Effective formulation reductions for the quadratic assignment problem

Huizhen Zhang* Cesar Beltran-Royo† Miguel Constantino‡

11/12/2008

Abstract

In this paper we study two formulation reductions for the quadratic assignment problem (QAP). In particular we apply these reductions to the well known Adams and Johnson [2] integer linear programming formulation of the QAP, which we call formulation IPQAP-I. We analyze two cases: In the first case, we study the effect of *constraint reduction.* In the second case, we study the effect of *variable reduction* in the case of a sparse cost matrix. Computational experiments with a set of 32 QAPLIB instances, which range from 12 to 32 locations, are presented. The proposed reductions turned out to be very effective: By applying the new constraint reduction or the new variable reduction to the IPQAP-I formulation, we solved 13 and 23 instances, respectively, compared to the 7 instances solved by formulation IPQAP-I.

Key words: quadratic assignment problem, linear integer programming, linear programming relaxation, sparse matrix.

1. Introduction

The quadratic assignment problem(QAP) was first proposed by Koopmans and Beckmann in 1957 as a mathematical model related to the location of a set of indivisible economical activities [26]. Consider the problem of assigning n facilities to n locations in such a way that each facility is designated to exactly one location and vice-versa. The objective is to minimize the quadratic interaction cost, a function of the distances and flows between the

*zhzzywz@gmail.com, School of Management, University of Shanghai for Science and Technology, Shanghai, China

†Corresponding author: cesar.beltran@urjc.es, Statistics and Operations Research, Rey Juan Carlos University, Madrid, Spain.

‡miguel.constantino@fc.ul.pt, DEIO/ Centro de Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, Lisbon, Portugal

facilities, plus the costs associated with allocating a facility to a certain location. Therefore, given three $n \times n$ matrices with real elements $F = (f_{ik})$, $D = (d_{jl})$ and $C = (c_{ij})$, where f_{ik} is the flow between the facility i and facility k, d_{jl} is the distance between the location j and l, and c_{ij} is the cost of allocating facility i at location j, the QAP can be stated as follows:

$$\min_{x \in X} \sum_{i,j,k,l=1}^{n} q_{ijkl} x_{ij} x_{kl} + \sum_{i,j=1}^{n} c_{ij} x_{ij}, \tag{1.1}$$

where $q_{ijkl} = f_{ik} d_{jl}$,

$$x_{ij} = \begin{cases} 1 & \text{if facility } i \text{ is assigned to location } j, \\ 0 & \text{otherwise,} \end{cases}$$

and X is the set of permutation matrices of dimension n.

This set of permutations can be defined as:

$$X = \Big\{ x \mid \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i \in N \tag{1.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j \in N \tag{1.3}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in N \Big\} \tag{1.4}$$

where $N = \{1, \ldots, n\}$.

Lawler [27] considered a more general QAP, where the q_{ijkl} coefficients in (1.1) are not restricted to flow-distance products, in contrast with the original Koopman-Beckmann formulation.

The QAP has drawn researcher's attention worldwide and extensive research has been done for more than half century. The QAP problem is considered one of the most difficult combinatorial problems: it is *NP*-hard, and even finding an ε-approximate solution is a hard problem [37]. It is surprising the number of fields where the QAP problem can be applied. In addition to its application in facility location, the QAP has been applied in many fields such as printed circuit board assembly process [17], typewriter keyboards and control panels design [34], scheduling [21], numerical analysis [8], and many others. Moreover, many well-known classical combinatorial optimization problems such as the traveling salesman problem, the graph partitioning problem, the maximum clique problem, can also be formulated as special cases of the QAP, see [33] for details.

The advances in theoretical aspects, solution techniques and applications of the QAP have been discussed in more detail, for example, in [10, 5, 13, 30]. Regarding recent QAP advances, it is worth it to mention that, during the last years some of the most challenging QAP instances have been solved by combining parallel branch-and-bound algorithms [31, 15] with grid computing [4].

Calculation of lower bounds is an essential component of exact QAP methods, which employ implicit enumeration in a branch and bound framework, see [18, 24, 35]. On the other hand, lower bounds are used to evaluate the quality of solutions produced by heuristic algorithms, like simulated annealing algorithms [14, 32], genetic algorithms(GA)

[3, 16], greedy randomized adaptive search procedure(GRASP) [29], ant colony algorithms [20], and so on. Different QAP bounds have been proposed: Gilmore-Lawler bound, eigenvalue bounds, quadratic programming bounds, LP bounds, polyhedral bounds, semidefinite bounds, among others. More details about QAP lower bounds can be found in [38, 1, 28, 6, 23, 39].

In this paper we have two main objectives. The first objective is to study the effect of constraint reduction in (linear) integer programming QAP formulations (IPQAP). In the second objective we study the effect of variable and constraint reduction in the case of some null flows (sparse flow matrix).

With the first objective in mind, we present a new LP bound for the QAP, which is more effective than previous LP bounds. It is known that LP and dual-LP bounds are tight for the QAP, but appear to be computationally prohibitive in many cases [5]. To develop the new LP bound, our starting point is the IPQAP formulation of Adams and Johnson [2], that we name *IPQAP-I*. Then, by virtually dividing by two the number of its constraints, we propose an equivalent formulation, that we name *IPQAP-II*. This new formulation is less tight than *IPQAP-I*, but its LP relaxation can be solved much faster. The final result, is that formulation *IPQAP-II* usually requires a B&B tree with more nodes but more efficient in terms of total solving time.

Regarding the second objective, we have observed that quite a lot of QAP instances have a sparse flow matrix. As far as we know, this fact has not been exploited yet in literature. We study how one can exploit those zero flows. The key point is that in presence of one single zero flow, say $f_{i_0k_0}$, many coefficient costs $q_{i_0jk_0l}$ become also zero. We will show how the associated variables $y_{i_0jk_0l}$ can be eliminated in the QAP formulation. We name *IPQAP-III* and *IPQAP-IV* this reduced variable version of formulations *IPQAP-II* and *IPQAP-I*, respectively.

In our numerical experiments, we have used a set of 32 QAPLIB instances [9], that range from 12 to 32 locations, all of them with a sparse flow matrix. The results obtained by the new formulations have been surprisingly remarkable, especially, if we take into account that we have conducted our tests with a standard laptop, CPLEX 9.0 with default parameters and 4 hours of CPU time limit. Within these conditions and by using the IPQAP formulations I, II, III and IV, we have solved up to optimality 7, 13, 21 and 23 QAPLIB instances, respectively.

This paper is organized as follows. In Section 2, we review the Adams and Johnson QAP linearization. In section 3 we study the constraint reduction. In section 4 and 5, we study some variable reductions, especially in the case of some null flows (sparse flow matrix). The numerical experiments are presented in section 6. Concluding remarks are made in the last section.

2. Adams and Johnson linearization

Adams and Johnson [2] linearization is the well-known linear integer programming QAP formulation:

$$\min_{x,y} \quad \sum_{i,j,k,l=1}^{n} q_{ijkl}\ y_{ijkl} + \sum_{i,j=1}^{n} c_{ij} x_{ij} \tag{2.5}$$

$$\text{s. t.} \quad \sum_{l=1}^{n} y_{ijkl} = x_{ij} \qquad i,j,k \in N \tag{2.6}$$

$$\sum_{k=1}^{n} y_{ijkl} = x_{ij} \qquad i,j,l \in N \tag{2.7}$$

$$y_{ijkl} = y_{klij} \qquad i,j,k,l \in N \tag{2.8}$$

$$y_{ijkl} \in \{0,1\} \qquad i,j,k,l \in N \tag{2.9}$$

$$x \in X \tag{2.10}$$

This formulation, that we name *IPQAP-I*, contains $o(n^4)$ variables and $o(n^4)$ constraints. Although, it produces tight LP bounds, usually it poses an obstacle for efficiently solving QAP instances from medium to large scale. Even to solve the associated LP relaxation can be difficult [36]. Other QAP linearizations can be found in literature: Lawler's linearization [22] as the first one, Kaufmann and Broeckx's linearization [25] has the smallest number of variables and constraints and Frieze and Yadegar's linearization [19], among others.

