An Algorithm for the Generalized Quadratic Assignment Problem

Peter M. Hahn[1]
Electrical and Systems Engineering, University of Pennsylvania
Philadelphia, PA 19104-6314, USA
hahn@seas.upenn.edu

Bum-Jin Kim
Electrical and Systems Engineering, University of Pennsylvania
Philadelphia, PA 19104-6314, USA
bumjin@seas.upenn.edu

Monique Guignard
Operations and Information Management, The Wharton School,
University of Pennsylvania, Philadelphia, PA 19104-6340, USA
guignard@wharton.upenn.edu

J. MacGregor Smith
Mechanical and Industrial Engineering, University of Massachusetts Amherst
Amherst, MA 01003-5220, USA
jmsmith@ecs.umass.edu

Yi-Rong Zhu
Electrical and Systems Engineering, University of Pennsylvania
Philadelphia, PA 19104-6314, USA
yrzhu@seas.upenn.edu

[1] Corresponding author
Current address: 2127 Tryon Street, Philadelphia, PA 19146-1228, USA
Fax: 215-546-4043

An Algorithm for the Generalized Quadratic Assignment Problem

Abstract: This paper reports on a new algorithm for the Generalized Quadratic Assignment problem (GQAP). The GQAP describes a broad class of quadratic integer programming problems, wherein *M* pair-wise related entities are assigned to *N* destinations constrained by the destinations' ability to accommodate them. This new algorithm is based on a Reformulation Linearization Technique (RLT) dual ascent procedure. Experimental results show that the runtime of this algorithm is as good or better than other known exact solution methods for problems as large as *M*=20 and *N*=15.

Acknowledgements: This material is based upon work supported by the U.S. National Science Foundation under grant No. DMI-0400155.

1. Introduction

1.1 Background and Literature Review

The GQAP covers a broad class of problems that involve the minimization of a total pair-wise interaction cost among *M* equipments, tasks or other entities and where placement of these entities into *N* possible destinations is dependent upon existing resource capacities. These problems include finding the assignment of equipments to fixed locations given limited capacities at each possible location and the assignment of processing tasks to distributed computer processors, each with limited computing resources. The Lee and Ma [28] version of the problem can be stated with reference to a practical situation where it is desired to locate *M* equipments among *N* fixed locations, where for each pair of equipments (i,k) a certain flow of commodities $f(i,k)$ is known and for each pair of locations (j,n) a corresponding distance $d(j,n)$ is known. The two-way transportation costs between equipments *i* and *k*, given that *i* is assigned to location *j* and *k* is assigned to location *n*, are $f(i,k)\cdot d(j,n) + f(k,i)\cdot d(n,j)$. The objective is to find an assignment minimizing the sum of all such transportation costs given that the capacity or resource constraints are met. In the general case of the GQAP, the cost of transportation between equipments is known but is not decomposable into a product of a flow and a distance matrix.

This paper proposes a branch-and-bound algorithm for the GQAP, whose bound is based on a Lagrangean dual derived using the Reformulation-Linearization Technique (RLT) of Adams and Sherali [1, 2] and Sherali and Adams [41, 42, and 43]. RLT is known for generating tight linear programming relaxations, not only for constructing exact solution algorithms, but also to design powerful heuristic procedures for large classes of discrete combinatorial and continuous non-convex programming problems. A dual ascent procedure using first level or Level-1 RLT (or RLT1) is the basis for the algorithm.

Several years ago, Hahn and Grant [20] introduced the Level-1 RLT Dual ascent Procedure (DP) for solving difficult instances of the Quadratic Assignment Problem (QAP). Continuous improvements in its implementation have achieved QAP exact solution run times that are at the leading edge for single processor platforms. See Loiola et al. [29], Drezner et al [12] and Hahn et al [22, 23]. These methods have also been applied to solving practical instances of the Quadratic 3-Dimensional Assignment Problem (Q3AP) described in Guignard et al. [19].

Loiola et al. [29] compare recently developed lower bounds for the QAP. These include the interior-point bound of Resende et al. [35]; the RLT1 dual-ascent bound from Hahn and Grant [20]; the dual-based bound from Karisch et al. [25]; the Level-2 (or RLT2) interior point bound from Ramakrishnan et al. [33]; the Hahn-Hightower RLT2 dual-ascent bound from Adams et al. [3]; the quadratic programming bound of Anstreicher and Brixius [4]; the lift-and-project semi-definite programming (SDP) bound from Burer and Vandenbussche [9]; and the bundle method bounds of Rendl and Sotirov [34]. Of all these new bounds, the tightest are the two RLT2 bounds and the lift-and-project SDP bound. However, tightness is not the only issue in an exact algorithm. Speed is also important. The only QAP bounds that are competitive in branch-and-bound algorithms are the quadratic programming bound of Anstreicher and Brixius [4], the RLT1 dual-ascent bound of Hahn and Grant and the Hahn-Hightower RLT2 dual-ascent bound.

The GQAP is a generalization of the QAP. It is also a generalization of the Generalized Assignment Problem (GAP). The GAP consists of assigning a set of tasks to a set of agents with minimum cost, wherein each agent has a limited amount of a single resource and each task must be assigned to one and only one agent, requiring a certain amount of the resource of the agent. Ross and Soland [37] gave the first formal

definition. The GAP was motivated by applications such as assigning jobs to computers (Balachandran [5]). Other applications are plant location (Ross and Soland [37]) and vehicle routing (Fisher, M.L. and R. Jaikumar [15]). Fisher et al. [16] showed that the GAP is NP-hard. The majority of exact solution algorithms are based on linear relaxation (Benders and van Nunen [6]), Lagrangean relaxation (Fisher et al. [16]), (Guignard and Rosenwein [18]), constraint deletion (Ross and Soland [37]), Lagrangean decomposition (Jörnsten and Näsberg [24]), and column generation (Savelsbergh [40]). Numerous inexact solution methods are based on metaheuristic methods, such as in Laguna et al. [27], Yagiura et al [46, 47], Romeijn and Romeros Morales [36] and Diaz and Fernandez [11].

Lee and Ma [28] only recently formulated the GQAP. However, problems that are special cases, including the QAP, have long been of interest to researchers in various fields, both because of their wide applicability and their resistance to reliable computer solution. Problems which come under the class of GQAP include the Process Allocation Problem of Sofianopoulous [44, 45], the Constrained Module Allocation Problem of Elloumi et al. [14], the Quadratic Semi-Assignment Problem covered by Billionnet and Elloumi [8], the Multiprocessor Assignment Problem by Magirou and Milis [31], the Task Assignment and Multiway Cut Problems of Magirou [30], the Memory Constrained Allocation problem of Roupin [38] and the constrained Task Assignment Problem of Billionnet and Elloumi [7].

The GQAP is considered to be one of the most difficult combinatorial optimization problems. Sahni and Gonzalez [39] have shown that the QAP is NP-hard (Non-deterministic Polynomial-time hard) and that, unless P = NP, it is not even possible to find an ε-approximation algorithm for a constant ε.

Exact solution strategies for GQAP type problems have been successful for only small instances (approximately $M \leq 30$). As a result, researchers have put forth a significant amount of effort in developing inexact, or heuristic methods that obtain good suboptimal assignments, using a reasonable amount of CPU time. Cordeau et al. [10], the only publication other than [28] we have found that deals with the newly formulated GQAP, discusses a memetic heuristic for the GQAP. They do not report using this method to find exact solutions.

1.2 Exact Solution Methods

The only known exact solution procedure for the GQAP is that of Lee and Ma [28]. They introduced three linearization approaches together with a branch-and-bound algorithm. Their branch-and-bound strategy differs from our algorithm both in the partial assignment strategy and in the computation of lower bounds.

The single, most important and widely studied ingredient of branch-and-bound algorithms is the lower bounds used for fathoming partial assignments. Two aspects of these lower bounds are critical to their success. The first is their tightness, i.e., their ability to fathom partial assignments when only few partial assignments have been made. The second is their simplicity, which permits them to be computed rapidly on single processor machines. The methods described in this paper meet both these criteria. In Section 5, we compare runtimes of our algorithm with that of Lee and Ma, demonstrating that ours is indeed much faster than theirs.

