

where $u_{ij} = 0, 1$

This formulation of the QAP is more general than that of the Koopmans-Beckmann [15] formulation of the problem, in that the cost coefficients C_{ijkn} need not be derived from a product of flows and distances. Also, non-zero linear costs C_{ijj} are permitted, which would be impossible in the Koopmans-Beckmann formulation.

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

It is easily seen that those elements of matrix \mathbf{C} , which contribute to the cost $R(\mathbf{U})$ of an assignment \mathbf{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 $R(\mathbf{U})$.

In each submatrix, the element occupying a position corresponding to the submatrix position in \mathbf{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.

It is further shown in [9] that certain operations may be performed on $\mathbf{C} = [C_{ijkn}]$ that 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 (see Grant [8] for proofs):

Class 1: Addition (or subtraction) of a constant to all feasible elements of some row or column of submatrix $[C_{ij}]$ 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 \mathbf{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 [9].

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 modified matrix \mathbf{C}' non-negative, then no assignment cost can become negative and the following relationship holds:

$$R' \leq R(\mathbf{U})_{\min} \quad (5)$$

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.

III. A level-1 RLT formulation

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:

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

Subject to:

$$v_{ijkn} = v_{ijij} \in \{0,1\} \quad (i, j, n), n < j \quad (6)$$

$$v_{ijkn} = v_{ijij} \in \{0,1\} \quad (i, j, k), k < i \quad (7)$$

$$v_{knij} = v_{ijkn} \quad (i, j, k, n), k > i, n < j \quad (8)$$

$$v_{ijij} = 1 \quad (j) \quad (9)$$

$$v_{ijij} = 1 \quad (i) \quad (10)$$

$$v_{ijkn} \geq 0 \quad (i, j, k, n), k < i, n < j \quad (11)$$

In Johnson [14] 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. The continuous relaxation of this formulation of the QAP (denoted CLP in [1]) provides the framework of a variety of lower bounding schemes including the Dual Procedure of Hahn and Grant [10]. Sherali and Adams [23] refer this formulation of the QAP as a Reformulation-Linearization Technique (RLT) of level-1. Warren Adams points out that the formulation LP can be easily derived by multiplying each QAP constraint (3) and (4) by each assignment variable u_{ij} and substituting v_{ijkn} for $u_{ij}u_{kn}$.

In order to proceed with the arguments of the next Section, it is necessary to obtain a smaller reformulation via the substitution of variables suggested by constraints (8). This reduces the number of variables and constraints each by $N^2(N-1)^2/2$ and halves the number of non-negativity restrictions in (11). Finally, the binary restrictions in (6) and (7) are relaxed, resulting in the following formulation:

CLP1: Minimize:

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

where $c_{ijkn} = C_{ijkn} + C_{knij}$

Subject to:

$$0 \leq \sum_{\substack{k \\ k > i}} v_{ijkn} + \sum_{\substack{k \\ k < i}} v_{knij} = v_{ijij} \quad 1 \quad (i, j, n), n > j \quad (12)$$

$$0 \leq \sum_{\substack{n \\ n > j}} v_{ijkn} = v_{ijij} \quad 1 \quad (i, j, k), k > i \quad (13)$$

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

$$v_{ijij} = 1 \quad (j) \quad (15)$$

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

$$v_{ijkn} \geq 0 \quad (i, j, k, n), k > i, n > j \quad (17)$$

IV. Lifting the QAP into a higher dimensional space for a Level -2 Bound

Ramachandran and Pekny [19] point out that the linear programming relaxation of an integer programming problem can be strengthened by lifting the problem into a higher dimensional space. They proceed to derive a tighter formulation of the QAP by using such a lifting procedure applied to CLP1. This is obtained by multiplying the constraints in (12), (13) and (14) by each assignment variable u_{ij} and substituting z_{ijknpq} for $u_{ij}u_{kn}u_{pq}$, and results in the following formulation:

CLP2: Minimize:

$$\sum_{\substack{i, j, k, n, p, q \\ k > i, n, j, p > k, q, n, j}} d_{ijknpq} z_{ijknpq} + \sum_{\substack{k, n, i, j \\ k > i, j, n}} c_{ijkn} v_{ijkn} + \sum_{i, j} C_{ijij} v_{ijij}$$

Subject to:

$$\sum_{\substack{p \\ p > k > i}} z_{ijknpq} + \sum_{\substack{p \\ k > p > i}} z_{ijpqkn} + \sum_{\substack{p \\ k > i > p}} z_{pqijkn} = v_{ijkn} \quad 1 \quad (i, j, k, n, q), q \neq n, j, k > i \quad (18)$$

$$\sum_{\substack{q \\ q \neq n}} z_{ijknpq} = v_{ijkn} \quad 1 \quad (i, j, k, n, p), p > k > i, n \neq j \quad (19)$$

$$\sum_{\substack{q \\ q \neq n}} z_{ijpqkn} = v_{ijkn} \quad 1 \quad (i, j, k, n, p), k > p > i, n \neq j \quad (20)$$

$$\sum_{\substack{q \\ q \neq n}} z_{pqijkn} = v_{ijkn} \quad 1 \quad (i, j, k, n, q), k > i > p, n \neq j \quad (21)$$

$$v_{ijij} = 1 \quad (j) \quad (22)$$

$$v_{ijij} = 1 \quad (i) \quad (23)$$

$$v_{ijkn} = 0 \quad (i, j, k, n), i > k, j \neq n \quad (24)$$

$$z_{ijknpq} = 0 \quad (i, j, k, n, p, q), p > k > i, q \neq n, j \quad (25)$$

They proceed further to define third order interaction cost coefficients $d_{ijknpq} = c_{ijkn} + c_{ijpq} + c_{knpq}$, which would result in the generation of a QAP whose objective function would be $(N-1)$ times that of the original QAP. This is entirely unnecessary. If

these third order (or cubic) costs are zero, the formulation is still correct and the objective function remains equal to that of the original QAP.