3. Formulation reduction by constraint elimination

In this section we introduce the new formulation *IPQAP-II*, which corresponds to formulation *IPQAP-I* without constraints (2.7), and with half of the constraints (2.6) relaxed into the $\leq$ form, that is:

$$\min_{x,y} \quad \sum_{i,j,k,l=1}^{n} q_{ijkl}\ y_{ijkl} + \sum_{i,j=1}^{n} c_{ij} x_{ij} \tag{3.11}$$

$$\text{s. t.} \quad \sum_{l=1}^{n} y_{ijkl} = x_{ij} \qquad i,j,k \in N,\ i \leq k \tag{3.12}$$

$$\sum_{l=1}^{n} y_{ijkl} \leq x_{ij} \qquad i,j,k \in N,\ i > k \tag{3.13}$$

$$y_{ijkl} = y_{klij} \qquad i,j,k,l \in N \tag{3.14}$$

$$y_{ijkl} \in \{0,1\} \qquad i,j,k,l \in N \tag{3.15}$$

$$x \in X \tag{3.16}$$

Proposition 3.1 *IPQAP-II is a (valid) formulation for the QAP.*

Proof: We consider the following equivalent formulation of the QAP in the (x, y) space (we name it *QAP'*):

$$\begin{array}{lll} \min\limits_{x,y} & \sum\limits_{i,j,k,l=1}^{n} q_{ijkl}\ y_{ijkl} + \sum\limits_{i,j=1}^{n} c_{ij}x_{ij} & \\ \text{s. t.} & x_{ij}x_{kl} = y_{ijkl} & i,j,k \in N \\ & x \in X & \end{array}$$

Let us name $F_{QAP'}, F_I$ and F_{II} the feasible sets of formulations *QAP'*, *IPQAP-I* and *IPQAP-II*, respectively. To prove this proposition, it is enough to prove that $F_{QAP'} = F_{II}$, since *QAP'* and *IPQAP-II* have the same objective function.

First, let us see that $F_{QAP'} \subset F_{II}$. We know that *QAP'* and *IPQAP-I* are equivalent formulations and that *IPQAP-II* is a relaxation of *IPQAP-I*. Therefore,

$$F_{QAP} = F_I \subset F_{II}.$$

Second, let us see that $F_{II} \subset F_{QAP'}$. We consider $(x, y) \in F_{II}$ and will prove that $(x, y) \in F_{QAP'}$. Since $x \in X$, it is enough to see that $y_{ijkl} = x_{ij}\ x_{kl}$ for $1 \leq i, j, k, l \leq n$. Without loss of generality, we assume that $i \leq k$.

We analyze the three possible cases:

1. If $x_{ij}\ x_{kl} = 0$, with $x_{ij} = 0$ then $\sum_{l=1}^{n} y_{ijkl} = x_{ij} = 0$ by equation (3.12). This implies $y_{ijkl} = 0$.

2. If $x_{ij}\ x_{kl} = 0$, with $x_{kl} = 0$, we distinguish two subcases:

   (a) If $i < k$, then $\sum_{j=1}^{n} y_{klij} \leq x_{kl} = 0$ by equation (3.13).

   (b) If $i = k$, then $\sum_{j=1}^{n} y_{klij} = x_{kl} = 0$ by equation (3.12).

   In both subcases $y_{klij} = 0$ and therefore $y_{ijkl} = 0$.

3. If $x_{ij}\ x_{kl} = 1$ then $x_{ij} = x_{kl} = 1$. Let us suppose that $y_{ijkl} = 0$ and get a contradiction. If $y_{ijkl} = 0$, considering that $\sum_{l=1}^{n} y_{ijkl} = x_{ij} = 1$, there must exist $l' \in N$ with $l' \neq l$ such that $y_{ijkl'} = 1$. We distinguish two subcases:

   (a) If $i < k$, then $1 = y_{ijkl'} = y_{kl'ij} \leq \sum_{j=1}^{n} y_{kl'ij} \leq x_{kl'} \leq 1$, by equation (3.13).

   (b) If $i = k$, then $1 = y_{ijkl'} = y_{kl'ij} = \sum_{j=1}^{n} y_{kl'ij} = x_{kl'} \leq 1$ by equation (3.12).

   In both cases $x_{kl'} = 1$ $(l' \neq l)$, which contradicts our hypothesis $x_{kl} = 1$ together with $\sum_{l=1}^{n} x_{kl} = 1$. ∎

Given an Integer Programming formulation *IPQAP*, we denote by *LPQAP* its Linear Programming (LP) relaxation, i.e., the formulation obtained by replacing the integrality constraints $y_{ijkl} \in \{0,1\}$ for $i,j,k,l \in N$ and $x_{ij} \in \{0,1\}$ for $i,j \in N$ by their continuous counterparts, $0 \le y_{ijkl} \le 1$ for $i,j,k,l \in N$ and $0 \le x_{ij} \le 1$ for $i,j \in N$ respectively.

In general, the optimal value of the LP relaxation *LPQAP-I* is larger than the optimal value of *LPQAP-II*, that is, the first formulation is tighter. Nevertheless, *LPQAP-II* has fewer constraints, so it is easier to solve.

4. Formulation reduction in the case of some null flows

In this section we assume that the quadratic cost coefficients are proportional to the flow and to the distance, that is, $q_{ijkl} = f_{ik}d_{jl}$. In many QAP instances, there are pairs of facilities i_0k_0 whose flow is zero ($f_{i_0k_0} = 0$). In this case, the associated quadratic cost coefficient $q_{i_0jk_0l}$ vanishes for all locations j and l. We call *zero flow variables* the associated variables $y_{i_0jk_0l}$. From the objective function point of view it is irrelevant the value of a zero flow variable. One would like to fix them, say, to zero and thus reduce the number of variables. In this context, when we say that we eliminate one variable, we mean that we fix it to zero. However, as we will see in the following proposition, the zero flow variables cannot be eliminated, since they are relevant from the feasible set point of view (in fact any y_{ijkl} is relevant).

Proposition 4.2 *In formulations IPQAP-I and IPQAP-II one cannot eliminate any zero flow variable y_{ijkl} ($i \neq k, j \neq l$).*

Proof: (By contradiction.) In formulation *IPQAP-I*, let us take $x^0 \in X$ and assume that $x^0_{i_0j_0} = x^0_{k_0l_0} = 1 (i_0 \neq k_0, j_0 \neq l_0)$, and $f_{i_0k_0} = 0$. If we eliminate variable $y_{i_0j_0k_0l_0}$, by constraint (2.6), we have (without loss of generality, we assume that $i_0 < k_0$):

$$\sum_{l=1,\ l \neq l_0}^{n} y_{i_0j_0k_0l} = x^0_{i_0j_0} = 1.$$

Therefore, there must exist $l_1 \neq l_0$ such that $y_{i_0j_0k_0l_1} = 1$. Now, by constraints (2.7) and (2.8) we have:

$$1 = y_{i_0j_0k_0l_1} = y_{k_0l_1i_0j_0} \le \sum_{i=1}^{n} y_{k_0l_1ij_0} = x^0_{k_0l_1} \le 1.$$

Thus, $x^0_{k_0l_1} = 1$ with $l_1 \neq l_0$, which contradicts our hypothesis $x^0_{k_0l_0} = 1$ for $x^0 \in X$.