1.3 Organization of the paper.

In Section 2, the governing equations for the general GQAP are presented along with a Level 1 Reformulation Linearization Technique (RLT1) formulation and a useful Lagrangean relaxation. In Section 3, an RLT1 Dual Ascent Procedure for solving the Lagrangean relaxation of Section 2 is presented, which in practice is efficient as a lower bounding technique. Section 4 describes the branch-and-bound algorithm that was written to test the effectiveness of the lower bounds generated by the RLT1 Dual Ascent Procedure. Section 5 describes the testing and evaluation of the RLT1 Dual Ascent Procedure as a lower bounding method in a branch-and-bound algorithm. The numbers of nodes (partial assignments) that have to be fathomed, as well as total branch-and-bound run times, are given. Finally, conclusions are presented in Section 6.

2. GQAP and the Reformulation-Linearization Technique (RLT).

We now introduce a formal model of the GQAP. As mentioned earlier, the quadratic costs can be very general, and don't need to exhibit the structure described in Lee and Ma [28]. The general form of the Generalized Quadratic Assignment problem is given by the following:

GQAP: $$\text{Minimize } Z = \sum_{i,j} B_{ij} \cdot u_{ij} + \sum_{i,j,k,n} C_{ijkn} \cdot u_{ij} \cdot u_{kn} \quad (1)$$

subject to

$$\sum_{i=1}^{M} a_{ij} u_{ij} \le S_j \quad (j=1,2,\ldots,N), \quad (2)$$

$$\sum_{j=1}^{N} u_{ij} = 1 \quad (i=1,2,\ldots,M). \quad (3)$$

$$u_{ij} \in \{0,1\} \quad \left(i = 1,2,...,M; j = 1,2,...,N\right), \quad (4)$$

where

B_{ij} is the linear cost of assigning entity i to location j

C_{ijkn} = is the quadratic cost of assigning entity i to location j and entity k to location n

A special case is $C_{ijkn} = f_{ik} d_{jn}$, where f_{ik} is a flow between entities and d_{jn} is a distance between locations. $\mathbf{F}, \mathbf{D}$ are each square matrices

u_{ij} is 1 iff entity i is assigned to location j

a_{ij} is the space needed if entity i is located at j

S_j is the space available at location j

M is the number of entities and N is the number of locations

The original model (GQAP) is transformed through the introduction of new variables into a linearized model called LIP, similar to those successfully used for solving the QAP. The technique proceeds as follows:

In order to transform GQAP into an equivalent Linearized mixed Integer Programming (LIP), let us first define:

$$v_{ijkn} = u_{ij} u_{kn}, \quad \forall (i,j,k,n). \quad (5)$$

The relation (5) cannot be included in LIP since we want LIP to be a linear model. Instead, in order to maintain the equivalence, we will have to include some constraints that are implied by (5).

Multiply both sides of (3), written as $\sum_n u_{kn} = 1, \quad \forall k,$ by $u_{ij}, i \neq k$, and obtain:

$$\sum_n v_{ijkn} = u_{ij} \quad \forall (i,j,k), i \neq k . \tag{6}$$

One could also multiply both sides of (3) by $u_{kn}, k \neq i$, and obtain:

$$\sum_j v_{ijkn} = u_{kn} \quad \forall (i,k,n), k \neq i . \tag{7}$$

Rather than using (7) in LIP, though, for computational reasons that will be made clear later in this section, given that $u_{ij}u_{kn} = u_{kn}u_{ij}, \forall (i,j,k,n),$ we will use the implied constraint

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

We also require that v_{ijkn} be nonnegative:

$$v_{ijkn} \geq 0 \quad \forall (i,j,k,n), i \neq k . \tag{9}$$

Using implications of (3) and (5), we can rewrite the objective function (1) of GQAP as

$$\sum_{j,n,i\neq k} C_{ijkn} v_{ijkn} + \sum_{i,j} (C_{ijij} + B_{ij}) u_{ij} . \tag{10}$$

The following LIP formulation, which consists of all equations from (2) through (10), except for (5) and (7), emerges:

LIP: Minimize $\sum_{j,n,i\neq k} C_{ijkn} v_{ijkn} + \sum_{i,j} (C_{ijij} + B_{ij}) u_{ij}$ (11)

subject to:

$$\sum_n v_{ijkn} = u_{ij} \qquad \forall (i,j,k), i \neq k \tag{12}$$

$$v_{ijkn} = v_{knij} \qquad \forall (i,j,k,n), i < k \tag{13}$$

$$\sum_i a_{ij} u_{ij} \leq S_j \qquad \forall (j) \tag{14}$$

$$\sum_j u_{ij} = 1 \qquad \forall(i) \tag{15}$$

$$v_{ijkn} \geq 0 \qquad \forall(i,j,k,n),\ i \neq k \tag{16}$$

$$u_{ij} \in \{0,1\} \qquad \forall(i,j)\,. \tag{17}$$

PROPERTY:

Problems GQAP and LIP are equivalent in the following sense. Given any feasible solution u of GQAP, there exists a feasible solution (u,v) in LIP with the same objective value. Conversely, given any feasible solution (u,v) of LIP, the corresponding solution u is feasible in GQAP with the same objective value.

PROOF:

For a given *u* satisfying the constraint of GQAP given in (2) through (4), let $v_{ijkn} = u_{ij}u_{kn}$, it is trivial to show that (*u*,*v*) is a feasible solution of LIP and the objective function values of GQAP and LIP match. Conversely, to show the other direction of the equivalence, given that LIP contains the constraints of GQAP with variables *u*, we have to show that a feasible solution (*u*,*v*) of LIP satisfies $v_{ijkn} = u_{ij}u_{kn}\ \forall(i,j,k,n),\ i < k$, which guarantees that the objective values of LIP and GQAP are equal. In fact, it suffices to show that if (*u*,*v*) is feasible to LIP v_{ijkn} is 0 unless u_{ij} and u_{kn} are both equal to 1, in which case it is also 1.

If $u_{ij} = 0$ for given *i* and *j*, then (12) and (16) together imply $v_{ijk'n'} = 0, \forall n', \forall k' \neq i$.

If $u_{kn} = 0$ for given *k* and *n*, then (12), (13) and (16) together imply $v_{i'j'kn} = 0, \forall j', \forall i' \neq k$.

So, if $u_{ij} = u_{kn} = 0$ for given *i*, *j*, *n*, and $k \neq i$, then $v_{ijkn} = 0$.

Now, we must show that $v_{ijkn} = 1$ if $u_{ij} = u_{kn} = 1$ for given *i*, $j, k \neq i$, and *n*.

From (15), $\sum_{j'} u_{ij'} = 1 \quad \forall i$, which implies that if $u_{ij} = 1$ then $u_{ij'} = 0 \quad \forall j' \neq j$.

If $u_{ij} = 1$ then from (12), $\sum_{n''} v_{ij'kn''} = u_{ij'} = 0 \quad \forall j' \neq j$, which in turn implies that $v_{ij'kn''} = 0 \quad \forall j' \neq j, \forall n''$, so in particular for $n'' = n$, $v_{ij'kn} = 0 \quad \forall j' \neq j$. Therefore, by (12) and (13),

$\sum_{j'} v_{knij'} = \sum_{j'} v_{ij'kn} = \sum_{j' \neq j} v_{ij'kn} + v_{ijkn} = v_{ijkn} = u_{kn}$, which implies that if $u_{kn} = 1$ then $v_{ijkn} = 1$. This completes the proof. □

The mathematical structure of LIP can be readily exploited via Lagrangean duality. A crucial step in constructing a Lagrangean dual of a given linear programming problem is the partitioning of the constraints into two sets: those constraints which are placed into the objective function through suitable multipliers and the remaining constraints which must still be satisfied, if LIP is to be optimized. The development of the Lagrangean dual for this problem is covered in Section 3.1.