Since Ramachandran and Pekny's explanation of this lifting process is identical to that for Adams and Sherali's level-2 reformulation linearization technique, the two concepts are one and the same.

V. Computational Techniques

Consider now a Cubic Assignment Problem that gives rise to the continuous relaxation given by CLP2:

Given N^6 cost coefficients D_{ijknpq} ($i, j, k, n, p, q=1, 2, \dots, N$) determine an $N \times N$ solution (i.e., permutation) matrix

$$\mathbf{U} = [u_{ab}] \quad (26)$$

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

$$R(\mathbf{U}) = \sum_{ijknpq} D_{ijknpq} u_{ij} u_{kn} u_{pq} \quad (27)$$

There is no loss of generality if we later were to let the costs:

$$D_{ijijkn} + D_{ijknij} + D_{knijij} + D_{ijknkn} + D_{knijkn} + D_{knknij} \quad C_{ijkn} \quad (28)$$

since they are actually quadratic costs. In (28) the arrow implies the collection of l.h.s. costs into the r.h.s. term.

We can now we arrange the N^6 cost coefficients D_{ijknpq} in an $N^2 \times N^2 \times N^2$ matrix \mathbf{D} as shown in Figure 2. Bold variables denote matrices. Matrix \mathbf{D} consists of submatrices \mathbf{D}_{ij} which further consist of sub-submatrices \mathbf{D}_{ijkn} . We term \mathbf{D}_{ij} the parent matrix of \mathbf{D}_{ijkn} and \mathbf{D} the parent matrix of \mathbf{D}_{ij} . The asterisks in Figure 2 denote disallowed matrices or elements, i.e., matrices or elements that cannot be included in any feasible solution. Further, there are no sub-submatrices in the C_{ijkn} positions since, as we argued above, all the elements that would exist in these sub-submatrices correspond to quadratic costs and are collected according to (28).

It can be shown that those elements of matrix \mathbf{D} , which contribute to the cost $R(\mathbf{U})$ of an assignment \mathbf{U} are confined to one submatrix in each row and one submatrix in each

column. Furthermore, within those submatrices, only one sub-submatrix from each submatrix row and one sub-submatrix from each submatrix column contribute to the cost $R(\mathbf{U})$. Finally, within those sub-submatrices, only one element from each sub-submatrix row and one element from each sub-submatrix column contribute to the cost $R(\mathbf{U})$.

In each submatrix and in each sub-submatrix, the element occupying a position corresponding to the matrix position in the parent matrix 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.

Similar to the argument given in Section II, certain operations may be performed on \mathbf{D} that 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 are indeed the Class 1 and Class 2 operations referred to in that Section.

Thus, for the Cubic assignment problem, if the operations on \mathbf{D} decrease the cost of assignments by an amount R' and are performed in a way that keeps the elements in modified matrix \mathbf{D}' non-negative, R' constitutes a valid lower bound on the Cubic Assignment Problem. Here, as in the case of the QAP, if the costs are decreased such that the cost elements comprise a solution pattern, then an optimum solution is proven. A solution pattern is given by:

- A. One submatrix in each submatrix row and one in each submatrix column.
- B. Within those submatrices, one sub-submatrix within each sub-submatrix row and one in each sub-submatrix column.
- C. And, within those sub-submatrices, one element in each row and one in each column.

Optimum solutions are indeed found in the \mathbf{D} matrix. This happens in the case of small QAPs ($N < 12$) and during branch-and-bound enumeration of larger problems.

The Hightower-Hahn Bound

We adopt the Hungarian algorithm (see Munkres [17]) as the core set of operations for the Hightower-Hahn bound. We specifically make use of the underlying interaction between elements. Namely, that in the formulation CLP2,

$$v_{ijkn} = v_{knij} \quad (8)$$

$$\text{and } z_{ijknpq} = z_{ijpqkn} = z_{pqijkn} = z_{knpqij} = z_{pqknij} = z_{knijpq} \quad (29)$$

The motivation is the same as for the level-1 RLT bound of Hahn and Grant [10], i.e., to extract as much as possible from the cost matrix \mathbf{D} thereby increasing the bound $R(\mathbf{U})$.

In setting up the problem, the linear costs C_{ijj} and quadratic costs c_{ijkn} are entered in the formulation of CLP2. The cubic costs d_{ijknpq} are all set to zero.

Step 1a:

The linear costs that are non-zero are each distributed among the quadratic cost elements of its submatrix. This is done by taking $1/(N-1)$ -th of the linear cost and adding it to the first $N-2$ rows of quadratic costs of the submatrix and adding the remaining linear cost (after $(N-2)/(N-1)$ has been removed) to the last row of quadratic costs. The result is that quadratic costs have been increased in a fashion that keeps all QAP solution costs the same. This is so because a solution that passes through any submatrix had to pass through its linear cost. Since the solution must also pass through all the rows of quadratic costs of the submatrix, the linear cost is still incurred.

Step 1b:

Now, the quadratic costs that are non-zero are each distributed among the cubic cost elements of its sub-submatrix. This is done by taking $1/(N-2)$ -th of the linear cost and adding it to the first $N-3$ rows of cubic costs of the sub-submatrix and adding the remaining linear cost (after $(N-3)/(N-2)$ has been removed) to the last row of cubic costs. The result is that cubic costs have been increased in a fashion that keeps all solution costs the same. This is so because a solution that passes through any sub-submatrix had to pass through its quadratic cost. Since the solution must also pass through all the rows of cubic costs of the sub-submatrix, the quadratic cost is still incurred.