Similarly, it can be shown that in formulation *IPQAP-II* one cannot eliminate any zero flow variable y_{ijkl} ($i \neq k, j \neq l$) either. ■

In this proposition, we have seen that, in order to obtain an equivalent QAP formulation, it is not possible to simply eliminate the zero flow variables in formulations *IPQAP-I* and *IPQAP-II*. It turns out that, if we eliminate zero flow variables, we also need to eliminate, what we call *zero flow constraints* (constraints with at least one eliminated zero flow variable). This is done in formulation *IPQAP-III*, where we reduce formulation *IPQAP-II*, by eliminating the zero flow variables and the zero flow constraints. To state *IPQAP-III* we need to define first the index set of zero flows, i.e.,

$$F_0 = \{(i,k) \in N \times N \mid f_{ik} = 0\}.$$

We assume that the flow matrix is symmetrical and therefore if $(i,k) \in F_0$ then $(k,i) \in F_0$ as well.

We state the *IPQAP-III* formulation as follows:

$$\min_{x,y} \quad \sum_{i,j,k,l=1,\ ik \notin F_0}^{n} q_{ijkl}\ y_{ijkl} + \sum_{i,j=1}^{n} c_{ij} x_{ij} \tag{4.17}$$

$$\text{s. t.} \quad \sum_{l=1}^{n} y_{ijkl} = x_{ij} \qquad i,j,k \in N,\ i \leq k,\ ik \notin F_0 \tag{4.18}$$

$$\sum_{l=1}^{n} y_{ijkl} \leq x_{ij} \qquad i,j,k \in N,\ i > k,\ ik \notin F_0 \tag{4.19}$$

$$y_{ijkl} = y_{klij} \qquad i,j,k,l \in N,\ ik \notin F_0 \tag{4.20}$$

$$y_{ijkl} \in \{0,1\} \qquad i,j,k,l \in N,\ ik \notin F_0 \tag{4.21}$$

$$x \in X \tag{4.22}$$

Proposition 4.3 *Formulation IPQAP-III is equivalent to IPQAP-II.*

Proof: Let F_{II} and F_{III} be the feasible sets of *IPQAP-II* and *IPQAP-III*, respectively. Let V_{II} and V_{III} be the sets of feasible objective values of formulations *IPQAP-II* and *IPQAP-III*, respectively. To prove this proposition, we will show that $V_{II} = V_{III}$, and therefore the minimum of both formulations is the same. Furthermore, we will also show how to compute an optimal *IPQAP-II* solution once we have an optimal *IPQAP-III* solution and vice versa.

First, let us show $V_{III} \subset V_{II}$. In fact, we prove something stronger: any solution $(x,y) \in F_{III}$ can be extended to a solution $(\widetilde{x}, \widetilde{y}, \bar{y}) \in F_{II}$ with the same objective value ($\bar{y}$ correspond to the coordinates y_{ijkl} with $(i,k) \in F_0$ and $j,l \in N$).

The procedure is as follows: define $\widetilde{x} = x$ and $\widetilde{y} = y$. To complete the extended solution $(\widetilde{x}, \widetilde{y}, \bar{y})$, the value of the removed variables $\bar{y}_{ijkl}$ for $(i,k) \in F_0$ and $i,j \in N$, is given by $\bar{y}_{ijkl} = x_{ij} \times x_{kl}$. Observe that we have $\sum_{l=1}^{n} y_{ijkl} = \sum_{l=1}^{n} x_{ij} x_{kl} = x_{ij} \sum_{l=1}^{n} x_{kl} = x_{ij}$, the last equality follows from (1.2). Hence $(\widetilde{x}, \widetilde{y}, \bar{y})$ satisfies constraints (3.12-3.16), so it is feasible for *IPQAP-II*. Furthermore, $(\widetilde{x}, \widetilde{y}, \bar{y})$ and (x,y) have the same objective function value since $q_{ijkl} = 0$ for all $(i,k) \in F_0$ and $j,l \in N$, Second, let us show $V_{III} \supset V_{II}$. In fact, we prove something stronger: given any solution $(\widetilde{x}, \widetilde{y}, \bar{y}) \in F_{II}$ ($\bar{y}$ correspond to the coordinates $ijkl$ with $(i,k) \in F_0$ and $j,l \in N$), it can be projected to a solution $(x,y) \in F_{III}$ with the same objective value. The procedure is as follows: define $x = \widetilde{x}$ and $y = \widetilde{y}$. Obviously, (x,y) is feasible for *IPQAP-III* and as before, (x,y) and $(\widetilde{x}, \widetilde{y}, \bar{y})$ have the same objective value. ∎

We would obtain a valid statement if in the previous Proposition we replace *IPQAP-II* and *IPQAP-III* by *LPQAP-II* and *LPQAP-III* respectively. In fact, the above proof remains valid in this case, as the integrality of the variables is never used. Hence we may conclude that the LP relaxations *LPQAP-II* and *LPQAP-III* are equivalent.

Similarly, we can eliminate the zero flow variables and the zero flow constraints in the formulation *IPQAP-I*. The reduced *IPQAP-I* is named as *IPQAP-IV*:

$$\min_{x,y} \quad \sum_{i,j,k,l=1,\ ik \notin F_0}^{n} q_{ijkl}\ y_{ijkl} + \sum_{i,j=1}^{n} c_{ij} x_{ij} \tag{4.23}$$

$$\text{s. t.} \quad \sum_{l=1}^{n} y_{ijkl} = x_{ij} \qquad i,j,k \in N,\ ik \notin F_0 \tag{4.24}$$

$$\sum_{k=1}^{n} y_{ijkl} = x_{ij} \qquad i,j,l \in N, \quad i \notin I_0 \tag{4.25}$$

$$y_{ijkl} = y_{klij} \qquad i,j,k,l \in N,\ ik \notin F_0 \tag{4.26}$$

$$y_{ijkl} \in \{0,1\} \qquad i,j,k,l \in N,\ ik \notin F_0 \tag{4.27}$$

$$x \in X \tag{4.28}$$

where I_0 is the the index set: $I_0 = \{i \in N\ | \exists k : f_{ik} = 0\}$. Furthermore, we can get the following proposition:

Proposition 4.4 *IPQAP-IV is a (valid) formulation for the QAP.*

Proof: Observe that *IPQAP-IV* is the same as *IPQAP-III* with additional constraints (4.25). From Propositions 3.1 and 4.3 it follows that these constraints are satisfied by the feasible set of *IPQAP-III*. Hence the feasible sets of *IPQAP-IV* and *IPQAP-III* are equal. Since the objective function is the same and *IPQAP-III* is a valid formulation for the QAP, the result follows. ∎

The above proof is not valid if we consider the linear relaxations *LPQAP-IV* and *LPQAP-III* instead of *IPQAP-IV* and *IPQAP-III*, because constraints (4.25) may not be redundant to *LPQAP-III*. Let $\bar{V}_I - \bar{V}_{IV}$ denote the optimal values of $IPQAP{-}I - IPQAP{-}IV$ respectively. We have the following relationship between the LP relaxations:

$$\bar{V}_{III} = \bar{V}_{II} \leq \bar{V}_{IV} \leq \bar{V}_I,$$

and there are instances for which the inequalities are strict (See section 6).