Consider now, an $M^2 \times N^2$ solution (or assignment) matrix **V** that is a Kronecker product of the *M*×*N* assignment matrix **U** with itself. (Given two matrices **A** and **B**, a Kronecker product is formed by post-multiplying each scalar element of **A** by **B,** thus forming a new matrix whose dimensions are the products of the respective dimensions of **A** and **B**.)

That is,

$$\mathbf{V} = \mathbf{U}\mathbf{x}\mathbf{U} = \begin{pmatrix} u_{11}\mathbf{U} & u_{12}\mathbf{U} & \cdots & u_{1N}\mathbf{U} \\ u_{21}\mathbf{U} & u_{22}\mathbf{U} & \cdots & u_{2N}\mathbf{U} \\ \vdots & \vdots & \ddots & \vdots \\ u_{M1}\mathbf{U} & u_{M2}\mathbf{U} & \cdots & u_{MN}\mathbf{U} \end{pmatrix} = \left[v_{ijkn} \right]$$

where, from (13): $v_{ijkn} = v_{knij}$

Equation 13 is very important to the effectiveness of the RLT1 Dual Ascent Procedure. It says that if an element v_{ijkn} is involved in an assignment (i.e., equal to 1) then it has a "complementary element" v_{knij} that is also equal to 1. These complementary pairs have an interesting property; they are always in two different

submatrices that never occupy the same submatrix row. The equality between the pair creates a valuable communication between submatrices that can be exploited in calculating GQAP lower bounds.

The **V** matrix is a partitioned matrix whose elements are comprised of the Null matrix (all elements zero) and matrix **U**. Furthermore, **V** exhibits a gross pattern identical to that of matrix **U**. An example of a **V** matrix for $M = 4$ and $N = 3$ is shown in Figure 1.

$$\mathbf{V} = \left(\begin{array}{ccc|ccc|ccc}
\{1\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{0\} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\{1\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{0\} \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\{0\} & 0 & 0 & 0 & \{1\} & 0 & 0 & 0 & \{0\} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\{0\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{1\}
\end{array}\right) \qquad \mathbf{U} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 1. Example of Assignment Matrix **V** and its **U** matrix.

Certain elements of **V** are always zero, specifically

$$v_{ijin} = 0 \text{ if } j \neq n. \tag{18}$$

These elements are referred to as "disallowed" elements. Elements with subscript *ijij* (i=1,…,M; j=1,…,N) are termed "linear-cost elements" and are shown bracketed in the figure. Observe that if there are any 1's in a submatrix $u_{ij}\mathbf{U}$ its linear-cost element is always unity and vice versa. Observe, too, that one and only one unity element may exist in each row of **V**. These constraints follow directly from the definition of **V** as the

product of two solution matrices. A linear-cost element is special in another way; it has no complementary element, because $v_{ijij} = u_{ij}u_{ij}$.

Now, suppose one arranges the $M^2 \times N^2$ cost coefficients C_{ijkn} $\left(i,k = 1,...,M; j,n = 1,...,N\right)$ in an $M^2 \times N^2$ matrix **C,** similar to matrix **V** and indexed in precisely the same fashion. This is demonstrated in Figure 2. The asterisks denote disallowed elements. Since **V** comprises *M* copies of solution matrix **U**, it is easy to see that the submatrices of **C** in Figure 2 would have to each meet the constraints on **U**. Thus, a submatrix of **C** may be thought of as a GAP subproblem of GQAP, similar to the concept that a submatrix of the QAP cost matrix can be thought of as a Linear Assignment Problem (LAP) (Hahn and Grant [20]).

$$\mathbf{C} = \left(\begin{array}{c|c|c} \mathbf{C_{11}} & \mathbf{C_{12}} & \mathbf{C_{13}} \\ \hline \mathbf{C_{21}} & \mathbf{C_{22}} & \mathbf{C_{23}} \\ \hline \mathbf{C_{31}} & \mathbf{C_{32}} & \mathbf{C_{33}} \\ \hline \mathbf{C_{41}} & \mathbf{C_{42}} & \mathbf{C_{43}} \end{array}\right) = \left(\begin{array}{ccc|ccc|ccc} C_{1111} & * & * & * & C_{1212} & * & * & * & C_{1313} \\ C_{1121} & C_{1122} & C_{1123} & C_{1221} & C_{1222} & C_{1223} & C_{1321} & C_{1322} & C_{1323} \\ C_{1131} & C_{1132} & C_{1133} & C_{1231} & C_{1232} & C_{1233} & C_{1331} & C_{1332} & C_{1333} \\ C_{1141} & C_{1142} & C_{1143} & C_{1241} & C_{1242} & C_{1243} & C_{1341} & C_{1342} & C_{1343} \\ \hline C_{2111} & C_{2112} & C_{2113} & C_{2211} & C_{2212} & C_{2213} & C_{2311} & C_{2312} & C_{2313} \\ C_{2121} & * & * & * & C_{2222} & * & * & * & C_{2323} \\ C_{2131} & C_{2132} & C_{2133} & C_{2231} & C_{2232} & C_{2233} & C_{2331} & C_{2332} & C_{2333} \\ C_{2141} & C_{2142} & C_{2143} & C_{2241} & C_{2242} & C_{2243} & C_{2341} & C_{2342} & C_{2343} \\ \hline C_{3111} & C_{3112} & C_{3113} & C_{3211} & C_{3212} & C_{3213} & C_{3311} & C_{3312} & C_{3313} \\ C_{3121} & C_{3122} & C_{3123} & C_{3221} & C_{3222} & C_{3223} & C_{3321} & C_{3322} & C_{3323} \\ C_{3131} & * & * & * & C_{3232} & * & * & * & C_{3333} \\ C_{3141} & C_{3142} & C_{3143} & C_{3241} & C_{3242} & C_{3243} & C_{3341} & C_{3342} & C_{3343} \\ \hline C_{4111} & C_{4112} & C_{4113} & C_{4211} & C_{4212} & C_{4213} & C_{4311} & C_{4312} & C_{4313} \\ C_{4121} & C_{4122} & C_{4123} & C_{4221} & C_{4222} & C_{4223} & C_{4321} & C_{4322} & C_{4323} \\ C_{4131} & C_{4132} & C_{4133} & C_{4231} & C_{4232} & C_{4233} & C_{4231} & C_{4332} & C_{4333} \\ C_{4141} & * & * & * & C_{4242} & * & * & * & C_{4343} \end{array}\right)$$

Figure 2. Matrix of Cost for $M = 4$ and $N = 3$.

The existence of complementary pairs opens the door to considerable flexibility in the **C** matrix. If one element of a complementary pair is involved in an assignment, 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.

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

Class 1: Subtraction of a constant from all non-disallowed elements of a submatrix $\mathbf{C}_{ij}$ row and the corresponding addition of this constant to either another row of the submatrix or to the submatrix linear cost element $B_{ij} + C_{ijij}$. The term "disallowed" was defined immediately after (18) above.

Class 2: Addition or subtraction of a constant to all non-disallowed elements of any row in matrix **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. Consequently, the transfer of cost from one submatrix row to another will maintain the cost of all assignments that pass through the submatrix. Furthermore, since the linear cost of an involved submatrix must also be involved in the assignment, the transfer of cost from a submatrix row to the linear cost, and vice versa, will also leave the cost of all assignments that intersect the submatrix unchanged.

In contrast, Class 2 operations work on the $M^2 \times N^2$ matrix level, and change the cost of <u>all</u> assignments by the amount added to or subtracted from the matrix row. This is true because one and only one cost element in a row of **C** can be involved in the overall cost of an assignment.

