An Improved Algorithm for the Generalized Quadratic Assignment Problem

Artur Alves Pessoa
Production Engineering Department
Universidade Federal Fluminense, Brazil
artur@producao.uff.br

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

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

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

Abstract

This paper is concerned with solving GQAPs (generalized quadratic assignment problems) to optimality. In these problems, given *M* facilities and *N* locations, the facility space requirements, the location available space, the facility installation costs, the flows between facilities, and the distance costs between locations, one must assign each facility to exactly one location so that each location has sufficient space for all facilities assigned to it and the sum of the products of the facility flows by the corresponding distance costs plus the sum of the installation costs is minimized. This problem generalizes the well-known quadratic assignment problem (QAP), one of the most difficult problems in the combinatorial optimization field.

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

We propose a new lower bound for the GQAP based on a new Lagrangean relaxation of a known RLT (Reformulation Linearization Technique) formulation. In order to efficiently solve the relaxed problem, we combine a subgradient-type algorithm with a dual ascent procedure. We imbedded the new bounding procedure in a previously proposed branch-and-bound method, and tested it on instances from the literature. The resulting hybrid algorithm solved 18 out of 19 instances with up to 35 facilities, in at most a few days. Six of these instances were solved to optimality for the first time.

1. Introduction

1.1 Background and problem formulations

The generalized quadratic assignment problem (GQAP) studies a class of problems that optimally assign M facilities to N locations subject to the resource limitation at each location. These problems arise naturally in yard management, where containers are to be located in the storage areas with limited capacity, and in distributed computing where processing tasks are to be assigned to processors with limited computing resources. The GQAP is a generalization of the QAP, with the difference that multiple facilities can now be assigned to a single location subject to resource capacity at locations.

Lee and Ma (2004) proposed the first formulation of the GQAP. Their study involves a facility location problem in manufacturing where M facilities must be located among N fixed locations, with a space constraint at each possible location. The objective is to minimize the total installation and interaction transportation cost. The formulation of the GQAP is then

$$\min \quad \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{n=1}^{N} f_{ik} d_{jn} x_{ij} x_{kn} + \sum_{i=1}^{M}\sum_{j=1}^{N} b_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} s_{ij} x_{ij} \le S_j \qquad j = 1,\ldots,N, \tag{2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad i = 1,\ldots,M, \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad i = 1,\ldots,M;\ j = 1,\ldots,N, \tag{4}$$

with

M the number of facilities,

N the number of locations,

f_{ik} the commodity flow from facility i to facility k,

d_{jn} the distance from location j to location n,

b_{ij} the cost of installing facility i at location j,

s_{ij} the space requirement if facility i is installed at location j,

S_j the space available at location j,

x_{ij} a binary variable, with $x_{ij} = 1$ iff facility i is installed at location j.

The objective function (1) sums the costs of installation and quadratic interactivity. The knapsack constraints (2) impose space limitations at each location, and the multiple choice constraints (3) ensure that each facility is to be installed at exactly one location.

In the more general case of the GQAP, the quadratic costs are known, but are not necessarily decomposable as products of flows and distances. Also, as in the Lawler QAP model, the linear cost b_{ij} can be combined into the $M^2 \cdot N^2$ cost coefficients $c_{ijkn}\ \forall (i,k = 1,...,M;\ j,n = 1,...,N)$ since the 0-1 variables x_{ij} satisfy $x_{ij}^2 = x_{ij}$. The general GQAP formulation is then

$$\min \quad \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{M}\sum_{n=1}^{N} c_{ijkn} x_{ij} x_{kn}$$

s.t. (2), (3) and (4) hold.

Here, the solution matrix $\boldsymbol{X} = \left[x_{ij} \right]$ is an $M \times N$ matrix, which determines the "assignment" of each facility to a corresponding location.

1.2 Complexity and related problems

As mentioned earlier, the GQAP is a generalization of the GAP (Generalized Assignment Problem) as well as of the QAP. Ross and Soland (1975) gave the first formal definition of the

GAP. Fisher et al. (1986) showed that the GAP is an NP-hard combinatorial optimization problem. Since the first computational work by Ross and Soland (1975), which was a Lagrangean relaxation-based branch-and-bound algorithm , numerous heuristics and exact algorithms have been devised. Cattrysse and Van Wassenhove (1992) reviewed both exact and heuristic methods for solving the GAP. Most approaches were based on branch-and-bound techniques with bounds provided by heuristics and by relaxations of the knapsack constraints (2) or of the multiple choice constraints (3). Fisher et al. (1986) investigated the Lagrangean relaxation where constraints (3) are dualized with multipliers updated by a heuristic adjustment method. Guignard and Rosenwein (1989) tightened the above Lagrangean relaxation by devising a dual-ascent procedure that also added a surrogate constraint. Barcia and Jörnsten (1990) combined Lagrangean decomposition and a bound improvement sequence algorithm to solve the GAP. Savelsbergh (1997) employed a column generation-based branch-and-bound to generate optimal solutions to a set partitioning formulation of the GAP. Recent development on the exact solution methods includes Nauss (2003), and Haddadi and Ouzia (2004). Nauss (2003) presented a special-purpose branch-and-bound algorithm that utilizes linear programming cuts, feasible solution generators, Lagrangean relaxation, and subgradient optimization to solve hard GAP problems with up to 3000 binary variables. Haddadi and Ouzia (2004) provided a breadth-first branch-and-bound algorithm for the GAP with largest-upper-bound-next branching strategy. They applied a new heuristic at each iteration of the subgradient method to construct a feasible assignment by solving a smaller GAP. The feasible solution found was then subjected to a solution improvement heuristic. See Nauss (2006) for a recent survey.

Even though the GQAP was formulated only recently, special cases have been studied extensively. Related problems, other than the QAP and the GAP, include the multiprocessor assignment problem of Magirou et al. (1989), the task assignment and multiway cut problems of Magirou (1992), the process allocation problem of Sofianopoulou (1992a, 1992b), the constrained task assignment problem of Billionnet et al. (1995), the quadratic semi-assignment problem of Billionnet et al. (2001), the constrained module allocation problem of Elloumi et al.