5. Further formulation reductions

In formulations IPQAP, I to IV, we can eliminate many variables, especially we can use only one of the duplicated variables $y_{ijkl} = y_{klij}$. We start by reducing the number of variables in formulation *IPQAP-II*. The symmetry constraints (3.14) imply that we can formulate *IPQAP-II* by using only variables y_{ijkl}, with $i \leq k$.

The elimination of the duplicated variables y_{ijkl} $(i > k)$ can be done as follows:

- In the objective function, substitute all variables y_{ijkl} with $i > k$ by y_{klij}.
- In constraint (3.13) substitute y_{ijkl} with $i > k$ by y_{klij}

$$\sum_{l=1}^{n} y_{klij} \leq x_{ij} \quad i, j, k \in N, \;\; i > k,$$

  and rename the $klij$ indexes as $ijkl$

$$\sum_{j=1}^{n} y_{ijkl} \leq x_{kl} \quad i, k, l \in N, \;\; i < k,$$

- Eliminate the symmetry constraints (3.14).

With the above variable reduction, we get an equivalent formulation of *IPQAP-II* that we call *IPQAP-II'*:

$$\min_{x,y} \quad \sum_{1 \leq i \leq k \leq n} \; \sum_{1 \leq j,l \leq n} q_{ijkl} \; y_{ijkl} + \sum_{1 \leq k < i \leq n} \; \sum_{1 \leq j,l \leq n} q_{klij} \; y_{ijkl} + \sum_{i,j=1}^{n} c_{ij} \; x_{ij} \qquad (5.29)$$

$$\text{s. t.} \quad \sum_{l=1}^{n} y_{ijkl} = x_{ij} \qquad i, j, k \in N, \;\; i \leq k \qquad (5.30)$$

$$\sum_{j=1}^{n} y_{ijkl} \leq x_{kl} \qquad i, k, l = 1 \in N, \;\; i < k \qquad (5.31)$$

$$y_{ijkl} \in \{0, 1\} \qquad i, j, k, l \in N, \;\; i \leq k \qquad (5.32)$$

$$x \in X \qquad (5.33)$$

Further variable reduction can be done in *IPQAP-II'* for variables y_{ijkl} with $i = k$ or $j = l$:

1. Case $i = k$ and $j = l$. Considering that for any (x, y) feasible for *IPQAP-II'*, we have $y_{ijij} = x_{ij}$ for all ij:
   (a) In the objective function, substitute all variables y_{ijij} by x_{ij}.
   (b) In constraints (5.30) substitute all variables y_{ijij} by x_{ij}.
2. Case $i = k$ and $j \neq l$. Considering that $y_{ijil} = 0$ for all i and $j \neq l$.
   (a) In the objective function eliminate the terms $ijil$ for all i and $j \neq l$.

(b) In constraints (5.30), eliminate the terms $ijil$ for all i and $j \neq l$. Note that, this term elimination together with substitution 1.(b) implies that in (5.30) we can eliminate all the equations with $i = k$.

3. Case $i \neq k$ and $j = l$. Considering that $y_{ijkj} = 0$ for all j and $i \neq k$.

   (a) In the objective function eliminate the terms $ijkj$ for all j and $i \neq k$.

   (b) In constraints (5.30-5.31), eliminate the terms $ijkj$ for all j and $i \neq k$.

4. Eliminate variables y_{ijil} and y_{ijkj}.

With the above variable reduction, we get an equivalent *reduced* formulation of *IPQAP-II'*, which we call *IPQAPR-II*:

$$\min_{x,y} \quad \sum_{1 \leq i < k \leq n} \; \sum_{1 \leq j \neq l \leq n} \widetilde{q}_{ijkl} \; y_{ijkl} + \sum_{i,j=1}^{n} p_{ij} \; x_{ij} \tag{5.34}$$

$$\text{s. t.} \quad \sum_{l=1,\; l \neq j}^{n} y_{ijkl} = x_{ij} \qquad i, j, k \in N, \; i < k \tag{5.35}$$

$$\sum_{j=1,\; j \neq l}^{n} y_{ijkl} \leq x_{kl} \qquad i, k, l \in N, \; i < k \tag{5.36}$$

$$y_{ijkl} \in \{0, 1\} \qquad i, j, k, l \in N, \; i < k, \; l \neq j \tag{5.37}$$

$$x \in X \tag{5.38}$$

where $\widetilde{q}_{ijkl} = q_{ijkl} + q_{klij}$ and $p_{ij} = c_{ij} + q_{ijij}$.

In the past years, some pioneer QAP researchers performed similar variable reductions in formulation *IPQAP-I*, to obtain what we call formulation *IPQAPR-I* (see [5]):

$$\min_{x,y} \quad \sum_{1 \leq i < k \leq n} \; \sum_{1 \leq j \neq l \leq n} \widetilde{q}_{ijkl} \; y_{ijkl} + \sum_{i,j=1}^{n} p_{ij} x_{ij} \tag{5.39}$$

$$\text{s. t.} \quad \sum_{l=1,\; l \neq j}^{n} y_{ijkl} = x_{ij} \qquad i, j, k \in N, \; i < k \tag{5.40}$$

$$\sum_{j=1,\; j \neq l}^{n} y_{ijkl} = x_{kl} \qquad i, k, l \in N, \; i < k \tag{5.41}$$

$$\sum_{k=1}^{i-1} y_{klij} + \sum_{k=i+1}^{n} y_{ijkl} = x_{ij} \qquad i, j, l \in N, \quad j \neq l \tag{5.42}$$

$$y_{ijkl} \in \{0, 1\} \qquad i, j, k, l \in N \textit{ and } k > i \tag{5.43}$$

$$x \in X \tag{5.44}$$

Finally, by performing the above variable reductions to formulation *IPQAP-III* and *IPQAP-IV*, we obtain what we call formulation *IPQAPR-III* and *IPQAPR-IV*, respectively:

IPQAPR-III :

$$\min_{x,y} \quad \sum_{1\le i<k\le n,\ ik\notin F_0} \quad \sum_{1\le j\ne l\le n} \widetilde{q}_{ijkl}\ y_{ijkl} + \sum_{i,j=1}^{n} p_{ij}\ x_{ij} \tag{5.45}$$

$$\text{s. t.} \quad \sum_{l=1,\ l\ne j}^{n} y_{ijkl} = x_{ij} \qquad i,j,k\in N,\ i<k,\ ik\notin F_0 \tag{5.46}$$

$$\sum_{j=1,\ j\ne l}^{n} y_{ijkl} \le x_{kl} \qquad i,k,l\in N,\ i<k,\ ik\notin F_0 \tag{5.47}$$

$$y_{ijkl}\in\{0,1\} \qquad i,j,k,l\in N,\ i<k,\ l\ne j,\ ik\notin F_0 \tag{5.48}$$

$$x\in X \tag{5.49}$$

IPQAPR-IV:

$$\min_{x,y} \quad \sum_{1\le i<k\le n,\ ik\notin F_0} \quad \sum_{1\le j\ne l\le n} \widetilde{q}_{ijkl}\ y_{ijkl} + \sum_{i,j=1}^{n} p_{ij}\ x_{ij} \tag{5.50}$$

$$\text{s. t.} \quad \sum_{l=1,\ l\ne j}^{n} y_{ijkl} = x_{ij} \qquad i,j,k\in N,\ i<k,\ ik\notin F_0 \tag{5.51}$$

$$\sum_{j=1,\ j\ne l}^{n} y_{ijkl} = x_{kl} \qquad i,k,l\in N,\ i<k,\ ik\notin F_0 \tag{5.52}$$

$$\sum_{k=1}^{i-1} y_{klij} + \sum_{k=i+1}^{n} y_{ijkl} = x_{ij} \qquad i,j,l\in N,\quad j\ne l,\quad i\notin I_0 \tag{5.53}$$

$$y_{ijkl}\in\{0,1\} \qquad i,j,k,l\in N\quad k>i,\quad j\ne l,\quad ik\notin F_0 \tag{5.54}$$

$$x\in X \tag{5.55}$$

6. Numerical experiments

In this section, we present the experimental results obtained with the formulations *IPQAPR-I*, *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV*, applied to some sparse flow-input matrices instances from QAPLIB [9]. The experiments were conducted on a laptop with an Intel Pentium M-1.70GHz processor and with 512 MB RAM. Matlab 7.0 was used to set up the formulations and Cplex 9.0 was used to solve the resulting IP instances. Cplex default parameters were used throughout. The program was allowed to run for 4 hours at most. The set up time (Matlab) and the IP solving time (Cplex) are reported.