3. A Dual Ascent Procedure Lower Bound Calculation

3.1 The Level 1 RLT Dual Ascent Procedure

The RLT1 Dual Ascent Procedure algorithm is based on the concept that constant amounts are subtracted from rows of each and every submatrix and added to the linear cost element of its submatrix (Class 1

operations). Further, constant amounts are then subtracted from the rows of **C** containing only linear costs and added to the expression in (11) (Class 2 operations). These are the steps:

1. For each of the $M \times N$ submatrices, first collect complementary costs into the submatrix. Then subtract the minimum cost from each row not containing the linear cost element and add it to the submatrix linear cost element. Adding complementary costs first assures the largest possible transfer of quadratic to linear costs.
2. After Step 1 has been done for all the submatrices, subtract the minimum cost from each row of the linear cost matrix and add it to a reduction constant which constitutes a lower bound to the problem. If the resulting pattern of zeros satisfies the problem constraints, the procedure terminates with an optimum solution to the problem. If not, continue to Step 3. Stop if bound increase is poor.
3. Scan the reduced matrix of linear costs for positive elements. Record locations of the non-positive (i.e., zero) linear costs as this information will be needed in Step 4. Divide each positive linear cost element into *M-1* approximately equal parts and add each part to a different (non linear-cost) row of its submatrix. Positive linear costs are divided into *M*-1 approximately equal parts as follows: First of all, one must understand that it is essential that round-off errors must never be allowed in manipulating the C matrix. Thus, the C matrix in our algorithm is always integer. So, to divide the linear cost $B_{ij} + C_{ijij}$ into *M*-1 almost equal parts, we divide $B_{ij} + C_{ijij}$ by *M*-1 and round that down to the nearest integer. That provides *M*-2 equal parts. We then add those parts up and subtract from $B_{ij} + C_{ijij}$ to get the remaining part. The motivation here is that by replacing costs into submatrices, there is a new opportunity to make those linear costs which had low or zero values after Step 2 larger, permitting even more cost to be eventually moved to the lower bound. Step 3 leaves the cost matrix **C** dramatically rearranged. Because of this rearrangement, the process can be repeated, i.e., additional costs can be subtracted from within submatrices and moved to the linear cost element. The result is an iterative procedure that produces growth in the lower bound with each round. This growth, while significant, diminishes with each round, so at some point it does not pay to continue the process.

4. Repeat Step 1, except this time be sure to do the collection of complementary costs and subtractions first for those submatrices whose linear costs were zero prior to Step 3. Go to Step 2.

3.2 Linear programming considerations

In order to see how the Dual Ascent Procedure attempts to calculate a lower bound on LIP it is necessary to obtain a smaller reformulation via the substitution of variables suggested by constraints (13), thereby reducing the number of variables and constraints each by $N^2M(M\text{-}1)/2$, in addition to halving the number of non-negativity restrictions in (16). Finally, we use the fact that $u_{ij}^{\,2} = u_{ij}$ and replace u_{ij} by v_{ijij}, then the v_{ijij}'s become 0-1 variables in the new model. Notice that in Section 2, one could have used (7) instead of (13) in the definition of LIP, and with some modifications in the proof, one would have obtained another equivalent formulation for GQAP. It is this formulation that is the basis for the new model LIP1 below.

Constraint (14) becomes

$$\sum_k a_{kn} v_{knkn} \le S_n \qquad \forall(j)$$

Multiplying this constraint by v_{ijij} produces

$$\sum_k a_{kn} v_{knkn} v_{ijij} \le S_n v_{ijij} \qquad \forall(i,j,n)\,.$$

Now remember that $v_{knkn} = u_{kn}$ and $v_{ijij} = u_{ij}$, thus $v_{knkn} v_{ijij} = u_{kn} u_{ij} = v_{knij} = v_{ijkn}, \forall i,j,k,n.$ Keeping only the relevant v_{knij} or v_{ijkn}, we obtain

$$\sum_{\substack{k \\ k>i}} a_{kn} v_{ijkn} + \sum_{\substack{k \\ k<i}} a_{kn} v_{knij} \le S_n v_{ijij}\,,$$ or in a weakened form, since $v_{ijij} \le 1,$

$$\sum_{\substack{k \\ k>i}} a_{kn} v_{ijkn} + \sum_{\substack{k \\ k<i}} a_{kn} v_{knij} \le S_n\,.$$

We then replace the binary restrictions (17) on $u_{ij} = v_{ijij}$ by $0 \le u_{ij} = v_{ijij} \le 1$, so that contrary to LIP that was a mixed-integer programming problem, LIP1 is a continuous optimization problem. We obtain the following model:

LIP1: Minimize $\sum_{\substack{i,j,k,n \\ k>i}} \tilde{C}_{ijkn} v_{ijkn} + \sum_{i,j} \left(B_{ij} + C_{ijij} \right) v_{ijij}$ where $\tilde{C} = C_{ijkn} + C_{knij}$

subject to:

$$\sum_{\substack{k \\ k>i}} a_{kn} v_{ijkn} + \sum_{\substack{k \\ k<i}} a_{kn} v_{knij} \le S_n \qquad \forall (i,j,n) \qquad (19)$$

$$\sum_{n} v_{ijkn} = v_{ijij} \qquad \forall (i,j,k),\ k > i \qquad (20a)$$

$$\sum_{j} v_{ijkn} = v_{knkn} \qquad \forall (i,k,n),\ k > i \qquad (20b)$$

$$\sum_{k} a_{kn} v_{knkn} \le S_n \qquad \forall (n) \qquad (21)$$

$$\sum_{n} v_{knkn} = 1 \qquad \forall (k) \qquad (22)$$

$$\begin{aligned} 0 \le v_{ijkn} \le 1 \qquad & \forall (i,j,k,n),\ k>i \\ 0 \le v_{knkn} \le 1 \qquad & \forall (k,n) \end{aligned} \quad . \qquad (23)$$

Notice that when added to the continuous model LIP1, constraint (19) may tighten it, because it makes use of information on v_{ijij} that is not initially contained in LIP1, i.e., (19) is not implied by constraints (20) to (23).

Now consider the following Lagrangean dual, whereby constraints (20a) and (20b) are placed into the objective function using multipliers **b** and **b'** respectively:

LD1: Maximize $\theta(\mathbf{b},\mathbf{b'})$ where

$$\theta(\mathbf{b},\mathbf{b'})=\min_{\mathbf{v}}\left[\sum_{\substack{i,j,k,n\\k>i}}\left(\tilde{C}_{ijkn}-b_{ijk}-b'_{ikn}\right)v_{ijkn}+\sum_{i,j}\left(B_{ij}+C_{ijij}+\sum_{\substack{k\\k>i}}b_{ijk}+\sum_{\substack{k\\i>k}}b'_{kij}\right)v_{ijij}\right], \qquad (24)$$

such that $0 \le v_{ijij} \le 1$ and subject to (19), (21), (22) and (23).

The operations implied in LD1 are just the Class 1 operations defined earlier. That is, constant amounts subtracted from (or added to) submatrix rows are added to (or subtracted from) their corresponding linear cost. The collecting of complimentary costs prior to solving each submatrix GAP is explained by the substitution of variables that eliminated constraints (16) in LIP. Thus, Steps 1 and 4 of the Dual Ascent Procedure are fully explained by this formulation. Since Step 3 of the Dual Ascent Procedure 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 (22) are placed into the objective function using multipliers **c**.

LD2: Maximize $\theta(\mathbf{b},\mathbf{b'},\mathbf{c})$ where

$$\theta(\mathbf{b},\mathbf{b'},\mathbf{c})=\min_{\mathbf{v}}\left[\sum_{\substack{i,j,k,n\\k>i}}(\tilde{C}_{ijkn}-b_{ijk}-b'_{ikn})v_{ijkn}+\sum_{i,j}\left(B_{ij}+C_{ijij}+\sum_{\substack{k\\k>i}}b_{ijk}+\sum_{\substack{k\\i>k}}b'_{kij}-c_i\right)v_{ijij}+\sum_i c_i\right], \qquad (25)$$

such that $0 \le v_{ijij} \le 1$ and subject to (19), (21) and (23).