(2003), the constrained-memory allocation problem of Roupin (2004), and the service allocation problem of Cordeau (2007).

1.3 Previous computational research

Previous studies on the GQAP are relatively limited. Lee and Ma (2004) presented three linearization approaches and a branch-and-bound algorithm to optimally solve the problem. In their branch-and-bound algorithm, the best feasible solution found by means of two greedy heuristics was taken as an initial upper bound for the branch-and-bound enumeration. Then, at each node of the search tree a lower bound was computed by a series of GAPs with the cost associated with the set of already assigned and yet unassigned facilities. They gave computational results for 26 test instances, the most difficult of which was of 16 facilities and 7 locations.

The other literature on the exact solution procedure for the GQAP is Hahn et al. (2006a). They presented a level-1 RLT formulation of the general version of the GQAP and a useful continuous Lagrangean relaxation. A dual-ascent procedure was then proposed to solve the Lagrangean relaxation so as to provide a lower bound in the branch-and-bound enumeration. Their exact solution method reported speed improvement on problem instances designed by Lee and Ma (2004), and was capable of optimally solving instances up to 20 facilities and 15 locations.

Cordeau et al. (2006) worked on a heuristic approach to obtain good suboptimal assignments using reasonable computing time. Their memetic heuristic was able to find optimal solutions for many instances. Kim (2006) developed a simulated annealing heuristic method with competitive performance. Finally the CHR approach of Ahlatcioglou and Guignard (2008) was able to obtain solutions comparable to those of Cordeau et al in seconds rather than minutes.

1.4 Our results

In this paper, we propose a new lower bound for the GQAP based on the new Lagrangean relaxation of the RLT formulation proposed in Hahn et al. (2006a). In order to efficiently solve the relaxed problem, we combine their dual ascent procedure with the general-purpose Volume Algorithm of Barahona and Anbil (2000). We tested the new bounding procedure using the same branch-and-bound method as that of Hahn et al. (2006a). The resulting hybrid algorithm proved the optimality of known solutions for 18 out of 19 instances selected from the literature. The largest solved instances have 35 facilities and 15 locations or 30 facilities and 20 locations, six of which could not be solved by the previous exact methods. All solved instances required at most a few days of running time.

2. Equivalent formulations and algorithms

2.1 RLT formulations

The Reformulation-Linearization Technique (RLT) of Adams and Sherali (1986 and 1990) and Sherali and Adams (1990, 1994, 1998 and 1999) is known for generating tight linear programming relaxations, not only for constructing exact solution algorithms, but also for designing powerful heuristic procedures for large classes of discrete combinatorial and continuous non-convex programming problems. The RLT provides different "levels" of representations that give increasing strength. Prior studies have shown that even the weakest level-1 form yields very tight bounds, which in turn lead to improved solution methodologies.

Now consider an $M^2 \times N^2$ solution matrix $\boldsymbol{Y}$, which is a Kronecker product of the $M \times N$ assignment matrix $\boldsymbol{X}$ with itself, that is:

$$\boldsymbol{Y} = \boldsymbol{X} \otimes \boldsymbol{X} = \begin{bmatrix} x_{11}\boldsymbol{X} & \cdots & x_{1N}\boldsymbol{X} \\ \vdots & \ddots & \vdots \\ x_{M1}\boldsymbol{X} & \cdots & x_{MN}\boldsymbol{X} \end{bmatrix} = \left[y_{ijkn} \right]_{M^2 \times N^2} \tag{5}$$

where $y_{ijkn} = x_{ij}x_{kn} = x_{kn}x_{ij} = y_{knij} \qquad \forall (i,k = 1,...,M;\ j,n = 1,...,N), k \neq i,$ (6)

$$y_{ijij} = x_{ij} \qquad \forall (i = 1,...,M;\ j = 1,...,N), \tag{7}$$

$$y_{ijin} = 0 \qquad \forall (i = 1,...,M;\ j,n = 1,...,N), n \neq j. \tag{8}$$

Here, (6)-(8) are indicated by the solution structure $\boldsymbol{X}$ of the GQAP and the definition of $\boldsymbol{Y}$. Also, the matrix $\boldsymbol{Y}$ is a partitioned matrix whose elements are composed of the Null matrix and matrix $\boldsymbol{X}$, and $\boldsymbol{Y}$ exhibits a gross pattern identical to that of the matrix $\boldsymbol{X}$. An example of a $\boldsymbol{Y}$ matrix and its corresponding $\boldsymbol{X}$ matrix for the GQAP of $M = 4$ and $N = 3$ are shown in Figure 1.

$$\boldsymbol{Y} = \left(\begin{array}{ccc|ccc|ccc}
\{1\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{0\} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\{1\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{0\} \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\{0\} & 0 & 0 & 0 & \{1\} & 0 & 0 & 0 & \{0\} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\{0\} & 0 & 0 & 0 & \{0\} & 0 & 0 & 0 & \{1\}
\end{array}\right) \quad \boldsymbol{X} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Figure 1
Example of a GQAP solution matrix $\boldsymbol{Y}$ and its $\boldsymbol{X}$ matrix.

Equation (6) gives the complementary pairs in the GQAP solution matrix $\boldsymbol{Y}$. If an element $y_{ijkn}\ (k \neq i)$ is part of a solution (i.e., equals to one) then its complementary element y_{knij} is also in that solution. These complementary pairs are always in two different submatrices

that never occupy the same submatrix row. However, the pair creates communication between submatrices that can be exploited in improving bounds. Elements defined by equation (8) are referred to as "disallowed elements", which are always being zero. Elements defined by equation (7) are termed "linear-cost elements" and are shown bracketed in Figure 1. A linear-cost element has no complementary element. Notice that if there is any 1's in a submatrix $x_{ij}\boldsymbol{X}$, its linear-cost element is always unity and vice versa. Notice, too, that one and only one unity element may exist in each row of solution matrix $\boldsymbol{Y}$.