Step 2a:

The Hungarian algorithm is applied to each sub-submatrix, beginning with those whose quadratic costs were zero prior to Step 1b. As each sub-submatrix is about to be solved by the Hungarian algorithm, the complimentary costs (i.e., those corresponding to the

equated variables in (29)) are collected in its elements. The cost removed from each sub-submatrix (i.e., the solution of the Hungarian algorithm) is now added to the quadratic cost of that sub-submatrix. Again the cost of all solutions to the QAP remain the same.

Step 2b:

The Hungarian algorithm is applied to each submatrix of revised quadratic costs, beginning with those whose quadratic costs were zero prior to Step 1a. As each submatrix of quadratic costs is about to be solved by the Hungarian algorithm, the complimentary costs (i.e., those corresponding to the equated variables in (8)) are collected in its elements. The cost removed from each submatrix (i.e., the solution of the Hungarian algorithm) is now added to the linear cost of that submatrix. Again the cost of all solutions to the QAP remain the same.

Step 3:

The Hungarian algorithm is next applied to the revised linear costs, which themselves can be arranged as an $N \times N$ matrix. The solution of this Hungarian algorithm is accumulated in a reduction constant R' , which constitutes the lower bound to the QAP and has an initial value of zero. Now, the costs of all solutions to the QAP have been reduced by the amount R' . At this point, 3 possibilities exist: 1) The zeros in the reduced cost matrix \mathbf{D}' satisfy a QAP solution pattern (as described in this Section), in which case R' is the cost of the optimum solution and the problem is solved. 2) The number of iterations has exceeded a predetermined value and the value of R' is the desired lower bound. Or 3) more iterations are desired to improve the lower bound, and the algorithm goes to Step 1.

The Hightower-Hahn Branch-and-Bound Algorithm (HHB&B)

The HHB&B algorithm is developed in a manner consistent with the Hightower-Hahn bound. It follows 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 2, this would be element C_{1111} , implying facility 1 is assigned to location 1.

Based on the selection of linear cost C_{ijj} , submatrix \mathbf{D}_{ij} is involved in the assignment. The submatrices remaining in the row and the column that contain submatrix \mathbf{D}_{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 and from each sub-submatrix of the original problem, with the exception of the original quadratic costs in submatrix \mathbf{D}_{ij} , which remain $N-1$ in size. To complete the formulation of the newly formed $N-1$ problem, these costs are added (by simple matrix addition) to the now size $N-1$ matrix of linear costs.

It is the application of the Hightower-Hahn bound on the newly formed $N-1$ size problem that attempts to fathom a partial assignment postulated by the selection of linear cost C_{ijj} . 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 Hightower-Hahn bound moves costs out of the \mathbf{D} matrix into a lower bound value, leaving a modified matrix \mathbf{D}' . 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 \mathbf{D}' 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 Hightower-Hahn bound at earlier (higher) levels of partial assignment. Using the modified matrix \mathbf{D}' 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 Hightower-Hahn bound 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.

The memory requirements for storing the \mathbf{D} matrix are rather large. Even larger are the memory demands of all the tables and pointers required to deal with constraints (18) to (23) and the interactions (8 and 29) of elements. For the algorithm to work efficiently, these arrays, tables and pointers must be stored in RAM for ready access to the CPU.

Figure 3 is a graph of random access memory required for the HHB&B algorithm. If we are to solve QAPs as large as size 30, available computer memory should be around 6 GBytes.

VI. Comparison of Lower Bound Calculations

The results of 2000 iterations of the Hightower-Hahn bound on three Nugent [18] instances of the QAP are summarized in Table 1.

For comparison, we include in Table 1, the bounds of the Gilmore-Lawler [7 and 16], the level-1 RLT interior point calculations of Resende, Ramakrishnan and Drezner [21], the level-1 RLT bounds of Hahn and Grant [10] and the level-2 RLT interior point calculations of Ramakrishnan, Resende, Ramachandran and Pekny [20]. In [20], Ramakrishnan, et. al. use the commercial linear programming package CPLEX to solve the level-2 RLT formulation of the QAP. They were able to solve QAP instances of size 12 and less. Larger instances of the QAP could not be solved due to limitations of CPLEX.

It should be noted that the level-1 RLT interior point calculations of Resende, et. al. are essentially optimum for the level-1 RLT formulation of the QAP. Similarly, the level-2 RLT interior point calculations of Ramakrishnan, et. al., are essentially optimum for that formulation. The Hahn-Grant bound comes very close to the former. And, the Hightower-Hahn bound comes very close to the latter.

Ramakrishnan, et. al. report a runtime of 6504.2 seconds runtime on a 250 MHz SGI Challenge for calculating the bound on the Nugent 12 instance. If cycle time is considered the relative CPU speed determinant, this would correspond to 4517 seconds on a 360 MHz Sun Ultra 10. As a contrast, the Hightower-Hahn bound of 577.53 for the Nugent 12 was calculated on the Sun Ultra 10 in just 93.4 seconds.

Table 2 lists the Hightower-Hahn bound values and runtimes (on the Sun Ultra 10) for the Nugent 20 as a function of the number of iterations of the algorithm.

VII. Comparison of Branch-and-bound Enumeration

Table 3 compares the performance of the leading branch-and-bound QAP algorithms. These are the Anstreicher-Brixius algorithm [4 and 5], based on a Quadratic Bound, the Hahn, et. al. algorithm [12], based on a level-1 RLT bound and the Hightower-Hahn algorithm, based on the level-2 RLT algorithm.