LD2 amounts to solving the following Lagrangean subproblems for given multipliers **b**, **b'** and **c**:

LR2(**b**,**b',c**):

$$\sum_k c_k+\sum_n\left\{\begin{array}{l}\min_{\mathbf{v}}\left[\sum_{\substack{i,j,k\\k>i}}(\tilde{C}_{ijkn}-b_{ijk}-b'_{ikn})v_{ijkn}+\sum_k\left(B_{kn}+C_{knkn}+\sum_{\substack{i\\i>k}}b_{kni}+\sum_{\substack{i\\i<k}}b'_{ikn}-c_k\right)v_{knkn}\right]\\ \text{subject to:}\\ \qquad\sum_{\substack{k\\k>i}}a_{kn}v_{ijkn}+\sum_{\substack{k\\k<i}}a_{kn}v_{knij}\le S_n \qquad \forall(i,j) \qquad (19)\\ \qquad\sum_k a_{kn}v_{knkn}\le S_n \qquad (21)\\ \qquad\begin{pmatrix}0\le v_{ijkn}\le 1 & \forall(i,j,k),\ k>i\\ 0\le v_{knkn}\le 1 & \forall(k)\end{pmatrix} \qquad (23)\end{array}\right\}$$

The n-th subproblem is solved over all v_{ijkn}, for $k>i$ and for all j, and all v_{knkn}, for all k . If all objective function coefficients are kept nonnegative, there is no incentive to make any variable positive, hence the n-th Lagrangean subproblem will always have a zero optimal solution, and the Lagrangean dual optimum will be $\sum_k c_k$ for the best combination of multipliers **b**, **b'** and **c**. Keeping the objective function coefficients nonnegative is not really a constraint on C, but is computationally convenient, as it keeps the solution of the Lagrangean problem equal to 0. Although, it could lead to suboptimal multipliers,

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

If operations on **C** decrease the cost by an amount R' and are performed in a way that keeps the elements of **C** non-negative then no assignment cost can become negative and the following relationship holds:

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

Thus a 'dual ascent procedure' bound calculation for the Generalized Quadratic Assignment Problem can be stated as follows: Maximize the sum of downward cost shifts R' permitted by Class 1 and 2 operations, under the constraint that no cost element in **C** is driven negative.

3.3 Adding the Resource Constraints to the Dual Ascent Procedure

Equality constraints (20a), (20b) and (22) were dualized and dealt with appropriately in the construction of the dual ascent procedure. The resource constraints (19) and (21) must be enforced in other ways. For the most part, this is done by calculating dual ascent procedure lower bounds on only those assignments that satisfy (19) and (21). However, an ingenious method was implemented that improves the dual ascent procedure bound value, by taking into account the resource constraints. Constraints (19) are enforced during the RLT1 lower bound calculation by raising to a very high cost value those submatrix elements whose selection would violate (19). Thus, it is possible to improve the lower bound beyond that available if only the equality constraints (20a) and (20b) were enforced.

Consider the Class 1 subtraction operations defined in Section 2. After collecting complementary costs in a given submatrix $\mathbf{C_{ij}}$, minimum costs are subtracted from submatrix rows and added to the corresponding linear cost element C_{ijij}. Prior to these subtractions, an important step has been added. Each element in the column containing the linear cost element is examined. If including this element in a solution would render infeasible for constraints (19) the solution pattern for this submatrix, then the cost for this element is raised to an arbitrarily large number called GREAT and dictated only by the integer size limitations of the computer. This process effectively bars infeasible elements from inclusion in a RLT1 Dual Ascent Procedure solution, and focuses the reduction efforts on those elements still eligible for consideration. The ensuing subtractions permit additional cost to be extracted from each and every submatrix, resulting in a much-improved lower bound. The resulting improvement in lower bounds is stunning. On a typical branch-and-bound enumeration, runtimes are improved by a factor of 3 and the number of nodes evaluated is improved by a factor of 6.

3.4 Advantages of the RLT1 Dual Ascent Procedure

The RLT1 Dual Ascent Procedure has a number of valuable properties that makes it ideal for use in a branch-and-bound algorithm:

- The RLT1 Dual Ascent Procedure is still among competing lower bounding procedures for the QAP, as demonstrated from experimental results (see Table 2 of Loiola et al. [29] and Drezner et al. [12]).
- Similar to a number of other lower bounding procedures, the RLT1 Dual Ascent Procedure generates a series of non-decreasing lower bounds on the GQAP. More importantly, with each such lower bound, it modifies the GQAP objective function so that a new GQAP is generated whose feasible solution set is identical to that of the original and whose objective function values are merely lessened by the amount of the lower bound.
- The Dual Ascent Procedure for a given partial assignment can be stopped as soon as the lower bound on the assumed partial assignment exceeds an upper bound on the original problem. It can also be stopped,

in favor of making an additional partial assignment, when it becomes obvious that from its slow progress it is unlikely to ever reach the upper bound.

- In a branch-and-bound algorithm, fathoming decisions can easily be recorded by arbitrarily increasing the costs of those elements in the cost matrix that correspond to partial assignments which have been eliminated as possibly being optimum. This modifies the GQAP, but is assured to fully enumerate all feasible solutions to the original problem.

4. The Level 1 RLT Dual Ascent Procedure in Branch-and-bound.

The RLT1 Dual Ascent Procedure is utilized within a branch-and bound algorithm (called the RLT1_B&B) as the auxiliary process for computing lower bounds. While there are some parallels between RLT1_B&B and the branch-and-bound algorithm of Hahn et al. [21], there are significant differences since the GQAP tree search must be limited to only those assignments that meet the resource constrains (19).

4.1 Branching Strategy

Branching follows the conventional technique of selecting a single facility-location assignment at the first (highest) level, as well as at subsequent levels of partial assignment. In order to implement this selection, a linear cost is chosen to be involved in the assignment. Based on the selection of linear cost C_{ijij}, submatrix $\mathbf{C}_{ij}$ is involved in the assignment. The remainder of the submatrices in the row containing submatrix $\mathbf{C}_{ij}$ disappear (as they cannot be involved in the assignment) and the problem is thus reduced to a GQAP of size $(M\text{-}1) \times N$. It turns out that one row likewise disappears from each submatrix of the original problem, with the exception of the original submatrix $\mathbf{C}_{ij}$, which remains $(M\text{-}1) \times N$ in size. To complete the formulation of the newly formed $(M\text{-}1) \times N$ problem, this submatrix is added (by simple matrix addition) to the now size $(M\text{-}1) \times N$ matrix of linear costs.

It is the application of the RLT1 Dual Ascent Procedure lower bounding calculation on the $(M\text{-}1) \times N$ size problem that attempts to fathom a partial assignment postulated by the selection of linear cost C_{ijij}. By

fathoming, one calculates a lower bound and tests it against the best-known upper bound. If the best know upper bound is exceeded, the partial assignment is eliminated from the problem.

Recall in Section 3, the RLT1 Dual Ascent Procedure moves costs out of the **C** matrix into a lower bound value, leaving a modified matrix **C**'. For subsequent branch-and-bound operations along a given partial assignment path, the strategy is to take advantage of this fact and to use this reduced cost matrix **C**' for setting up subsequent sub-problems deeper into the tree. Thus, lower bounds are calculated not from the original problem, but from the sub-problems that the RLT1 Dual Ascent Procedure already processed at earlier (higher) levels of partial assignment. Using the modified matrix **C**' of each of these sub-problems has the additional benefit that the sub-problem is brought closer to dual solution, making it more likely that better feasible solutions will be found or that lower bounds will exceed upper bounds, thus cutting off a branch. The choice of the number of dual iterations at a given partial assignment is a dynamic decision, based on the progress of the lower bound achieved after a fixed number of iterations. If after a small number of iterations at the current partial assignment, sufficient progress is not reached, the lower bounding attempt stops and the algorithm proceeds to make an additional assignment. The choice of this threshold was determined experimentally.

4.2 Tree Elaboration Strategy

As mentioned above, tree search is depth-first and based on single-facility location assignment. The order in which facility assignments are made is very important and has profound influence on branch-and-bound runtime. A detailed study of the order in which assignments should be made is covered in Hahn et al. [22] for the QAP. Experimental testing on the GQAP confirms that similar rules hold for the GQAP.

Levels in the search tree are defined by how many partial assignments are made. The root is where no partial assignments have been made and is considered level 0. A single partial assignment is considered level 1. A pair of partial assignments is considered level 2, etc. Since the tree search involves one-to-one assignments, at each level of the tree it is necessary only to examine a given facility and its assignments to all possible (i.e., feasible) locations.