Hahn et al. (2006a) used the following mixed-integer programming formulation for their RLT1 algorithm:

$$[\text{LIPR}] \qquad \min \quad \sum_{i=1}^{M}\sum_{j=1}^{N} b_{ij}x_{ij} + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{M}\sum_{n=1}^{N} c_{ijkn}y_{ijkn} \tag{9}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} s_{ij}x_{ij} \le S_j \qquad j=1,\ldots,N, \tag{2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad i=1,\ldots,M, \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad i=1,\ldots,M;\ j=1,\ldots,N, \tag{4}$$

$$\sum_{k=1}^{M} s_{kn}y_{ijkn} \le S_n x_{ij} \qquad i=1,\ldots,M;\ j,n=1,\ldots,N, \tag{10}$$

$$\sum_{n=1}^{N} y_{ijkn} = x_{ij} \qquad i,k=1,\ldots,M;\ j=1,\ldots,N;k\neq i, \tag{11}$$

$$y_{ijkn} = y_{knij} \qquad i,k=1,\ldots,M;\ j,n=1,\ldots,N;k>i, \tag{12}$$

$$y_{ijkn} \ge 0 \qquad i,k=1,\ldots,M;\ j,n=1,\ldots,N;k\neq i. \tag{13}$$

Next, we show that the continuous relaxation $\overline{\text{LIPR}}$ of the LIPR relaxation can be slightly improved using (7), that is, $y_{ijij} = x_{ij}$. Consider constraint (10).

For $n = j$, we get

. $$\sum_{k\neq i} s_{kj} y_{ijkj} + s_{ij} y_{ijij} \leq S_j x_{ij}$$

that is, $$\sum_{k\neq i} s_{kj} y_{ijkj} \leq (S_j - s_{ij}) x_{ij}$$

For $n \neq j$, we keep (10) unchanged.

We refer to the resulting strengthened formulation as LIPR+. It is identical with model LIPR, except that constraint (10) is replaced by (10-1) and (10-2) as follows:

$$\sum_{k=1}^{M} s_{kn} y_{ijkn} \leq S_n x_{ij} \qquad i=1,...,M;\ j,n=1,...,N;\ j\neq n, \qquad (10\text{-}1)$$

$$\sum_{\substack{k=1\\k\neq i}}^{M} s_{kj} y_{ijkj} \leq (S_j - s_{ij}) x_{ij} \qquad i=1,...,M;\ j=1,...,N. \qquad (10\text{-}2)$$

The strengthened model can now be written as

$$[\text{LIPR+}]\ \min \sum_{i=1}^{M}\sum_{j=1}^{N} b_{ij} x_{ij} + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{M}\sum_{n=1}^{N} c_{ijkn} y_{ijkn} \qquad (9)$$

s.t. $$\sum_{i=1}^{M} s_{ij} x_{ij} \leq S_j \qquad j=1,...,N, \qquad (2)$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad i=1,...,M, \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad i=1,...,M;\ j=1,...,N. \qquad (4)$$

$$\sum_{k=1}^{M} s_{kn} y_{ijkn} \leq S_n x_{ij} \qquad i=1,...,M;\ j,n=1,...,N,\ j\neq n, \qquad (10\text{-}1)$$

$$\sum_{\substack{k=1\\k\neq i}}^{M} s_{kj} y_{ijkj} \leq (S_j - s_{ij}) x_{ij} \qquad i=1,...,M;\ j=1,...,N, \qquad (10\text{-}2)$$

$$\sum_{n=1}^{N} y_{ijkn} = x_{ij} \qquad i,k=1,...,M;\ j=1,...,N;k\neq i, \qquad (11)$$

$$y_{ijkn} = y_{knij} \qquad i,k = 1,...,M;\ j,n = 1,...,N;k > i\,, \tag{12}$$

$$y_{ijkn} \geq 0 \qquad i,k = 1,...,M;\ j,n = 1,...,N;k \neq i\,. \tag{13}$$

2.2 Lagrangean relaxation strategy and its properties

In this section, we consider that all variables y_{ijkn} in the model LIPR+ are binary. The resulting integer programming model LIPR+ is highly degenerate. Of all $MN^2 + M(M-1)N + \frac{M^2N^2}{2} + M + N$ constraints, only the N knapsack constraints of (2) and M equality constraints of (3) have nonzero right-hand-side (RHS) values. The continuous relaxation $\overline{\text{LIPR+}}$ of LIPR+ will yield a lower bound $V\left(\overline{\text{LIPR+}}\right)$ on the optimal value of LIPR+, i.e., $V(\text{LIPR+})$, and thus on $V(\text{GQAP})$, where $V(P)$ is the optimal value of problem (*P*). A stronger bound may be obtained by Lagrangean relaxation if the Lagrangean subproblems do not have the Integrality Property (Geoffrion, 1974).

Consider dualizing constraints (3), (11) and (12), so that the remaining constraints are either over variables x_{ij} alone, or constraints over some y_{ijkn} variables with a single x_{ij} in the right hand side. The Lagrangean subproblem for given Lagrangean multipliers $\boldsymbol{\lambda} = \left[\lambda_{ijkn}\right]_{i,j,k,n;k>i}$, $\boldsymbol{\theta} = \left[\theta_{ijk}\right]_{i,j,k;k\neq i}$, and $\boldsymbol{\mu} = [\mu_i]_i$, is then