The problem instances in Table 3 are from the Nugent set, which is considered very difficult. Of these, only the size 20 and 30 are from Nugent, et. al.'s original paper [18]. The rest were derived from that paper by later researchers. Data sets for these problems may be found on the QAPLIB web site [6].

The runtimes are given in equivalent single-processor CPU-seconds. These are normalized to minutes in the last row of the Table, based on an estimate of the ratio of speeds of the actual machines to a single CPU HPC3000 workstation.

Figure 4 depicts the trend in runtime as a function of Nugent problem size. Three branch-and-bound elaboration algorithms are plotted: the Hahn, et. al. level-1 bound algorithm, the Anstreicher-Brixius quadratic programming bound algorithm and the Hightower-Hahn level-2 bound algorithm. The plot is the logarithm (base 10) of normalized runtime (in minutes) of a single HPC3000 CPU versus Nugent problem size.

For Nugent instances 22 and above, the Hahn, et. al. and Anstreicher-Brixius logarithmic increase in runtime are linearly related to problem size. Both slopes are almost identical, i.e., approximately 0.45 per unit increase in problem size. This corresponds to approximately a 2.82-fold increase in runtime per unit size increase. The trend for the Hightower-Hahn algorithm is not linear and its slope decreases at the higher Nugent sizes. While this trend needs to be confirmed with further experimentation, it is very encouraging. The slope between the Nugent 25 and 27 instances is approximately 0.21 per unit increase in problem size. This corresponds to approximately a 1.62-fold increase in runtime per unit Nugent size increase. If this slope were to hold for larger instances, the most difficult of the Nugent problem instances, the Nugent 30 could be solved in about four months on a single HPC3000 CPU. This compares favorably to the only known solution of the Nugent 30 [5], which took the equivalent of 6.9 years on a single HPC3000 CPU.

VIII. Conclusions

We have shown that it is possible to implement a dual ascent procedure for calculating near optimum solutions to the level-2 RLT formulation of the Quadratic Assignment Problem. These calculations not only produce tight bounds, but also are efficient enough for implementation in an effective branch-and-bound scheme for exact solution of the QAP. Our experiments show promise for solving larger problem instances.

Since the level-2 RLT calculations are computationally quite expensive, it is only for the large size problems that the strength (tightness) of the new bound reveals its advantage and potential power. Only on the Nugent 27 is there a significant improvement over older, less effective methods. We very much need to experiment with the size 28 and size 30 problem to assure ourselves and the research community that the new bound meets its promise. We hope to be able to accomplish that, before this paper is actually published.

The Hightower-Hahn algorithm for branch-and-bound enumeration of the QAP was implemented merely by enhancing the lower bound code we developed earlier for the Hahn-Grant level-1 RLT bound. We purposely did not change the portion of that code which controls the branching strategy. Thus, the branching strategies are identical to those described in Hahn, et., al. [12]. In fact, the Hahn-Grant bound is still used for branching decisions near the root of the search tree. In [12] we pointed out the need for better branching information in terms of more accurate bounds near the root. Thus, it becomes a natural step for us to test whether the level-2 RLT bounds should be used for that purpose as well. When sufficient computing resources become available, this will be one of the first tasks on our agenda.

Another, but just as important area of investigation, will be the development of new strategies that take advantage of the many possible interactions between the linear, quadratic and cubic cost matrices. This should serve to reduce runtime and to give greater insight into the effectiveness of these dual-ascent procedures. We believe that such insights can be carried over to other optimization problems involving zero-one variables.

As mentioned in Section V, the new bound calculation currently requires large amounts of random access memory (RAM). It will therefore be necessary to find innovative ways to reduce the memory requirements of the level-2 RLT bound calculation without incurring significant loss in algorithm efficiency.

An interesting thought arises from this work. Perhaps the methods used in this investigation may find some application in solving integrated machine allocation and layout problems such as those discussed by T. Urban, et. al. [24].

Finally, we hope to use the newly developed algorithms that arise from this investigation to solve larger and larger QAP instances. Such progress should certainly encourage others to consider the QAP as a fertile ground for useful and challenging research.

If one succeeds in using level-2 RLT formulations for solving QAPs larger than size 30, it is likely that a level-3 RLT formulation will work even better. Understandably, it will require even larger amounts of RAM. We plan to experiment with level-3 RLT bounds and will try to find a practical implementation.

ACKNOWLEDGEMENTS

The authors wish to thank Professors Bakhtier Farouk and Nihat Bilgutay of the College of Engineering, Drexel University and Professor Preston B. Moore of the Department of Chemistry, University of Pennsylvania who graciously provided the computing resources necessary for this work. This work was supported in part by an international travel grant INT-9900376 from the National Science Foundation.

References:

- [1] Adams WP and Johnson TA, Improved Linear Programming-Based Lower Bounds for the Quadratic Assignment Problem, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1994; vol. 16, pp. 43-76.
- [2] Adams WP and Sherali HD, A Tight Linearization and an Algorithm for Zero-One Quadratic Programming Problems, Management Science, 1986; vol. 32, no. 10, pp. 1274-1290.
- [3] Anstreicher KM and Brixius NW, A New Bound for the Quadratic Assignment Problem Based on Convex Quadratic Programming, Mathematical Programming, 2001; 89, 341-357.
- [4] Anstreicher KM and Brixius NW, Solving quadratic assignment problems using convex quadratic programming relaxations, Optimization Methods and Software, 2001; 16, 49-68.
- [5] Anstreicher KM, Brixius NW, Goux J-P and Linderoth J, Solving large quadratic assignment problems on computational grids, to appear in Mathematical Programming, Series B, 2001; Currently available on the Web from <http://www.biz.uiowa.edu/faculty/anstreicher/mwqap.ps>.
- [6] Burkard RE, Karisch SE and Rendl F, QAPLIB - A Quadratic Assignment Problem Library, European Journal of Operational Research, 1991; vol. 55, no. 99, pp. 115-119.
- [7] Gilmore P.C., Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem, Journal of the Society of Industrial and Applied Mathematics, 1962; vol. 10, no. 2, pp. 305-313.
- [8] Grant TL, An Evaluation and Analysis of the Resolvent Sequence Method for Solving the Quadratic Assignment Problem, Master's Thesis, University of Pennsylvania, 1989.
- [9] Hahn PM, Minimization of Cost in Assignment of Codes to Data Transmission, Ph.D. Dissertation, University of Pennsylvania, 1968.
- [10] Hahn PM and Grant TL, Lower Bounds for the Quadratic Assignment Problem Based Upon a Dual Formulation, Operations Research, 1998; vol. 46, no. 6, Nov-Dec, pp. 912-922.
- [11] Hahn PM, Grant TL and Hall N, A Branch-and-Bound Algorithm for the Quadratic Assignment Problem Based on the Hungarian Method, European Journal of Operational Research, 1998; vol. 108, pp. 629-640.
- [12] Hahn P.M., Hightower W.L., Johnson TA, Guignard-Spielberg M. and Roucairol C., Tree elaboration strategies in branch-and-bound algorithms for solving the quadratic assignment problem, Yugoslav Journal of Operations Research, 2001; vol. 11 No.1, pp. 41-60.
- [13] Hahn PM and Krarup J, A hospital facility problem finally solved, The Journal of Intelligent Manufacturing, publication in late 2001. Also available on web site: <http://www.seas.upenn.edu/~hahn/>, 2001.
- [14] Johnson TA, New Linear Programming-Based Solution Procedures for the Quadratic Assignment Problem, Ph.D. Dissertation, Clemson University, Clemson, SC, 1992.
- [15] Koopmans T.C. and Beckmann M.J., Assignment Problems and the Location of Economic Activities, Econometrica, 1957; vol. 25, pp. 53-76.

- [16] Lawler EL, The Quadratic Assignment Problem, *Management Science*, 1963; vol. 9, pp. 586-599.
- [17] Munkres M, Algorithms for the Assignment and Transportation Problems, *Journal of the Society of Industrial and Applied Mathematics*, 1957; vol. 5, No. 1 (March), pp. 32-38.
- [18] Nugent CE, Vollman TE and Ruml J, An Experimental Comparison of Techniques for the Assignment of Facilities to Locations, *Operations Research*, 1968; vol. 16, pp. 150-173.
- [19] Ramachandran R and Pekny JF, Dynamic Factorization Methods for Using Formulations Derived from Higher Order Lifting Techniques in the Solution of the Quadratic Assignment Problem, *State of the Art in Global Optimization: Computational Methods and Applications*, Kluwer Academic Publishers, 1996; pp. 75-92.
- [20] Ramakrishnan KG, Resende MGC, Ramachandran B. and Pekny J.F., Tight QAP Bounds via Linear Programming, to appear in *Combinatorial and global optimization*, A. Migdalas and P.M. Pardalos, editors, Kluwer Academic Publishers, 2001.
- [21] Resende MGC, Ramakrishnan KG, and Drezner Z, Computing Lower Bounds for the Quadratic Assignment Problem with an Interior Point Algorithm for Linear Programming, *Operations Research*, 1995; vol. 43, pp. 781-79.
- [22] Sherali HD and Adams WP, A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems, *SIAM Journal on Discrete Mathematics*, 1990; vol. 3, No. 3, pp. 411-430.
- [23] Sherali HD and Adams WP, A Hierarchy of Relaxations and Convex Hull Characterizations for Mixed-Integer Zero-One Programming Problems, *Discrete Applied Mathematics*, 1994; vol. 52, no. 1, pp. 83-106.
- [24] Urban TL, Chiang W-C and Russell R. A., The Integrated Machine Allocation and Layout Problem, *International Journal of Production Research*, 2000; 38(13), pp. 2911–2930.