In the GQAP, more than one facility can reside at a given location. Therefore, one has to exhaust all possible locations at a given level, before one can determine if a better solution exists or if a branch of the tree at that level can be cut off. The exception to this rule is that no further consideration need be given to evaluating assignments if the resource constraints tell us that the locations can accept no further assignments.

Figure 3 illustrates the tree elaboration strategy and thus constitutes a high level flowchart for the branch-and-bound algorithm.

4.3 Search Strategy

The search strategy for selecting the next node is a simple depth first strategy. If fathoming a given node is unsuccessful, one makes an additional partial assignment, thus increasing the depth into the tree. If a node has been fathomed successfully, the next available node is selected. Partial assignments at a given level are selected according to a set of look-ahead bounding calculations that essentially determine the difficulty of eliminating the branch containing that assignment. Assignments selected in the order of increasing difficulty are made when it is advantageous to prune the tree quickly. Assignments selected in the order of decreasing difficulty are made when it is desired to find attractive feasible solutions to replace the upper bound, early in the search.

When the algorithm accumulates sufficient fathoming information, permanent decisions can be made on the assignment matrix that certain elements of the assignment matrix v_{ijkn} are zero. These are recorded in the modified cost matrix **C**' by setting the corresponding element costs to a very large value, denoted 'GREAT'. This effectively bars the 'decided zero' elements from inclusion in a RLT1 Dual Ascent Procedure solution, and focuses the reduction efforts on those elements still eligible for consideration. The communication of permanent decisions to the RLT1 Dual Ascent Procedure generally permits additional cost to be extracted from the matrix, resulting in an improved lower bound.

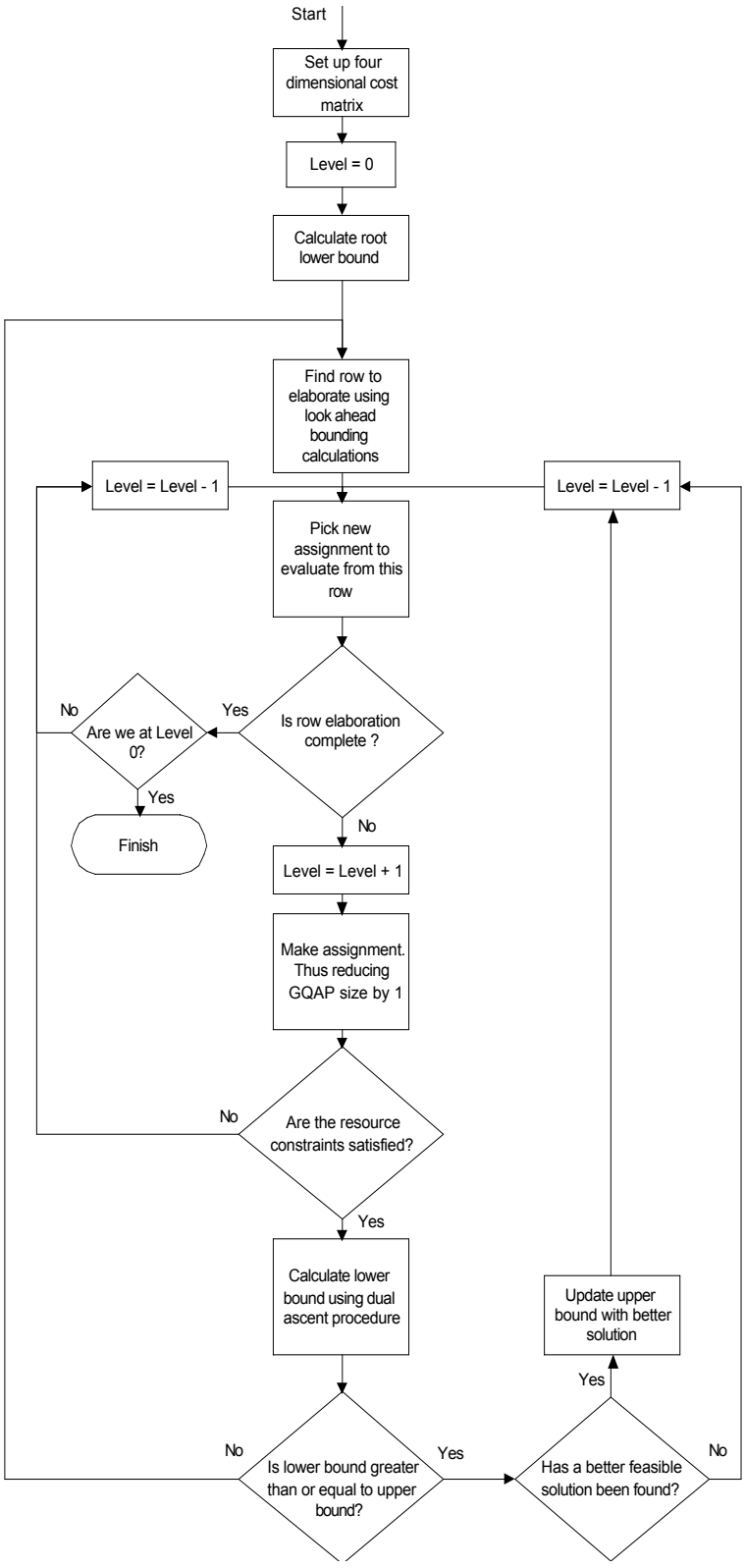

Figure 3 Flowchart of Branch-and-Bound Algorithm to Illustrate Tree Elaboration

5. RLT1 Dual Ascent Procedure Branch-and-Bound Evaluation

Eight GQAP instances were solved exactly using the RLT1_B&B. The experiments were run on a single 733 MHz CPU of a Dell 7150 PowerEdge server. The code is FORTRAN 77 and compiled using Intel FORTRAN 8.0 for Linux. Column 3 of Table 1 gives the optimum value of the instance. Column 4 gives the number of optima. Where available, Column 5 gives the runtimes and nodes elaborated by other researchers. Column 6 gives the RLT1_B&B total run time in seconds and the number of nodes searched. Column 7 gives the time it took the RLT1-B&B to reach the first occurrence of an optimum solution during the branch-and-bound elaboration. Instances #1-#4 were selected from a web site dedicated to the Task Assignment Problem (TAP) and the Constrained Task Assignment Problem (CTAP) set up by Sourour Elloumi at the Centre de Recherche en Informatique du CNAM (http://cedric.cnam.fr/oc/TAP/TAP.html). For instance #3, linearization and semidefinite programming were employed to give lower bounds in the CNAM branch-and-bound method (see Elloumi et al. [13] and Roupin [38]). Instance #5 was selected from Cordeau et al. [10]. Instances #6-#8 were selected from Lee and Ma [28], who solved those instances by branch-and-bound. Their results are shown in the last three rows of Column 5.

Table 1 – Optima and branch-and-bound time/nodes

GQAP instance	Size	Optimum value	No. of optima	Time/nodes others in secs	Time/nodes RLT1_B&B in secs	Secs to optimum
#1 Elloumi c2005De	20×5	5435	3	N/A	17,640 20,498,196 nodes	14,390
#2 Elloumi 2005Aa	20×5	3059	1	N/A	136 211,447 nodes	128
#3 Elloumi 2408Ca	24×8	1028	2	11.0 [1] 5,213 nodes	6.3 1,880 nodes	3.3
#4 Elloumi 2408Aa	24×8	5643	1	N/A	719,862 226,961,852 nodes	185,726
#5 CGL&M 20-15-35	20×15	1471896	1	N/A[3]	735 54,496 nodes	528.8
#6 Lee & Ma 14×9-E-G	14×9	2326370	1	3,279 [2] nodes N/A	169 207,666 nodes	90.7
#7 Lee & Ma 15×8	15×8	2856850	1	5,008 [2] nodes N/A	130 103,893 nodes	39.5
#8 Lee & Ma 16×7-1115	16×7	2809870	1	5,922 [2] nodes N/A	548 540,733 nodes	512.6

N/A = not available [1]from Elloumi by e-mail (CPU unknown) [2]Pentium III 677 MHz
[3]Cordeau et al.'s memetic heuristic finds opt. in an avg. 933 secs on a Sun 1.2 GHz workstation

As mentioned in the first paragraph of Section 4.3, the branch-and-bound search strategy has several variations, one of which permits the finding of attractive feasible solutions early in the search process. When this strategy is implemented, the experimental test cases show very attractive feasible solutions found very close to the start of the branch-and-bound enumerations. This is shown clearly in Figure 4 where, for four GQAP instances, the normalized objective function value is plotted as a function of normalized time to reach the optimum objective function value. Details not shown in Figure 4 are presented in the next paragraph.