$\left[\text{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})\right]$

$$\begin{aligned} \text{Min} \quad & \sum_{i=1}^{M}\sum_{j=1}^{N} b_{ij}x_{ij} + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{M}\sum_{n=1}^{N} c_{ijkn}y_{ijkn} + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{\substack{k=1\\k>i}}^{M}\sum_{n=1}^{N} \lambda_{ijkn}\left(y_{knij} - y_{ijkn}\right) \\ & + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{M} \theta_{ijk}\left(x_{ij} - \sum_{n=1}^{N} y_{ijkn}\right) + \sum_{i=1}^{M}\mu_i\left(1 - \sum_{j=1}^{N} x_{ij}\right) \end{aligned} \tag{14}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} s_{ij}x_{ij} \leq S_j \qquad j = 1,...,N\,, \tag{2}$$

$$x_{ij} \in \{0,1\} \qquad i=1,...,M;\ j=1,...,N, \qquad (4)$$

$$\sum_{k=1}^{M} s_{kn} y_{ijkn} \le S_n x_{ij} \qquad i=1,...,M;\ j,n=1,...,N,\ j \ne n, \qquad (10\text{-}1)$$

$$\sum_{\substack{k=1 \\ k \ne i}}^{M} s_{kj} y_{ijkj} \le (S_j - s_{ij}) x_{ij} \qquad i=1,...,M;\ j=1,...,N, \qquad (10\text{-}2)$$

$$y_{ijkn} \in \{0,1\} \qquad i,k=1,...,M;\ j,n=1,...,N; k \ne i. \qquad (13\text{-}1)$$

The corresponding Lagrangean dual is therefore the optimization problem over the multipliers $\boldsymbol{\lambda}, \boldsymbol{\theta},$ and $\boldsymbol{\mu}$ given by

[LD]

$$\underset{\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu}}{\text{Max}} \ V\left[\text{LR}\left(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu}\right)\right].$$

Its optimal value $V(\text{LD})$ is the Lagrangean bound on $V\left(\text{LIPR+}\right) = V\left(\text{GQAP}\right)$. Problem $\text{LR}\left(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu}\right)$ does not have the Integrality Property, so that the Lagrangean bound may be better than the linear programming (LP) bound $V\left(\overline{\text{LIPR+}}\right)$, in the sense of being closer to $V\left(\text{LIPR+}\right)$.

One can now take advantage of the Integer Linearization Property (Geoffrion 1974, Geoffrion and McBride 1978, and Guignard 2003) to solve $\text{LR}\left(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu}\right)$ easily. Ignoring temporarily the constraints that are solely over x_{ij}, i.e., (2), one can see that the only true constraints remaining are (10-1) and (10-2), which we will refer to "globally" as (10'). The problem decomposes into one subproblem for each (i, j): indeed, each x_{ij}, whose value can only be 0 or 1, plays the role of a right-hand-side parameter for the corresponding subproblem. If x_{ij} is 0, all associated y_{ijkn} variables are also 0. If x_{ij} is 1, the constraints associated with x_{ij} separate into one for each *n*, i.e., there is no overlap between the variables y_{ijkn} of the *N*

corresponding constraints. The x_{ij}-problem then decomposes into N independent 0-1 knapsack problems, indexed by (i, j, n), which we will call $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta})^{ijn}$, with β_{ijn} =$V(\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta})^{ijn})$.

$\left[\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta})^{ijn}\right]$

$$\min \sum_{\substack{k=1\\k<i}}^{M}\left(c_{ijkn}+\lambda_{knij}-\theta_{ijk}\right)y_{ijkn}+\sum_{\substack{k=1\\k>i}}^{M}\left(c_{ijkn}-\lambda_{ijkn}-\theta_{ijk}\right)y_{ijkn} \tag{15}$$

s.t. (10') and (13-1),

then $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ decomposes into N subproblems as follows. For each $j=1,\ldots,N$, we must solve

$\left[\mathrm{KP}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})^{j}\right]$

$$\min \sum_{i=1}^{M}\left(b_{ij}+\sum_{n=1}^{N}\beta_{ijn}+\sum_{\substack{k=1\\k\neq i}}^{M}\theta_{ijk}-\mu_i\right)x_{ij} \tag{16}$$

$$\text{s.t.} \quad \sum_{i=1}^{M}s_{ij}x_{ij}\le S_j \tag{2}$$

$$x_{ij}\in\{0,1\} \qquad i=1,...,M\,. \tag{4}$$

The above problems are actually linear (0-1) knapsack problems (or KPs). Thus, the solution of $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ requires solving exactly $(M\times N^2+N)$ KPs, and the final bound is given by

$$V(\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu}))=\sum_{j=1}^{N}V(\mathrm{KP}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})^{j})+\sum_{i=1}^{M}\mu_i\,.$$

Experiments show that this bound is tighter than the linear programming bound $V\left(\overline{\mathrm{LIPR}}\right)$ in many cases.

2.3 The Level 1 RLT Dual Ascent Procedure

In this section, we briefly describe some operations performed by the RLT1 Dual Ascent Procedure proposed by Hahn et al. (2006a), which are also applied by our lower bounding procedure.

Consider now that $M^2 \times N^2$ quadratic cost coefficients C_{ijkn} $(i,k=1,...,M;\ j,n=1,...,N)$ are arranged in an $M^2 \times N^2$ matrix $\boldsymbol{C}$, similarly to matrix $\boldsymbol{Y}$ and indexed in precisely the same fashion. Initially, let us assume that $C_{ijkn} = c_{ijkn}$, for $i \neq k$, and $C_{ijij} = b_{ij}$. In this case, note that the cost of a feasible solution represented by $\boldsymbol{Y}$ is given by $Z(\boldsymbol{Y}) = tr\left(C \times \boldsymbol{Y}^T\right)$, where $tr(\boldsymbol{A})$ denotes the trace of matrix $\boldsymbol{A}$, that is, the sum of values in its principal diagonal.

The RLT1 Dual Ascent Procedure relies on the fact that certain operations may be performed on $\boldsymbol{C} = \{C_{ijkn}\}$ that will change the cost $Z(\boldsymbol{Y})$ of assignments $\boldsymbol{Y}$ in such a way that all assignment costs are shifted by an identical amount, thus preserving their ranking with respect to cost. Here, we are only interested in the operations described below, which are applied to each submatrix of $\boldsymbol{C}$:

Operation 1: For each cost element C_{ijkn} $(i \neq k)$ of the current submatrix, subtract a constant equal to $\sigma \times \max\{0, C_{knij}\}$ from its complementary pair C_{knij}, and add this constant to C_{ijkn}, where $\sigma \in (0,1]$ is a parameter for our algorithm that we define later.