In table 1 we summarize some characteristics of the instances used in the tests, namely the *density* of the flow-input matrices (DFM), which is defined as the percent of the number of non-zero elements in the matrix. We also report the dimension of each formulation *IPQAPR-I*, *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV*, namely the number of variables x_{ij} (Nb of x) and y_{ijkl} (Nb of y), and the number of constraints (Nb of constraints).

In table 2 we present the value of the objective function of the best integer solution (Cost) obtained for each problem instance, as well as the gap measured by Cplex directly. The gap is given by the formula $Gap = \frac{Cost-bestnode}{Cost} \times 100\%$, where *bestnode* denotes the lower bound obtained by Cplex.

In table 3 we present the linear programming relaxation values ($LPcost$), gaps, $LPgap = \frac{optv-LPcost}{optv} \times 100\%$, where *optv* is the optimal objective value, and the CPU times spent in solving the linear programming relaxation of every tested instances.

The corresponding CPU times for setting up the formulations and solving the mixed integer linear programs are shown in table 4. In this table we also present the CPU time given in the literature for some of the tested instances, with special purpose exact algorithms. Unfortunately, we didn't find the CPU time for solving the instances Esc32a - Esc32d, Esc32g and Esc32h with exact algorithms, so they are absent in the table 4. For the instances, Chr12a-Chr25a, Esc16a-Esc16j, Esc32e and Esc32f, the literature CPU times are obtained in [11], [12] and [7], respectively.

We make the following observation remarks regarding the computational results presented in tables 1- 4 :

- As expected, formulation *IPQAPR-II* has considerably fewer constraints than formulation *IPQAPR-I* or *IPQAP-I*. The reduced formulations *IPQAPR-IV* and *IPQAPR-III* have the same variables, and in most cases they have the same number of constraints, except for instance Esc16h, as shown in table 1. In all but this instance, $I_0 = N$, that is, for all $i \in N$ there exists k such that the flow f_{ik} is null.

- The linear relaxation bounds obtained with formulation *IPQAPR-I* are in general better than the ones obtained with formulations *IPQAPR-II*, as expected. Nevertheless, the integer programming solutions obtained with *IPQAPR-II*, shown in table 2, are better or at least with equal value than the solutions obtained by *IPQAPR-I*. On the other hand the number of nodes in the Branch and Bound tree is larger for *IPQAPR-II*. Although the LP relaxation is weaker, it can be solved faster, so more nodes can be searched, leading to a superior performance of *IPQAPR-II* over *IPQAPR-I*. For the 32 tested instances, 7 and 20 optimal solutions are computed with formulations *IPQAPR-I* and *IPQAPR-II* respectively, and for 7 and 13 of them optimality has been proved.

- When we look at the results obtained with *IPQAPR-III* and *IPQAPR-IV* we see that these formulations outperform *IPQAPR-I* and *IPQAPR-II*. Here, a smaller number of variables makes the LP relaxations much easier to solve, allowing an even larger Branch and Bound tree within the time limits. For those instances solved to optimality, the LP CPU times as shown in table 3 and the IP CPU times shown in table 4 are smaller for formulations *IPQAPR-III* and *IPQAPR-IV*. For the 32 tested instances, 26 and 27 optimal solutions are computed with formulations *IPQAPR-III* and *IPQAPR-IV* respectively, and for 21 and 23 of them optimality has been proved.

- The results obtained with *IPQAPR-III* and *IPQAPR-IV* are very similar. As it was mentioned before, $I_0 = N$ for all instances except for Esc16h. This means these two formulations only differ on constraints (5.47) and (5.52). Observe that for all instances except for Esc16h the LP bounds are the same.

From these observations, we can assert that the formulation *IPQAPR-I* has a tighter LP bound than the formulations *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV*. Although the LP relaxation of the formulation *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV* may be not so tight, the reduced number of variables and constraints make them much easier to solve. Even though more Branch and Bound nodes are needed, the instances are solved in a shorter time, or better solutions are obtained within the time limit.

Regardless that *IPQAPR-II* and *IPQAPR-III* are LP-equivalent, and the linear programming relaxation values of *IPQAPR-IV* are worse than *IPQAPR-I*, it is clear that a lot of time and resources are saved by eliminating the zero flow variables and constraints to reduce the dimensions of the problems in the formulation *IPQAPR-III* and *IPQAPR-IV*. For example, there are absolutely zero flows between 12 of the facilities in the flow matrix of the instance Esc32c. Thus, for any placement of the remaining 20 facilities in the 32 locations, there are 12!=476,001,600 solutions with exactly the same value. This multiplicity of solutions is avoided in *IPQAPR-III* and *IPQAPR-IV*.

7. Concluding remarks

In this paper we presented Integer Programming Formulations obtained by eliminating some variables and/or constraints from the QAP formulation of Adams and Johnson [2]. Although the reduced formulations are less tight, they have considerably fewer variables and constraints, so the LP relaxations are much easier to solve. As a result, using *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV* proved to be overall more effective in solving a set of instances from the literature.

The effectiveness of some of the QAP linearizations depends strongly on the sparsity of the cost matrix. When the cost coefficients are given by the product of a

Table 1: Dimensions of the QAP tested instances implemented with the formulations *IPQAPR-I*, *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV*