For the very hard to solve c2005De test case (instance #1), the solution value at 3.4% of the branch-and-bound runtime is only 7% higher than the optimum solution value. Getting this very attractive solution took about 1000 seconds on a 733 MHz CPU of a Dell 7150 server. For the 2005Aa test case (instance #2), the solution value at 1.1% of the branch-and-bound runtime is 17% higher that the optimum solution value. For this test case, the solution value at 13.3% of the branch-and-bound runtime is only 4% higher than the optimum solution value. For the 20-15-35 test case (instance #5), the solution value at 2.7% of the branch-and-bound runtime is only 4% higher than the optimum solution value. For the size 16×7 test case (instance #8), the solution value at 0.6% of the branch-and-bound runtime is 17% higher than the optimum solution value.

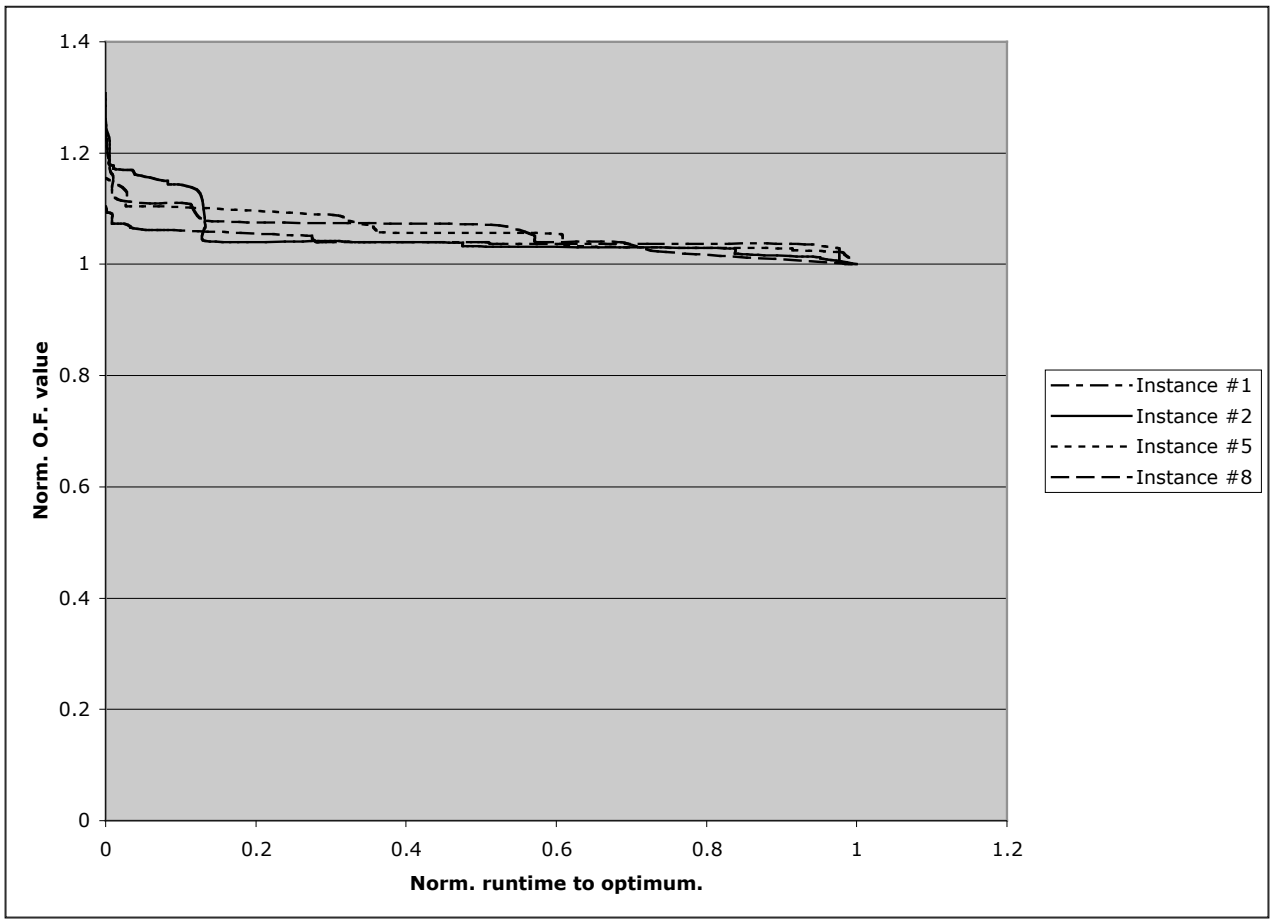

Figure 4 Objective Function Value vs. Time to Optimum

6. Conclusions and future challenges

This paper describes a Lagrangean dual for the GQAP based on a Level 1 Reformulation Linearization Technique (RLT1) Dual Ascent Procedure similar to one successfully used for solving the Quadratic Assignment Problem (QAP). A unique and very important aspect of the RLT1 Dual Ascent Procedure is that at each stage, the GQAP is restructured as fully equivalent to the original GQAP in a manner that brings it closer to solution. Just as importantly, at each stage of the RLT1 Dual Ascent Procedure, a high quality lower bound is economically calculated that is ideally suited for use in a branch-and-bound algorithm. The dual ascent procedure described herein is applicable whether or not it is possible to decompose the quadratic cost matrix in the objective function into a product of a "flow" and a "distance" matrix. Thus, the algorithms presented here are of wider applicability than other published methods.

The RLT1 Dual Ascent Procedure was embedded in a branch-and-bound algorithm that is also unique in many respects. A number of test cases supplied by Elloumi [13] and by Cordeau et al. [10], some previously unsolved, were solved in record time. The RLT1_B&B is generally faster than the other methods and is perhaps 20 times faster than the only other known result for the difficult Lee & Ma 16_7 instance. Experimentation with the addition of a Subgradient Optimization technique for getting better lower bounds from the individual submatrix GAPs shows promise. If successful, it will be covered in a separate publication. In addition, new heuristic search strategies for searching the solution space will be considered. Preliminary experiments with some new search strategies have been promising, yet a detailed analysis of their overall effectiveness remains to be completed.

Also, it has been shown that the branch-and-bound method could be utilized in a way that produces good feasible solutions (within the range 4% to 17% higher than the optimum value) at runtimes that are less than 4% of the branch-and-bound runtime. It is planned that the next phase of research on this widely applicable combinatorial optimization problem will address the development of heuristic (probably meta-heuristic) methods for achieving good solutions. This is necessary to be able to handle even larger problem instances than are achievable with exact solution methods and also to provide the branch-and-bound algorithm with good starting upper bounds, thus reducing branch-and-bound runtime.

References

1. W.P. Adams and H.D. Sherali, "A tight linearization and an algorithm for zero-one quadratic programming problems," Management Science vol. 32, pp. 1274-1290, 1986.
2. W.P. Adams and H.D. Sherali, "Linearization strategies for a class of zero-one mixed integer programming problems," Operations Research vol. 38, pp. 217-226, 1990.
3. W.P. Adams, M. Guignard, P.M. Hahn, and W.L. Hightower, "A level-2 reformulation-linearization technique bound for the quadratic assignment problem," European Journal of Operational Research, to appear.