Operation 2: For each row $[C_{ijk1},\ldots,C_{ijkN}]$ of the current submatrix $(i \neq k)$ having all its cost elements positive, subtract a constant equal to $\min_{n=1,\ldots,N}\{C_{ijkn}\}$ from each cost element, and add this constant to the corresponding linear cost C_{ijij}.

2.4 Combining the Dual Ascent Procedure and the solution of the Lagrangean dual

Now consider the Lagrangean subproblem $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ defined above. The Lagrangean dual problem is to find values for $\boldsymbol{\lambda},\boldsymbol{\theta},$ and $\boldsymbol{\mu}$ such that the value of $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ is maximized. Next, we show that the operations described in the previous section can be interpreted as changes in the values of corresponding Lagrangean multipliers in $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$. For that, let us redefine

$$C_{ijkn} = c_{ijkn} - \lambda_{ijkn} - \theta_{ijk}, \text{ for } i < k,$$
$$C_{ijkn} = c_{ijkn} + \lambda_{knij} - \theta_{ijk}, \text{ for } i > k,$$
$$C_{ijij} = b_{ij} + \sum_{\substack{k=1 \\ k \neq i}}^{M} \theta_{ijk} - \mu_i, \text{ and}$$
$$LB = \sum_{i=1}^{M} \mu_i.$$

Let also $\boldsymbol{Y}^*$ be an optimal solution[2] for $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$, for certain values of $\boldsymbol{\lambda},\boldsymbol{\theta},$ and $\boldsymbol{\mu}$. In this case, the optimum value of $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ is exactly to $LB + Z(\boldsymbol{Y}^*)$[3].

Note that the Lagrangean multipliers that compose the value of each C_{ijkn} allow for the same operations described in the previous section, that is, any of these operations can be obtained through a change in a corresponding Lagrangean multiplier. For instance, the transfer of a cost δ from the C_{knij} to its complementary pair C_{ijkn} can be obtained by adding δ to λ_{ijkn}. Moreover, an operation that subtracts a constant amount δ from the submatrix row $\left[C_{ijk1},...,C_{ijkN}\right]$ and adds it to the corresponding linear cost C_{ijij} can be obtained by adding δ to θ_{ijk}. Hence, the operations

[2] Note that, since the constraints (11), (12) and (3) are relaxed in $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$, $\boldsymbol{Y}^*$ may have more than one "1" in the same row.

[3] Remember that $Z(\boldsymbol{Y}) = tr(C \times \boldsymbol{Y}^T)$.

described in the previous section can always be applied to improve the current lower bound given by $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ during the optimization relative to $\boldsymbol{\lambda},\boldsymbol{\theta},$ and $\boldsymbol{\mu}$.

The operations described in the previous section, however, are usually not sufficient to give a good lower bound for the GQAP. To perform the full optimization relative to $\boldsymbol{\lambda},\boldsymbol{\theta},$ and $\boldsymbol{\mu}$, we use a general-purpose subgradient-type method: the Volume Algorithm proposed by Barahona and Anbil (2000). The operations described in the previous section are then applied to improve the performance of the lower bound calculation. Next, we briefly describe the Volume Algorithm in order to explain how these operations are introduced in the method.

For the sake of simplicity, we rewrite the three vectors of Lagrangean multipliers used in $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ as a single vector $\boldsymbol{\pi}=(\pi_1,\ldots,\pi_I)$, such that each multiplier of $\boldsymbol{\lambda},\boldsymbol{\theta},$ or $\boldsymbol{\mu}$ corresponds to exactly one component of $\boldsymbol{\pi}$. Thus $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})$ becomes just $\mathrm{LR}(\boldsymbol{\pi})$. In the same spirit, we represent the constraint that is associated to the multiplier π_i as the generic constraint $a_1^i z_1+\cdots+a_J^i z_J=b^i$, where a_j^i and b^i are constant coefficients and z_j is a binary variable (contained in matrix $\boldsymbol{Y}$), for $i=1,\ldots,I$ and $j=1,\ldots,J$. For a given value of $\boldsymbol{\pi}$, let $\bar{\boldsymbol{z}}=(\bar{z}_1,\ldots,\bar{z}_J)$ be the optimal solution of $\mathrm{LR}(\boldsymbol{\pi})$, and $\boldsymbol{s}=(s_1,\ldots,s_I)$ a vector of components s_i, with $s_i=b^i-(a_1^i\bar{z}_1+\cdots+a_J^i\bar{z}_J)$. In this case, $-\boldsymbol{s}$ is a subgradient for the function $-\mathrm{LR}(\boldsymbol{\pi})$, and the classical subgradient method adds $\boldsymbol{s}$ multiplied by a step size δ to $\boldsymbol{\pi}$ (i.e., $\boldsymbol{\pi}\leftarrow\boldsymbol{\pi}+\delta\boldsymbol{s}$) to try to improve the current lower bound. Then, the value of δ is decreased at each iteration until a good lower bound is found. On the other hand, the Volume Algorithm uses a convex combination $\tilde{\boldsymbol{z}}$ of all values of $\bar{\boldsymbol{z}}$ obtained so far to calculate a *fake* subgradient $-\boldsymbol{s}$ in the same way (just replace $\bar{\boldsymbol{z}}$ by $\tilde{\boldsymbol{z}}$ in the definition of **s**). After each iteration, $\tilde{\boldsymbol{z}}$ is updated based on the current value of $\bar{\boldsymbol{z}}$ (e.g. $\tilde{\boldsymbol{z}}\leftarrow(1-\alpha)\tilde{\boldsymbol{z}}+\alpha\bar{\boldsymbol{z}}$). Moreover, the step size δ is calculated as a function of $\|\boldsymbol{s}\|$ and $(LB-UB)$, and multiplied by an auxiliary value γ (see Barahona and Anbil (2000) for more details), where $\|\boldsymbol{s}\|$ denotes the L_2 norm of **s**, and *LB* and *UB* are the current

upper and lower bounds, respectively (in our implementation, we maintain the initial values of *UB* and *LB* fixed because changes in these values can cause premature convergence of the method). The value of γ is initially set as γ_0 and decreased after every *K* iterations without improvement on the lower bound value. Whenever the lower bound is not improved by the Lagrangean step, the vector $\boldsymbol{\pi}$ is restored to its previous value. If an improvement is found, then the value of γ may be increased depending on the angle between $-\boldsymbol{s}$ and the true subgradient that corresponds to $\boldsymbol{\pi}$.