FIGURES w/ CAPTIONS

$$C = \begin{array}{c} \begin{array}{|c|c|c|c|} \hline C_{11} & C_{12} & C_{13} & C_{14} \\ \hline C_{21} & C_{22} & C_{23} & C_{24} \\ \hline C_{31} & C_{32} & C_{33} & C_{34} \\ \hline C_{41} & C_{42} & C_{43} & C_{44} \\ \hline \end{array} \\ \\ \begin{array}{|c|c|c|c|} \hline C_{1111} & & & \\ \hline C_{1122} & C_{1123} & C_{1124} & \\ \hline C_{1132} & C_{1133} & C_{1134} & \\ \hline C_{1142} & C_{1143} & C_{1144} & \\ \hline C_{2112} & C_{2113} & C_{2114} & \\ \hline C_{2132} & C_{2133} & C_{2134} & \\ \hline C_{2142} & C_{2143} & C_{2144} & \\ \hline C_{3112} & C_{3113} & C_{3114} & \\ \hline C_{3122} & C_{3123} & C_{3124} & \\ \hline C_{3142} & C_{3143} & C_{3144} & \\ \hline C_{4112} & C_{4113} & C_{4114} & \\ \hline C_{4122} & C_{4123} & C_{4124} & \\ \hline C_{4132} & C_{4133} & C_{4134} & \\ \hline C_{4141} & & & \\ \hline \end{array} & \begin{array}{|c|c|c|c|} \hline C_{1212} & & & \\ \hline C_{1221} & C_{1223} & C_{1224} & \\ \hline C_{1231} & C_{1233} & C_{1234} & \\ \hline C_{1241} & C_{1243} & C_{1244} & \\ \hline C_{2211} & C_{2213} & C_{2214} & \\ \hline C_{2221} & C_{2223} & C_{2224} & \\ \hline C_{2241} & C_{2243} & C_{2244} & \\ \hline C_{3211} & C_{3213} & C_{3214} & \\ \hline C_{3221} & C_{3223} & C_{3224} & \\ \hline C_{3241} & C_{3243} & C_{3244} & \\ \hline C_{4211} & C_{4213} & C_{4214} & \\ \hline C_{4221} & C_{4223} & C_{4224} & \\ \hline C_{4231} & C_{4233} & C_{4234} & \\ \hline C_{4241} & & & \\ \hline \end{array} & \begin{array}{|c|c|c|c|} \hline C_{1313} & & & \\ \hline C_{1321} & C_{1322} & * & C_{1324} \\ \hline C_{1331} & C_{1332} & & C_{1334} \\ \hline C_{1341} & C_{1342} & & C_{1344} \\ \hline C_{2311} & C_{2312} & & C_{2314} \\ \hline C_{2331} & C_{2332} & & C_{2334} \\ \hline C_{2341} & C_{2342} & & C_{2344} \\ \hline C_{3311} & C_{3312} & & C_{3314} \\ \hline C_{3321} & C_{3322} & & C_{3324} \\ \hline C_{3341} & C_{3342} & & C_{3344} \\ \hline C_{4311} & C_{4312} & & C_{4314} \\ \hline C_{4321} & C_{4322} & & C_{4324} \\ \hline C_{4331} & C_{4332} & & C_{4334} \\ \hline C_{4341} & & & \\ \hline \end{array} & \begin{array}{|c|c|c|c|} \hline C_{1414} & & & \\ \hline C_{1421} & C_{1422} & C_{1423} & \\ \hline C_{1431} & C_{1432} & C_{1433} & \\ \hline C_{1441} & C_{1442} & C_{1443} & \\ \hline C_{2411} & C_{2412} & C_{2413} & \\ \hline C_{2431} & C_{2432} & C_{2433} & \\ \hline C_{2441} & C_{2442} & C_{2443} & \\ \hline C_{3411} & C_{3412} & C_{3413} & \\ \hline C_{3421} & C_{3422} & C_{3423} & \\ \hline C_{3441} & C_{3442} & C_{3443} & \\ \hline C_{4411} & C_{4412} & C_{4413} & \\ \hline C_{4421} & C_{4422} & C_{4423} & \\ \hline C_{4431} & C_{4432} & C_{4433} & \\ \hline C_{4441} & & & \\ \hline \end{array} \end{array}$$

Figure 1. Quadratic Assignment Problem matrix of costs for $N = 4$.

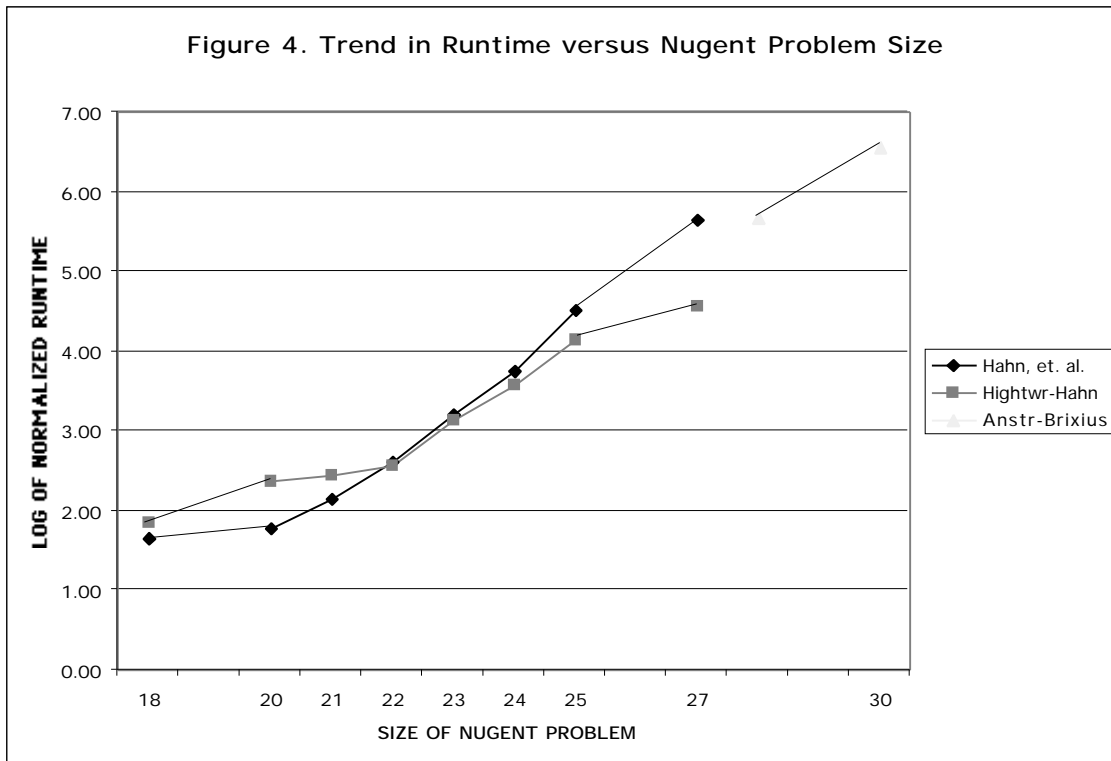
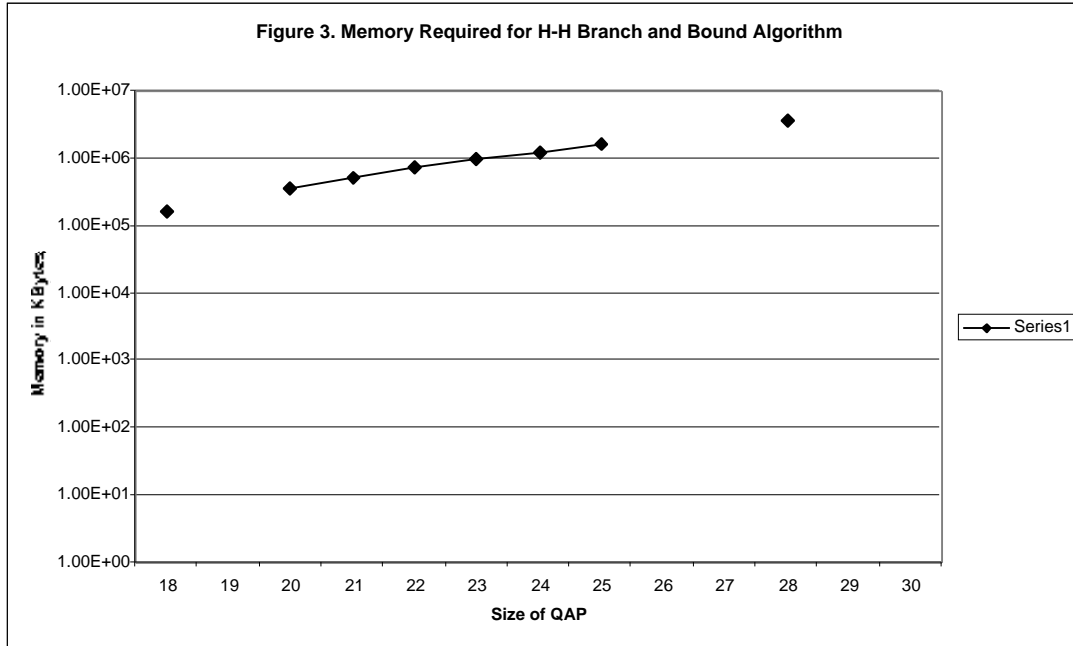
$$D = \begin{array}{|c|c|c|c|} \hline D_{11} & D_{12} & D_{13} & D_{14} \\ \hline D_{21} & D_{22} & D_{23} & D_{24} \\ \hline D_{31} & D_{32} & D_{33} & D_{34} \\ \hline D_{41} & D_{42} & D_{43} & D_{44} \\ \hline \end{array}$$