Ins.	n	DFM (%)	No. of x	No. of y		No. of constraints			
				R-I[a]/R-II	R-III/R-IV	R-I	R-IV	R-II	R-III
Chr12a	12	15.28	144	8712	1452	3192	288	1608	288
Chr12b	12	15.28	144	8712	1452	3192	288	1608	288
Chr12c	12	15.28	144	8712	1452	3192	288	1608	288
Chr15a	15	12.44	225	22050	2940	6330	450	3180	450
Chr15b	15	12.44	225	22050	2940	6330	450	3180	450
Chr15c	15	12.44	225	22050	2940	6330	450	3180	450
Chr18a	18	10.49	324	46818	5202	11052	648	5544	648
Chr18b	18	10.49	324	46818	5202	11052	648	5544	648
Chr20a	20	9.50	400	72200	7220	15240	800	7640	800
Chr20b	20	9.50	400	72200	7220	15240	800	7640	800
Chr20c	20	9.50	400	72200	7220	15240	800	7640	800
Chr22a	22	8.68	484	106722	9702	20372	968	10208	968
Chr22b	22	8.68	484	106722	9702	20372	968	10208	968
Chr25a	25	7.68	625	180000	14400	30050	1250	15050	1250
P.ave.[b]	17.6	11.3	324.9	56854.7	5646.0	11941.7	649.7	5988.4	649.7
Esc16a	16	29.69	256	28800	9120	7712	1248	3872	1248
Esc16b	16	71.88	256	28800	22080	7712	2976	3872	2976
Esc16c	16	39.84	256	28800	12240	7712	1664	3872	1664
Esc16d	16	16.41	256	28800	5040	7712	704	3872	704
Esc16e	16	16.41	256	28800	5040	7712	704	3872	704
Esc16f	16	0.00	256	28800	0	7712	32	3872	32
Esc16g	16	16.41	256	28800	5040	7712	704	3872	704
Esc16h	16	89.84	256	28800	27600	7712	6112	3872	3712
Esc16i	16	11.72	256	28800	3600	7712	512	3872	512
Esc16j	16	9.38	256	28800	2880	7712	416	3872	416
P.ave.	16	30.2	256.0	28800.0	9264.0	7712.0	1507.2	3872.0	1267.2
Esc32a	32	14.45	1024	492032	73408	63552	4800	31808	4800
Esc32b	32	21.09	1024	492032	107136	63552	6976	31808	6976
Esc32c	32	25.59	1024	492032	129952	63552	8448	31808	8448
Esc32d	32	17.58	1024	492032	89280	63552	5824	31808	5824
Esc32e	32	1.17	1024	492032	5952	63552	448	31808	448
Esc32f	32	1.17	1024	492032	5952	63552	448	31808	448
Esc32g	32	1.76	1024	492032	8928	63552	640	31808	640
Esc32h	32	27.54	1024	492032	139872	63552	9088	31808	9088
P.ave.	32.0	13.8	1024.0	492032.0	70060.0	63552.0	4584.0	31808.0	4584.0
T.ave.[c]	20.7	17.8	478.0	156882.0	22880.0	23523.0	1901.0	11782.0	1826.0

[a]In these tables, R-I, R-II, R-III and R-IV signify IPQAPR-I, IPQAPR-II, IPQAPR-III and IPQAPR-IV, respectively.

[b]In this paper, P. ave. is the abbreviation for partial average.

[c]T. ave. is the abbreviation for total average.

Table 2: (Mixed) Integer programming results for the tested QAP instances implemented with the formulations *IPQAPR-I*, *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV* (-: no integer feasible solution found in the limited running time)

Ins.	Opt. or B.known Sol.	(Mixed) IP								No. of nodes			
		R-I		R-IV		R-II		R-III					
		Cost	B&B Gap(%)	Cost	B&B Gap(%)	Cost	B&B Gap(%)	Cost	B&B Gap(%)	R-I	R-IV	R-II	R-III
Chr12a	9552	9552	0.0	9552	0.0	9552	0.0	9552	0.0	1	9	741	36
Chr12b	9742	9742	0.0	9742	0.0	9742	0.0	9742	0.0	1	120	1295	21
Chr12c	11156	11156	0.0	11156	0.0	11156	0.0	11156	0.0	1	60	889	100
Chr15a	9896	9896	0.0	9896	0.0	9896	0.0	9896	0.0	15	60	3997	127
Chr15b	7990	7990	0.0	7990	0.0	7990	0.0	7990	0.0	1	138	2341	588
Chr15c	9504	9504	0.0	9504	0.0	9504	0.0	9504	0.0	1	1	1	1
Chr18a	11098	-	-	11098	0.0	11098	0.0	11098	0.0	-	58	941	71
Chr18b	1534	-	-	1534	0.0	1534	0.0	1534	0.0	-	4	13	8
Chr20a	2192	-	-	2192	0.0	2192	0.0	2192	0.0	-	414	1168	413
Chr20b	2298	-	-	2298	0.0	2298	0.0	2298	0.0	-	655	598	1431
Chr20c	14142	-	-	14142	0.0	15100	39.1	14142	0.0	-	2110	4401	17708
Chr22a	6156	-	-	6156	0.0	6156	0.0	6156	0.0	-	505	1374	230
Chr22b	6194	-	-	6194	0.0	6194	0.0	6194	0.0	-	72	605	230
Chr25a	3796	-	-	3796	0.0	5432	39.7	3796	0.0	-	1872	441	5164
P.ave.	7517.86	-	-	7518	0.0	7703.14	5.6	7518	0.0	-	434	1343	1866
Esc16a	68	100	52.0	68	14.2	68	100.0	68	35.5	1	156136	5822	155689
Esc16b	292	314	11.5	292	97.3	294	100.0	292	100.0	1	9276	936	1590
Esc16c	160	-	-	160	77.5	162	95.1	162	77.5	-	67056	8120	55170
Esc16d	16	38	89.5	16	0.0	16	100.0	16	0.0	1	260927	1071	92691
Esc16e	28	52	73.1	28	0.0	28	100.0	28	0.0	1	42637	3748	51465
Esc16f	0	0	0.0	0	0.0	0	0.0	0	0.0	1	1	1	1
Esc16g	26	46	69.6	26	0.0	26	76.9	26	0.0	1	10001	2847	9601
Esc16h	996	-	-	996	30.7	996	99.8	996	99.7	-	1	10625	11881
Esc16i	14	54	100.0	14	0.0	14	100.0	14	0.0	1	4072	3631	5211
Esc16j	8	26	92.3	8	0.0	8	62.5	8	0.0	1	691	2935	418
P.ave.	160.80	-	-	161	22.0	161.20	83.4	161.00	31.3	-	55080	3974	38372
Esc32a	130(B[a])	-	-	194	100.0	-	-	244	100.0	-	91	-	140
Esc32b	168(B)	-	-	300	100.0	-	-	400	100.0	-	21	-	17
Esc32c	642(B)	-	-	736	100.0	-	-	716	100.0	-	26	-	31
Esc32d	200(B)	-	-	236	100.0	-	-	256	100.0	-	71	-	136
Esc32e	2	-	-	2	0.0	-	-	2	100.0	-	367	-	3643725
Esc32f	2	-	-	2	0.0	-	-	2	100.0	-	367	-	3643577
Esc32g	6	-	-	6	0.0	-	-	6	0.0	-	7742	-	9064
Esc32h	438(B)	-	-	532	100.0	-	-	534	100.0	-	15	-	34
P.ave.	198.50	-	-	251	62.5	-	-	270	87.5	-	1088	-	912091
T.ave.	3388.94	-	-	3402	22.5	-	-	3407	31.7	-	17674	-	240830

[a]B means the value is the best known solution.

Table 3: Linear programming relaxation results for the tested QAP instances implemented with the formulations *IPQAPR-I*, *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV*