The most important difference between the original Volume Algorithm and our implementation is that we apply the operations described in Section 2.4 on the current modified cost matrix $\boldsymbol{C}$ upon every lower bound improvement. As we have mentioned before, the operations performed by this procedure can be translated into changes on the value of $\boldsymbol{\pi}$. Our experiments show that the combination of these two kinds of dual improvements results in a much faster convergence of the whole method.

A unique and very important aspect of the subgradient optimization dual-ascent lower bound 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 subgradient optimization dual ascent calculation, a high quality lower bound is economically calculated that is ideally suited for use in a branch-and-bound algorithm. The lower bound calculation 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.

2.5 Implementation details

In this section, we give some implementation details of our lower bounding procedure that played an important role on achieving the reported results.

First, we discuss how the knapsack subproblems introduced in Subsection 2.2 are solved. When solving $\mathrm{KP}(\boldsymbol{\lambda},\boldsymbol{\theta},\boldsymbol{\mu})^j$, for $j=1,\ldots,N$, we use a standard dynamic programming algorithm that spends $O(MS_j)$ time for each subproblem. However, running time of the whole bounding procedure is dominated by the solutions of the $M \times N^2$ subproblems of type $\mathrm{LR}(\boldsymbol{\lambda},\boldsymbol{\theta})^{ijn}$. For these subproblems, we used the following improvements:

1. Only consider the items such that the corresponding Lagrangean cost $C_{ijkn} < 0$ (otherwise the item can be removed from the knapsack).
2. If the number of remaining items is sufficiently small, then solve the subproblem via explicit enumeration of all subsets of items (this avoids the S_j factor of the dynamic programming time complexity).

Another important issue is finding good parameter values for both the Dual Ascent Procedure and the Volume Algorithm. We have used the same parameters in all experiments. For the Dual Ascent, we set σ = 0.7 (see Subsection 2.3 for the definition of σ). For our implementation of the Volume Algorithm, which is described in Subsection 2.4, we set γ_0 = 1, K = 20, and α = 0.01. We multiply the value of γ by 0.66 when it must decrease, and by 1.1 when it must increase.

3. Experimentation

In order to test out the effectiveness of the new lower bounding method described above, we have implemented the lower bound calculation in a C++ program. Using the C++ code, the procedure was first tested on a variety of GQAP test cases. In those tests, the effectiveness of the lower bounds was clearly demonstrated and for certain GQAP instances, problems were solved with only a root lower bound calculation.

For a lower bounding method to be effective, it must meet four very important requirements. First, the bounds must be accurate. Second, they must be tight. Third, they must be calculated reasonably fast. And most importantly, these bounds must work in a branch-and-

bound algorithm, in that for all instances that are enumerated using the algorithm, at least one optimum solution must be found.

Accordingly, we tested our new lower bounding technique in the branch-and-bound algorithm we developed several years ago for the RLT1 dual-ascent lower bounding technique reported in Hahn, et al. (2006a). This branch-and-bound algorithm was written in the FORTRAN programming language. The algorithm uses a unique look-ahead branching strategy (see Hahn, et. al. 2001) that considerably reduces the amount of tree search nodes that have to be evaluated. Rather than replacing the RLT1 dual-ascent lower bound calculation with the newer subgradient optimization dual-ascent calculation, we employed both calculations in the branch-and-bound code. The older, weaker RLT1 calculation is tried first at each node of the branch-and-bound tree. If the RLT1 calculation does not fathom the node, then the newer subgradient optimization bound is attempted. Combining weak and strong bounds in this manner has proven very effective with other branch-and-bound combinatorial optimization problems (See Adams, et al. 2007). In all experiments, we started our algorithm with the best upper bound found in the literature by a heuristic. This means that the total running time of our hybrid branch-and-bound algorithm would be the time reported in Table 1 plus the sum of the running times of the three heuristics mentioned in Subsection 1.3 (required to find the initial upper bound used). However, note that these additional running times are always negligible when compared to the corresponding branch-and-bound running times.