4. K.M. Anstreicher and N.W. Brixius. “A new bound for the quadratic assignment problem based on convex quadratic programming,” Mathematical Programming vol. 89, pp. 341-357, 2001

5. V. Balachandran, “An integer generalized transportation model of optimal job assignment to computer networks,” Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, Working Paper 34-72-3, 1972.

6. J.F. Benders and J.A.E.E. van Nunen, “A property of assignment type mixed integer linear programming problems,” Operations Research Letters vol. 2, pp. 47-52, 1983.

7. A. Billionnet and S. Elloumi, “An algorithm for finding the k-best allocations of a tree structured program,” Journal of Parallel and Distributed Computing vol. 26, pp. 225-232, 1995.

8. A. Billionnet and S. Elloumi, “Best reduction of the quadratic semi-assignment problem,” Discrete Applied Mathematics (DAM) vol. 109, pp. 197-213, 2001.

9. S. Burer and D. Vandenbussche, "Solving lift-and-project relaxations of binary integer programs," SIAM Journal on Optimization vol. 16, pp. 726-750, 2006.

10. J-F. Cordeau, M. Gaudioso, G. Laporte, and L. Moccia, “A memetic heuristic for the generalized quadratic assignment problem,” INFORMS Journal on Computing, to appear.

11. J.A. Diaz and E. Fernandez, “A tabu search heuristic for the generalized assignment problem,” European Journal of Operational Research vol. 132, pp. 22-38, 2001.

12. Z. Drezner, P.M. Hahn, and E. Taillard, “Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods,” Annals of Operations Research vol. 139, pp. 65-94, 2005.

13. S. Elloumi, “The task assignment problem and the constrained task assignment problem,” Instances and Solution files, 2003. Available at: http://cedric.cnam.fr/oc/TAP/TAP.html.

14. S. Elloumi, F. Roupin, and E. Soutif, “Comparison of different lower bounds for the constrained module allocation problem,” Technical Report TR CEDRIC No 473, 2003.

15. M.L. Fisher and R. Jaikumar, “A generalized assignment heuristic for vehicle routing,” Networks vol. 11, pp. 109-124, 1981.

16. M.L. Fisher, R. Jaikumar and L.N. Van Wassenhove, “A multiplier adjustment method for the generalized assignment problem,” Management Science vol. 32, pp. 1095-1103, 1986.

17. T. L. Grant, “An evaluation and analysis of the resolvent sequence method for solving the quadratic assignment problem,” Master’s Thesis, Dept. of Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, 1989.

18. M. Guignard and M. Rosenwein, “An improved dual-based algorithm for the generalized assignment problem,” Operations Research vol. 37, pp. 658-663, 1989.

19. M. Guignard, P.M. Hahn, and Z. Ding, “Hybrid ARQ symbol mapping in digital wireless communication systems based on the quadratic 3-dimensional assignment problem (Q3AP),” NSF Grant: DMI-0400155, 2004.

20. P.M. Hahn, and T.L. Grant, “Lower bounds for the quadratic assignment problem based upon a dual formulation,” Operations Research vol. 46, pp. 912-922, 1998.

21. P.M. Hahn, T.L. Grant and N. Hall, “A branch-and-bound algorithm for the quadratic assignment problem based on the Hungarian method,” European Journal of Operational Research vol. 108, pp. 629-640, 1998.

22. P.M. Hahn, W.L. Hightower, T.A. Johnson, M. Guignard-Spielberg and C. Roucairol, “Tree elaboration strategies in branch-and-bound algorithms for solving the quadratic assignment problem,” Yugoslav Journal of Operations Research vol. 11, pp. 41-60, 2001.

23. P.M. Hahn and J. Krarup, “A hospital facility problem finally solved,” The Journal of Intelligent Manufacturing vol. 12, pp. 487-496, 2001.

24. K. Jörnsten and M. Näsberg, “A new Lagrangean-relaxation approach to the generalized assignment problem,” European Journal of Operational Research vol. 27, pp. 313-323, 1986.

25. S.E. Karisch, E. Çela, J. Clausen, and T. Espersen, “A dual framework for lower bounds of the quadratic assignment problem based on linearization,” Computing vol. 63, pp. 351-403, 1999.

26. B-J. Kim, “Investigation of Methods for Solving New Classes of Quadratic Assignment Problems (QAPs),” PhD thesis, Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, May 2006.

27. M. Laguna, J. P. Kelly, J. L. González-Velarde and F. Glover, "Tabu search for the multilevel generalized assignment problem," European Journal of Operational Research vol. 82, pp. 176-189, 1995.

28. C-G. Lee, and Z. Ma, “The generalized quadratic assignment problem,” Research Report, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, M5S 3G8, Canada, 2004.

29. E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P.M. Hahn, and T. Querido, “A survey for the quadratic assignment problem”, Invited Review, European Journal of Operational Research, vol. 176, pp 657-690.

30. V. F. Magirou, “An improved partial solution to the task assignment and multiway cut problems,” Operations Research Letters vol. 12, pp. 3-10, 1992.

31. V. F. Magirou and J. Z. Milis, “An algorithm for the multiprocessor assignment problem,” Operations Research Letters vol. 8, pp. 351-356, 1989.

32. S. Martello and P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley, New York, NY, 1990.

33. K.G. Ramakrishnan, M.G.C. Resende, B. Ramachandran, and J.F. Pekny, “Tight QAP bounds via linear programming,” In: Combinatorial and Global Optimization, edited by P.M. Pardalos, A. Migdalas, and R.E. Burkard, World Scientific Publishing, Republic of Singapore, pp. 297-303, 2002.

34. F. Rendl and R. Sotirov, "Bounds for the quadratic assignment problem using the bundle method," accepted to Mathematical Programming (available at http://www.springerlink.com/content/a878n099r184510g/).

35. M.G.C. Resende, K.G. Ramakrishnan and Z. Drezner, “Computing lower bounds for the quadratic assignment with an interior point algorithm for linear programming,” Operations Research vol. 43, pp. 781-791, 1995.

36. H. Edwin Romeijn, Dolores Romero Morales, “A class of greedy algorithms for the generalized assignment problem,” Discrete Applied Mathematics vol. 103, pp. 209-235, 2000.

37. G.T. Ross and R.M. Soland, “A branch-and-bound algorithm for the generalized assignment problem,” Mathematical Programming vol. 8, pp. 91-103, 1975.

38. F. Roupin, “From linear to semi-definite programming: an algorithm to obtain semidefinite relaxations for bivalent quadratic problems,” Journal of Combinatorial Optimization vol. 8, pp. 469-493, 2004.

39. S. Sahni and T. Gonzalez, “P-complete approximation problems,” Journal Assoc. Computing Machinery vol. 23, pp. 555-565, 1976.

40. M. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem," Operations Research vol. 45, pp. 831–841, 1997.

41. H.D. Sherali and W.P. Adams, “A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems,” SIAM Journal on Discrete Mathematics vol. 3, pp. 411-430, 1990.

42. H.D. Sherali and W.P. Adams, “A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems,” Discrete Applied Mathematics vol. 52, pp. 83-106, 1994.

43. H.D. Sherali and W.P. Adams, A reformulation-linearization technique for solving discrete and continuous non-convex problems, 1st edition, Kluwer Academic Publishers, Norwell, MA 02061, 1999.

44. S. Sofianopoulous, "Simulated annealing applied to the process allocation problem," European Journal of Operational Research vol. 60, pp. 327-334, 1992.

45. S. Sofianopoulous, "The process allocation problem: A survey of the application of graph-theoretic and integer programming approaches," Journal of the Operational Research Society vol. 43, pp. 407-413, 1992.

46. M. Yagiura, T. Yamaguchi, T. Ibaraki, "A variable depth search algorithm with branching search for the generalized assignment problem," Optim. Methods Software vol. 10, pp. 419-441, 1998.

47. M. Yagiura, T. Yamaguchi, and T. Ibaraki, "A variable depth search algorithm with branching search for the generalized assignment problem," In: Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization edited by S. Martello, I.H. Osman, and C. Roucairol, Kluwer Academic Publishers, Boston, MA, pp. 459-471, 1999.