Ins.	LP relaxation								LP CPU Time (Sec.)			
	R-I		R-IV		R-II		R-III					
	LPcost	LPgap (%)	LPcost	LPgap (%)	LPcost	LPgap (%)	LPcost	LPgap (%)	R-I	R-IV	R-II	R-III
Chr12a	9552.0	0.0	8593.1	10.0	8593.1	10.0	8593.1	10.0	19.8	0.0	0.8	0.0
Chr12b	9742.0	0.0	7184.0	26.3	7184.0	26.3	7184.0	26.3	15.9	0.0	0.9	0.0
Chr12c	11156.0	0.0	10042.7	10.0	10042.7	10.0	10042.7	10.0	33.4	0.0	0.8	0.0
Chr15a	9513.1	3.9	8621.9	12.9	8621.9	12.9	8621.9	12.9	1319.6	0.1	5.0	0.1
Chr15b	7990.0	0.0	5141.0	35.7	5141.0	35.7	5141.0	35.7	558.6	0.1	6.0	0.1
Chr15c	9504.0	0.0	9504.0	0.0	9504.0	0.0	9504.0	0.0	145.4	0.1	3.5	0.1
Chr18a	10758.3	3.1	9515.3	14.3	9515.3	14.3	9515.3	14.3	11206.6	0.2	13.5	0.2
Chr18b	-	-	1534.0	0.0	1534.0	0.0	1534.0	0.0	14400(*)[a]	0.1	55.4	0.2
Chr20a	-	-	2156.0	1.6	2156.0	1.6	2156.0	1.6	14400(*)	0.2	181.7	0.3
Chr20b	-	-	2242.9	2.4	2242.9	2.4	2242.9	2.4	14400(*)	0.5	342.2	0.5
Chr20c	-	-	8816.6	37.7	8816.6	37.7	8816.6	37.7	14400(*)	0.3	130.7	0.4
Chr22a	-	-	5993.6	2.6	5993.6	2.6	5993.6	2.6	14400(*)	0.5	189.4	0.6
Chr22b	-	-	6099.0	1.5	6099.0	1.5	6099.0	1.5	14400(*)	0.6	82.1	0.6
Chr25a	-	-	3272.0	13.8	3272.0	13.8	3272.0	13.8	14400(*)	0.9	707.4	1.1
P.ave.	-	-	6336.9	12.1	6336.9	12.1	6336.9	12.1	8150.0	0.3	122.8	0.3
Esc16a	48.0	29.4	0.0	100.0	0.0	100.0	0.0	100.0	6709.9	1.9	15.9	1.1
Esc16b	278.0	4.8	0.0	100.0	0.0	100.0	0.0	100.0	8869.6	12.1	14.4	6.3
Esc16c	-	-	0.0	100.0	0.0	100.0	0.0	100.0	14400(*)	2.9	13.8	1.7
Esc16d	4.0	75.0	0.0	100.0	0.0	100.0	0.0	100.0	9614.6	0.3	13.8	0.2
Esc16e	14.0	50.0	0.0	100.0	0.0	100.0	0.0	100.0	6550.5	0.2	15.2	0.2
Esc16f	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4969.2	0.0	14.4	0.0
Esc16g	14.0	46.2	0.0	100.0	0.0	100.0	0.0	100.0	5449.4	0.3	16.4	0.2
Esc16h	704.0	29.3	690.0	30.7	0.0	100.0	0.0	100.0	7846.0	1532.8	14.1	12.9
Esc16i	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	3547.3	0.1	14.6	0.1
Esc16j	2.0	75.0	0.0	100.0	0.0	100.0	0.0	100.0	5023.1	0.1	13.8	0.1
P.ave.	-	-	69.0	83.1	0.0	90.0	0.0	90.0	7298.0	155.1	14.6	2.3
Esc32a	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	217.3	14400(*)	94.5
Esc32b	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	756.3	14400(*)	285.8
Esc32c	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	845.6	14400(*)	402.8
Esc32d	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	345.0	14400(*)	151.8
Esc32e	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	0.1	14400(*)	0.1
Esc32f	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	0.1	14400(*)	0.1
Esc32g	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	0.2	14400(*)	0.1
Esc32h	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	1159.5	14400(*)	471.2
P.ave.	-	-	0.0	100.0	-	-	0.0	100.0	14400(*)	415.5	14400(*)	175.8
T.ave.	-	-	2793.9	73.8	-	-	2772.4	58.4	9446.2	152.4	3658.3	44.8

[a]The CPU time limit, 14400 seconds, was reached.

Table 4: The CPU times spent of solving the tested QAP instances implemented with the formulations *IPQAPR-I*, *IPQAPR-II*, *IPQAPR-III* and *IPQAPR-IV*

Ins.	Formulation setup time (Sec.)				IP CPU Time (Sec.)				Litera.[a]
	R-I	R-IV	R-II	R-III	R-I	R-IV	R-II	R-III	
Chr12a	5.6	0.8	1.8	0.5	23.7	0.6	49.4	1.0	9.4
Chr12b	5.5	0.7	1.9	0.3	36.8	0.9	126.5	1.0	2.8
Chr12c	5.3	0.7	1.7	0.3	53.7	0.8	67.8	1.2	1.2
Chr15a	23.4	1.7	7.1	0.8	14400(*)[b]	2.1	720.4	3.2	61.3
Chr15b	23.5	1.7	7.2	0.9	741.1	2.9	793.7	7.9	28.0
Chr15c	23.4	1.5	7.0	0.8	190.9	0.1	3.7	0.1	11.7
Chr18a	109.7	3.7	25.3	2.2	14400(*)	4.8	974.8	6.6	89.7
Chr18b	98.8	3.4	26.2	2.4	14400(*)	2.5	835.6	3.6	37.1
Chr20a	320.5	6.2	57.0	4.3	14400(*)	25.8	8691.8	28.7	181.4
Chr20b	322.6	5.7	57.1	3.8	14400(*)	33.0	7237.1	56.0	56.2
Chr20c	318.8	6.4	57.2	4.5	14400(*)	69.1	14400(*)	313.6	138.0
Chr22a	920.6	10.8	136.9	8.1	14400(*)	30.1	10346.7	21.4	166.3
Chr22b	922.8	10.2	137.8	7.4	14400(*)	9.7	4186.7	15.1	143.3
Chr25a	3319.0	26.0	585.2	17.9	14400(*)	121.1	14400(*)	274.0	333.3
P.ave.	458.5	5.7	79.3	3.9	9331.9	21.7	4488.1	52.4	90.0
Esc16a	37.3	5.1	11.5	3.2	14400(*)	14400(*)	14400(*)	14400(*)	65.0
Esc16b	37.3	12.9	11.3	7.7	14400(*)	14400(*)	14400(*)	14400(*)	546.0
Esc16c	37.4	6.6	10.9	4.0	14400(*)	14400(*)	14400(*)	14400(*)	3990.0
Esc16d	37.3	2.8	10.9	1.6	14400(*)	2685.0	14400(*)	1334.5	492.0
Esc16e	38.2	2.9	10.9	1.7	14400(*)	1001.7	14400(*)	1086.7	66.0
Esc16f	37.3	0.5	11.6	0.2	4811.4	0.0	14.7	0.0	0.0
Esc16g	37.4	2.9	11.6	1.8	14400(*)	348.2	14400(*)	301.5	7.0
Esc16h	37.4	30.2	11.6	11.0	14400(*)	14400(*)	14400(*)	14400(*)	22082.0
Esc16i	37.3	2.1	11.6	1.2	14400(*)	65.2	14400(*)	65.2	14.0
Esc16j	38.2	1.8	10.9	1.0	14400(*)	9.3	14400(*)	19.4	1.0
P.ave.	37.5	6.8	11.3	3.3	13441.1	6170.9	12961.5	6040.7	2726.3
Esc32a	29161.5	728.2	7095.2	638.7	14400(*)	14400(*)	14400(*)	14400(*)	-
Esc32b	29176.7	1253.3	7198.3	1071.1	14400(*)	14400(*)	14400(*)	14400(*)	-
Esc32c	29186.2	1633.2	7109.0	1363.4	14400(*)	14400(*)	14400(*)	14400(*)	-
Esc32d	29271.5	1143.1	7131.2	1009.5	14400(*)	14400(*)	14400(*)	14400(*)	-
Esc32e	29168.1	62.8	7265.1	58.4	14400(*)	5.9	14400(*)	14400(*)	600.0
Esc32f	30419.7	63.0	7135.3	58.1	14400(*)	5.9	14400(*)	14400(*)	600.0
Esc32g	29258.3	119.3	7499.9	112.2	14400(*)	208.4	14400(*)	100.2	-
Esc32h	29318.7	1740.6	7510.1	1447.0	14400(*)	14400(*)	14400(*)	14400(*)	-
P.ave.	29370.1	842.9	7243.0	719.8	14400(*)	9027.5	14400(*)	12612.5	-
T.ave.	7554.9	215.3	1849.0	182.7	11883.1	4194.8	9614.0	5063.8	-

[a] For the instances Chr12a - Char25a, the CPU times are obtained with a UNIVAC 1100/60. For the instances Esc16a - Esc16j, the CPU times are obtained with a 16-processor MEIKO Computing Surface with Intel i860 processors. For the instances Esc32e and Esc32f, the CPU times are obtained with NEC Cenju-3.