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

				RLT1 subtraction bound		New bound	
GQAP instance	Size	Optimum value	No. of optima	Root bound Blade1000 time	Published time/nodes RLT1_B&B	Root bound Ultra 45 time	Ultra45 time/nodes Hybrid B&B
Elloumi c2005De	20x5	5435	3	0 0.02 secs	17,640 secs 20,498,196 nodes	4710 43 secs	10,085 secs 2,615 nodes
Elloumi 2005Aa	20x5	3059	1	784 0.1 sec	136 secs 211,447 nodes	2480 23 secs	3,018 secs 885 nodes
Elloumi 2408Ca	24x8	1028	2	924 0.3 sec	6 secs 1,880 nodes	1028** 78 secs	117.secs 89 nodes
Elloumi 2408Aa	24x8	5643	1	1037 0.1 sec	719,862 secs 226,961,852 nodes	4270 106 secs	1,031,637 secs 125.730
CGL&M 20-15-35	20x15	1471896	1	921444 0.4 secs	735 sec 54,496 nodes	1366752 271 secs	2,729 secs 130 nodes
CGL&M 20-15-55	20x15	1723638	1	1185376 6.9 sec	456 secs*** 84,704 nodes	1588923 149 secs	4,370 secs 499 nodes
CGL&M 20-15-75	20x15	1953188	1	1281377 6.3 secs	811 secs*** 169,791 nodes	1886034 122 secs	1,340 secs 176 nodes
CGL&M 30-08-55	30x8	3501695	1	2910778 0.2 secs	835 secs*** 565,560 nodes	3473219 234 secs	2,826 secs 97 nodes
CGL&M 30-20-35	30x20	3379359	1	2269193 34 secs	658,009 secs*** 8,380,859 nodes	2977357 1,077 secs	228,177 secs 3,628 nodes
CGL&M 30-20-55	30x20	3593105	1	2332618 82 secs	N/A	3205303 886 secs	707,027 secs 16,651 nodes
CGL&M 30-20-75	30x20	4050938	1	2419929 29.2 secs	N/A	3900811 933 secs	18,803 secs 528 nodes
CGL&M 30-20-95	30x20	5710645	1	2471615 28.1 secs	N/A	5710645** 2,221 secs	2,612 secs 269 nodes
CGL&M 35-15-35	35x15	4456670	1	2810012 4.41 secs	N/A	3932796 1,046 secs	978,664 secs 20,231 nodes
CGL&M 35-15-55	35x15	4639128	1	3064974 60.1 secs	N/A	4284169 1,030 secs	413,241 secs 7,472 nodes
CGL&M 35-15-75	35x15	6301723 ?	N/A	3647704 56.1 secs	N/A	5675819 1,254 secs	N/A
CGL&M 35-15-95	35x15	6670264	1	3642354 56.2 secs	N/A	6429128 2,319 secs	205,931 secs 1,645 nodes
Lee & Ma 14x9-E-G	14x9	2326370	1	1024570 0.1 sec	169 secs 207,666 nodes	2235605 25 secs	139 secs 68 nodes
Lee & Ma 15x8	15x8	2856850	1	1193007 0.68 secs	130 secs 103,893 nodes	2697989 59 secs	359 secs 82 nodes
Lee & Ma 16x7-1115	16x7	2809870	1	817500 0.04 secs	548 secs 540,733 nodes	2716594 74 secs	311 secs 74 nodes

N/A = not available **optimum value ***Blade 1000 ? best-known solution

Table 1 summarizes the results of our experimentation with the new hybrid (C++/FORTRAN) subgradient optimization dual-ascent branch-and-bound algorithm on certain classical problem instances. The Table also provides a comparison with earlier efforts to solve these same instances. Specifically, Columns 5 and 6 contain earlier results from experiments with the RLT1 subtraction-only methods based on the RLT1 dual-ascent lower bounding method. Column 5 contains the root lower bound values and runtimes of RLT1 dual ascent made on a single 900 MHz CPU of Sun Blade 1000 workstation. Column 6 contains the branch-and-bound runtimes on those problem instances listed that could be solved exactly with the RLT1 subtraction-only bound code. In column 6, published values of runtimes and tree search node counts are used, wherever they are available. Otherwise, Sun Blade 1000 runtimes and node counts are presented. All published runtimes are from the paper by Hahn, et al. (2006a) and were recorded on a single CPU of a single 733 MHz CPU of a Dell 7150 PowerEdge server. Column 7 contains the root bound calculated by the hybrid algorithm and the number of seconds required for the root bound calculation on a single 1.6 GHz CPU of an Ultra 45 Sun workstation. Column 8 contains the branch-and-bound runtime of the hybrid algorithm (in seconds) on a single 1.6 GHz CPU of an Ultra 45 Sun workstation and the number of nodes of the search tree that must be evaluated for the complete branch-and-bound search.

The problem instances in Table 1 were chosen for two reasons. First, every instance mentioned in the paper by Hahn, et al. was included. Second, six very difficult instances that had never been solved were included. One additional instance from the Cordeau, et al. (2006) paper (CGL&M 35-15-75) that could not be solved in less than two weeks by our new branch-and-bound algorithm was also included in Table 1.

The improvement in root lower bounds is dramatic. The average gap between root lower bound and optimum value for the 19 instances listed is 47% of optimum value for the earlier RLT1 subtraction only method, whereas, for the new subgradient optimization dual ascent method it is only 8 percent. This compares favorably to the 9 percent average gap value for 27

unsolved instances of the QAP from size 30 to size 256 listed on QAPLIB. Even though the QAP is also NP hard, it is easier to solve than the GQAP.

The most impressive results are the ability of the new algorithm to solve large GQAP instances that were unsolvable by existing methods and the relatively small node count required for their exact solution. None of the problems solved required more than 12 days to solve on a single CPU of a modern workstation. We are confident that with some more experimentation and tuning of parameters, all the runtimes experienced can be significantly improved.

4. Conclusion

This paper concentrates on solving GQAP instances via a strong RLT-1 remodelization, called (LIPR+). The hybrid method proposed combines a dual ascent approach for computing lower bounds based on the LP relaxation of the (LIPR+) model (Hahn and Grant, 1998) and a Lagrangean relaxation that makes efficient use of the integer linearization property in its modeling phase and of the volume algorithm in its solution phase. All operations performed by the hybrid algorithm can be viewed as operations on the Lagrangean dual variables. The hybrid method is shown to outperform either of the two basic algorithms. It can be imbedded very efficiently in a branch-and-bound code, and the excellent numerical results presented in the paper show that this new method can solve much harder problems than was previously possible.

The lower bound improvement provided by embedding a dual ascent method in a subgradient-modification algorithm, here the volume algorithm, is a significant algorithmic progress for the GQAP. Similar techniques should apply equally well to other difficult problems for which dual ascent steps permit the immediate improvement of the Lagrangean objective function value. Problems that come to mind immediately are the 3-dimensional Assignment Problem (Y.-R. Zhu, 2007), the Quadratic 3-dimensional Assignment Problem (Hahn et al., 2006b) and the Generalized Quadratic 3-dimensional Assignment Problem (Y.-R. Zhu, 2007).

Acknowledgements: This material is based upon work supported by the U.S. National Science Foundation under grant No. DMI-0400155. AAP was partially financed by CNPq grants 301175/2006-3.

Bibliography

Adams, W.P., and T.A. Johnson, 1994, Improved linear programming-based lower bounds for the quadratic assignment problem, in: Pardalos, P.M., and H. Wolkowicz (Eds.), Quadratic Assignment and Related Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. **16,** AMS, Providence, RI, 43-75.