C_{1111}					C_{1212}					C_{1313}					C_{1414}
	C_{1122}	C_{1123}	C_{1124}	C_{1221}		C_{1223}	C_{1224}	C_{1321}	C_{1322}	*	C_{1324}	C_{1421}	C_{1422}	C_{1423}	
	D_{1122}	D_{1123}	D_{1124}	D_{1221}		D_{1223}	D_{1224}	D_{1321}	D_{1322}		D_{1324}	D_{1421}	D_{1422}	D_{1423}	
	C_{1132}	C_{1133}	C_{1134}	C_{1231}		C_{1233}	C_{1234}	C_{1331}	C_{1332}		C_{1334}	C_{1431}	C_{1432}	C_{1433}	
	D_{1132}	D_{1133}	D_{1134}	D_{1231}		D_{1233}	D_{1234}	D_{1331}	D_{1332}		D_{1334}	D_{1431}	D_{1432}	D_{1433}	
	C_{1142}	C_{1143}	C_{1144}	C_{1241}		C_{1243}	C_{1244}	C_{1341}	C_{1342}		C_{1344}	C_{1441}	C_{1442}	C_{1443}	
	D_{1142}	D_{1143}	D_{1144}	D_{1241}		D_{1243}	D_{1244}	D_{1341}	D_{1342}		D_{1344}	D_{1441}	D_{1442}	D_{1443}	
	C_{2112}	C_{2113}	C_{2114}	C_{2211}		C_{2213}	C_{2214}	C_{2311}	C_{2312}		C_{2314}	C_{2411}	C_{2412}	C_{2413}	
	D_{2112}	D_{2113}	D_{2114}	D_{2211}		D_{2213}	D_{2214}	D_{2311}	D_{2312}		D_{2314}	D_{2411}	D_{2412}	D_{2413}	
C_{2121}					C_{2222}					C_{2323}					C_{2424}
	C_{2132}	C_{2123}	C_{2134}	C_{2231}		C_{2233}	C_{2234}	C_{2331}	C_{2332}		C_{2334}	C_{2431}	C_{2432}	C_{2433}	
	D_{2132}	D_{2133}	D_{2134}	D_{2231}		D_{2233}	D_{2234}	D_{2331}	D_{2332}		D_{2334}	D_{2431}	D_{2432}	D_{2433}	
	C_{2142}	C_{2143}	C_{2144}	C_{2241}		C_{2243}	C_{2244}	C_{2341}	C_{2342}		C_{2344}	C_{2441}	C_{2442}	C_{2443}	
	D_{2142}	D_{2143}	D_{2144}	D_{2241}		D_{2243}	D_{2244}	D_{2341}	D_{2342}		D_{2344}	D_{2441}	D_{2442}	D_{2443}	
	C_{3112}	C_{3113}	C_{3114}	C_{3211}		C_{3213}	C_{3214}	C_{3311}	C_{3312}		C_{3314}	C_{3411}	C_{3412}	C_{3413}	
	D_{3112}	D_{3113}	D_{3114}	D_{3211}		D_{3213}	D_{3214}	D_{3311}	D_{3312}		D_{3314}	D_{3411}	D_{3412}	D_{3413}	
	C_{3122}	C_{3123}	C_{3124}	C_{3221}		C_{3223}	C_{3224}	C_{3321}	C_{3322}		C_{3324}	C_{3421}	C_{3422}	C_{3423}	
	D_{3122}	D_{3123}	D_{3124}	D_{3221}		D_{3223}	D_{3224}	D_{3321}	D_{3322}		D_{3324}	D_{3421}	D_{3422}	D_{3423}	
C_{3131}					C_{3232}					C_{3333}					C_{3434}
	C_{3142}	C_{3143}	C_{3144}	C_{3241}		C_{3243}	C_{3244}	C_{3341}	C_{3342}		C_{3344}	C_{3441}	C_{3442}	C_{3443}	
	D_{3142}	D_{3143}	D_{3144}	D_{3241}		D_{3243}	D_{3244}	D_{3341}	D_{3342}		D_{3344}	D_{3441}	D_{3442}	D_{3443}	
	C_{4112}	C_{4113}	C_{4114}	C_{4211}		C_{4213}	C_{4214}	C_{4311}	C_{4312}		C_{4314}	C_{4411}	C_{4412}	C_{4413}	
	D_{4112}	D_{4113}	D_{4114}	D_{4211}		D_{4213}	D_{4214}	D_{4311}	D_{4312}		D_{4314}	D_{4411}	D_{4412}	D_{4413}	
	C_{4122}	C_{4123}	C_{4124}	C_{4221}		C_{4223}	C_{4224}	C_{4321}	C_{4322}		C_{4324}	C_{4421}	C_{4422}	C_{4423}	
	D_{4122}	D_{4123}	D_{4124}	D_{4221}		D_{4223}	D_{4224}	D_{4321}	D_{4322}		D_{4324}	D_{4421}	D_{4422}	D_{4423}	
	C_{4132}	C_{4133}	C_{4134}	C_{4231}		C_{4233}	C_{4234}	C_{4331}	C_{4332}		C_{4334}	C_{4431}	C_{4432}	C_{4433}	
	D_{4132}	D_{4133}	D_{4134}	D_{4231}		D_{4233}	D_{4234}	D_{4331}	D_{4332}		D_{4334}	D_{4431}	D_{4432}	D_{4433}	
C_{4141}					C_{4242}					C_{4343}					C_{4444}