[b] The CPU time limit, 14400 seconds, was reached.

flow and a distance, $q_{ijkl} = f_{ik}d_{jl}$, one obtains a sparse matrix if some of the factors are null. In most situations some values f_{ik} are zero, and we exploited this situation here. The results obtained here also apply, by symmetry, to the case where $d_{jl} = 0$.

Finally we would like to point out that these results are an example showing that the tighter formulation is not always the most effective to solve a problem. If we consider as a measure the solving time of the IP, or the quality of the integer solution within some time limit, the less tight formulations outperformed the tightest. The technique used here to weaken a formulation to make its LP relaxation much easier to solve may prove fruitful in other applications.

Acknowledgement

Thanks to the support from the project ADONET-Marie Curie Research Training Networks-Contract Nr MRTN-CT-2003-504438. We wish to thank the Faculty of Sciences of the University of Lisbon for providing the software Cplex 9.0, and thank Peter Hahn for providing the information of the CPU times in the literatures. We also thank the support of the grant URJC-CM-2007-CET-1622 from Rey Juan Carlos University and Comunidad de Madrid (Spain) and the grant MTM2006-14961-C05-05 from the Spanish government.

References

[1] 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*, 180(3):983–996, 2007.

[2] W. P. Adams and T. A. Johnson. Improved linear programming-based lower bounds for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society*, 16:43–75, 1994.

[3] R. K. Ahuja, J. B. Orlin, and A. Tivari. A greedy genetic algorithm for the quadratic assignment problem. *Computers and Operations Research*, 27(10):917–934, 2000.

[4] K. Anstreicher, N. Brixius, J.-P. Goux, and J. Linderoth. Solving large quadratic assignment problems on computational grids. *Mathematical Programming*, 91(3):563–588, 2002.

[5] K. M. Anstreicher. Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, 97(1-2):27–42, 2003.

[6] N. W. Brixius and K. M. Anstreicher. Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, 16(1-4):49–68, 2001.

[7] A. Brungger, A. Marzetta, J. Clausen, and M. Perregaard. Joining forces in solving large-scale quadratic assignment problem in parallel. *11th International Parallel Processing Symposium (IPPS '97), 1-5 April 1997, Geneva, Switzerland, Proceedings*, pages 418–427.

[8] M. J. Brusco and S. Stahl. Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices. *Journal of Classification*, 17(2):197–223, 2000.

[9] R. E. Burkard, S. E. Karisch, and F. Rendl. Qaplib-a quadratic assignment problem library. *Journal of Global Optimization*, 10(4):391– 403, 1997.

[10] E. Cela. *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer Academic Publishers, London, 1998.

[11] N. Christofides and E. Benavent. An exact algorithm for the quadratic assignment problem. *Operations Research*, 37(5):760–768, 1989.

[12] J. Clausen and M. Perregaard. Solving large quadratic assignment problems in parallel. *Computational Optimization and Applications*, 8(2):111–127, 1997.

[13] C. W. Commander. A survey of the quadratic assignment problem, with applications. *Morehead Electronic Journal of Applicable Mathematics*, (4):1–15, 2005.

[14] D. T. Connolly. An improved annealing scheme for the qap. *European Journal of Operational Research*, 46(1):93–100, 1990.

[15] T. G. Crainic, B. Le Cun, and C. Roucairol. *Parallel Combinatorial Optimization*, chapter 1, Parallel Branch-and-Bound Algorithms, pages 1–28. John Wiley & Sons, Inc., 2006.

[16] Z. Drezner. Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Computers and Operations Research*, 35(3):717–736, 2008.

[17] E. Duman and I. Or. The quadratic assignment problem in the context of the printed circuit board assembly process. *Computers and Operations Research*, 34(1):163–179, 2007.

[18] G. Erdogan and B. Tansel. A branch-and-cut algorithm for quadratic assignment problems based on linearizations. *Computers and Operations Research*, 34(4):1085–1106, 2007.

[19] A. M. Frieze and J. Yadegar. On the quadratic assignment problem. *Discrete Applied Mathematics*, 5(1):89–98, 1983.

[20] L. M. Gambardella, E. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2):167–176, 1999.

[21] A. M. Geoffrion and G. W. Graves. Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/lp approach. *Operations Research*, 24(4):595–610, 1976.

[22] P. C. Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *SIAM Journal on Applied Mathematics*, 10(2):305–313, 1962.

[23] S. W. Hadley, F. Rendl, and H. Wolkowicz. A new lower bound via projection for the quadratic assignment problem. *Mathematics of Operations Research*, 17(3):727–739, 1992.

[24] P. Hahn, T. Grant, and N. Hall. A branch-and-bound algorithm for the quadratic assignment problem based on the hungarian method. *European Journal of Operational Research*, 108(3):629–640, 1998.

[25] L. Kaufmann and F. Broeckx. An algorithm for the quadratic assignment problem using benders' decomposition. *European Journal of Operational Research*, 2(3):204–211, 1978.

[26] T. C. Koopmans and M. J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, 1957.

[27] E. L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963.

[28] Y. Li, P. M. Pardalos, K. G. Ramakrishnan, and et al. Lower bounds for the quadratic assignment problem. *Annals of Operations Research*, 50(1):387–441, 1994.

[29] Y. Li, P. M. Pardalos, and M G. C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society*, 16:237–261, 1994.

[30] E. M. Loiola, N. M. M. Abreu, P. O. Boaventura-Netto, and et al. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007.

[31] B. Mans, T. Mautor, and C. Roucairol. A parallel depth first search branch and bound algorithm for the quadratic assignment problem. *European Journal of Operational Research*, 81(3):617–628, 1995.

[32] A. Misevicius. A modified simulated annealing algorithm for the quadratic assignment problem. *Informatica*, 14(4):497–514, 2003.

[33] P. M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society*, 16:1–42, 1994.

[34] M. A. Pollatschek, N. Gershoni, and Y. T. Radday. Optimization of the typewriter keyboard by simulation. *Angewandte Informatik*, 17(0):438–439, 1976.

[35] K. G. Ramakrishnan, M. G. C. Resende, and P. M. Pardalos. A branch and bound algorithm for the quadratic assignment problem using a lower bound based on linear programming. *In State of the Art in Global Optimization: Computational Methods and Applications, Kluwer Academic Publishers*, pages 57–73, 1996.

[36] M. G. C. Resende, K. G. Ramakrishnan, and Z. Drezner. Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. *Operations Research*, 43(5):781–791, 1995.

[37] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the Association of Computing Machinery*, 23(3):555–565, 1976.

[38] Y. Xia. Improved gilmore-lawler bound for quadratic assignment problems. *Chinese Journal of Engineering Mathematics*, 24(3):401–413, 2007.

[39] Q. Zhao, S. E. Karisch, F. Rendl, and et al. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1):71–109, 1998.