Adams, W.P., and H.D. Sherali, 1986, A tight linearization and an algorithm for zero-one quadratic programming problems, Management Science **32**, 1274-1290.

Adams, W.P., and H.D. Sherali, 1990, Linearization strategies for a class of zero-one mixed integer programming problems, Operations Research **38**, 217-226.

Adams, W.P., M. Guignard, P.M. Hahn, and W.L. Hightower, 2007, A level-2 reformulation-linearization technique bound for the quadratic assignment problem, European Journal of Operational Research **180**, 983-996.

Ahlatcioglu, A., and M. Guignard, 2007, The convex hull relaxation for nonlinear integer programs with linear constraints, OPIM Department Report, revised, 2008.

Barahona, F., and R. Anbil, 2000, The volume algorithm: Producing primal solutions with a subgradient method, Mathematical Programming A **87**, pp? .

Barcia, P., and K. Jörnsten, 1990, Improved Lagrangean decomposition: an application to the generalized assignment problem, European Journal of Operational Research **46**, 84-92.

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

Billionnet, A., and S. Elloumi, 2001, Best reduction of the quadratic semi-assignment problem, Discrete Applied Mathematics **109**, 197-213.

Cattrysse, D.G., and L.N. Van Wassenhove, 1992, A survey of algorithms for the generalized assignment problem, European Journal of Operational Research **60**, 260-272.

Cordeau, J.-F., M. Gaudioso, G. Laporte, and L. Moccia, 2006, A memetic heuristic for the generalized assignment problem, INFORMS Journal on Computing **18**, 433-443.

Cordeau, J.-F., M. Gaudioso, G. Laporte, and L. Moccia, 2007, The service allocation problem at the Gioia Tauro maritime terminal, European Journal of Operational Research **176**, 1167-1184.

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

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

Geoffrion, A.M., 1974, Lagrangian relaxation and its uses in integer programming, Mathematical Programming Study **2**, 82-114.

Geoffrion, A.M., and R. McBride, 1978, Lagrangean relaxation applied to capacitated facility location problems, AIIE Transactions **10**, 40-47.

Graham, A., 1981, Kronecker Products and Matrix Calculus with Applications, Ellis Horwood Limited, Chichester, 21-23.

Guignard, M., and M.B. Rosenwein, 1989, An improved dual based algorithm for the generalized assignment problem, Operations Research **37**, 658-663.

Guignard, M., 2003, Lagrangean relaxation, TOP, Vol. **11**, 151-228.

Guignard, M., 2006, Note on the solution of the Lagrangean subproblems for an RLT-1 form of the GQAP, Research Report 06-06-01, Department of Operations and Information Management, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104.

Guignard, M., P.M. Hahn, Z. Ding, B.-J. Kim, and Y.-R. Zhu, 2006, Extensions and variations on quadratic assignment problems including the quadratic 3-dimensional assignment problem (Q3AP), Proceedings of 2006 NSF Design, Service, and Manufacturing Grantees and Research Conference, St. Louis, MO.

Haddadi, S., and H. Ouzia, 2004, Effective algorithm and heuristic for the generalized assignment problem, European Journal of Operational Research **153**, 184-190.

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

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

Hahn, P.M., B.-J. Kim, M. Guignard, J.M. Smith, and Y.-R. Zhu, 2006a, An algorithm for the generalized quadratic assignment problem, An algorithm for the generalized quadratic assignment problem, Computational Optimization and Applications, http://www.springerlink.com/content/665775m470t56v34/.

Hahn, P.M., B.-J. Kim, T. Stützle, S. Kanthak, W.L. Hightower, H. Samra, Z. Ding, and M. Guignard, 2006b, The quadratic three-dimensional assignment problem: exact and approximate solution methods, European Journal of Operational Research, **184**, (2), 416-428.

Kim, B.-J., 2006, Investigation of methods for solving new classes of quadratic assignment problems (QAPs), Ph.D. Dissertation, University of Pennsylvania, Philadelphia, PA 19104.

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

Loiola, E.M., N.M.M. de Abreu, P.O. Boaventura-Netto, P.M. Hahn, and T. Querido, 2006, A survey for the quadratic assignment problem, to appear in European Journal of Operational Research.

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

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

Meller, R.D., and Y.A. Bozer, 1997, Alternative approaches to solve the multi-floor facility layout problem, Journal of Manufacturing Systems **16**, 192-203.

Nauss, R.M., 2003, Solving the generalized assignment problem: an optimizing and heuristic approach, INFORMS Journal on Computing **15**, 249-266.

Nauss, R.M., 2006, The generalized assignment problem, in: Karlof, J.K, (Ed.), Integer Programming: Theory and Practice, CRC Press, 39-55.

Ross, G.T., and R.M. Soland, 1975, A branch-and-bound algorithm for the generalized assignment problem, Mathematical Programming **8**, 91–103.

Roupin, F., 2004, From linear to semidefinite programming: an algorithm to obtain semidefinite relaxations for bivalent quadratic problems, Journal of Combinatorial Optimization **8**, 469-493.

Savelsbergh, M., 1997, A branch-and-price algorithm for the generalized assignment problem, Operations Research **45**, 831-841.

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

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

Sherali, H.D., and W.P. Adams, 1998, Reformulation-linearization techniques for discrete optimization problems, in: Du, D.-Z., and P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Vol. **1**, Kluwer Academic Publishers, Dordrecht, 479-532.

Sherali, H.D., and W.P. Adams, 1999, A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Sofianopoulous, S., 1992a, Simulated annealing applied to the process allocation problem, European Journal of Operational Research **60**, 327-334.

Sofianopoulous, S., 1992b, The process allocation problem: a survey of the application of graph-theoretic and integer programming approaches, Journal of the Operational Research Society **43**, 407-413.

Y.-R. Zhu, 2007, "Recent advances and challenges in quadratic assignment and related problems," Ph.D. dissertation, ESE Department, University of Pennsylvania, December 2007.