where, for example

$$D_{2134} = \begin{array}{cc} D_{213412} & D_{213413} \\ D_{213442} & D_{213443} \end{array}, \quad D_{1433} = \begin{array}{cc} D_{143321} & D_{143322} \\ D_{143341} & D_{143342} \end{array}, \quad \text{etc.}$$

Figure 2. Cubic Assignment Problem matrix of costs for $N = 4$.



TABLES

Nugent instance	Gilmore-Lawler	Resende et. al.	Hahn-Grant	Ramakrishnan, et. al.	Hightower-Hahn	Optimum solution
12	493	523	523	578	578	578
15	963	1041	1039	N/A	1150	1150
20	2057	2182	2179	N/A	2487	2570

No. Iterations	200	300	1000	1300
Bound value	2473.2	2480.8	2486.2	2486.7
Runtime (secs)	2,897.5	4,327.8	14,310.2	18,633.4

Size	Bound	CPU	CPU-seconds	Speed Ratio	No. of nodes evaluated	Who	Minutes Normal'd
20	QP	HP C3000	8,748	1.0	1,040,308	Anstr-Brixius	146
20	H-G-1	Ultra 10	5,087	0.7	181,073	Hahn, et. al.	59
20	H-H-2	HP J5000	9,929	1.4	2,257	Hightower-Hahn	232
22	QP	HP C3000	8,058	1.0	1,225,892	Anstr-Brixius	134
22	H-G-1	Ultra 10	48,917	0.7	1,354,837	Hahn, et. al.	571
22	H-H-2	HP J5000	15,246	1.4	2,381	Hightower-Hahn	356
24	QP	HP C3000	349,794	1.0	31,865,440	Anstr-Brixius	5,830
24	H-G-1	Alpha	1,487,724	0.48	16,710,701	Hahn, et. al.	11,902
24	H-H-2	SG10 2000	414,756	0.55	13,995	Hightower-Hahn	3,802
25	QP	HP C3000	715,000	1.0	71,770,751	Anstr-Brixius	11,917
25	H-G-1	HP J5000	1,393,117	1.4	27,409,486	Hahn, et. al.	35,506
25	H-H-2	SG10 2000	1,534,564	0.55	30,718	Hightower-Hahn	14,067
27	QP	HP C3000	5,676,480	1.0	~402,000,000	Anstr-Brixius	94,608
27*	H-G-1	Ultra 10	39,336,263	0.7	297,648,966	Hahn, et. al.	458,923
27	H-H-2	IBM	1,579,248	1.43	46,315	Hightower-Hahn	37,639
28	QP	HP C3000	27,751,680	1.0	~2,230,000,000	Anstr-Brixius	462,528
30	QP	HP C3000	218,859,840	1.0	11,892,208,412	Anstr-Brixius	3,647,664

*Results extrapolated from 95% elaboration of branch-and-bound tree.

N.B. Five of the results in Table 3 were reported in [12]. Recently, the CPU, speed ratios and minutes normalized for corresponding entries in Table 7 of [12] were found to be in error. They are correct here. A corrigendum is being sent to the Yugoslav Journal of Operations Research.