Solving the Quadratic Assignment Problem Using Semi-Lagrangian Relaxation

Huizhen Zhang[1,*] *Cesar Beltran-Royo*[2] *Bo Wang*[1] *Liang Ma*[1] *Ziying Zhang*[3]

1. School of Management, University of Shanghai for Science and Technology, Shanghai 200093, P.R.China;
2. Department of Statistics and Operations Research, Rey Juan Carlos University, Mostoles (Madrid) 28933, Spain;
3. School of Materials Engineering, Shanghai University of Engineering Science, Shanghai 201620, P.R.China

Abstract: The Semi-Lagrangian relaxation (SLR), a new exact method for combinatorial optimization problems with equality constraints, is applied to the quadratic assignment problem (QAP). A dual ascent algorithm with finite convergence is developed for solving the Semi-Lagrangian dual problem associated to the QAP. We have performed computational experiments on 30 moderately difficult QAP instances by using the mixed integer programming solvers, Cplex, and SLR+Cplex, respectively. The numerical results not only further illustrate that the SLR and the developed dual ascent algorithm can be used to solve the QAP reasonably, but also disclose an interesting fact: comparing with solving the unreduced problem, the reduced oracle problem cannot be always effectively solved by using Cplex in terms of the CPU time.

Manuscript received May 4, 2015.

*Corresponding author.

This work was supported by the National Natural Science Foundation of China (71401106), the Innovation Program of Shanghai Municipal Education Commission (14YZ090), the Shanghai Natural Science Foundation (14ZR1418700), the Shanghai First-class Academic Discipline Project (S1201YLXK), the Hujiang Foundation of China (A14006), the grant S2009/esp-1594 from the Comunidad de Madrid (Spain) and the grant MTM2012-36163-C06-06 from the Spanish government.

Keywords: Quadratic assignment problem (QAP), Semi-Lagrangian relaxation (SLR), Lagrangian relaxation, Dual ascent algorithm

1 Introduction

The quadratic assignment problem (QAP) is one of the most well-known and difficult NP-hard combinatorial optimization problems. Even finding an ε-approximate solution is difficult [1]. It can be applied in several fields such as backboard wiring [2], typewriter keyboards and control panels design [3], scheduling [4], numerical analysis [5], etc. Moreover, many well-known classical combinatorial optimization problems such as the traveling salesman problem, the graph partitioning problem and the maximum clique problem, can be reformulated as special cases of the QAP, see [6] and [7] for more details.

In general, the QAP can be described as a one-to-one assignment problem of n facilities to n locations, which minimizes the sum of the total quadratic interaction cost, the flow between the facilities multiplied by their distances, and the total linear cost associated with allocating a facility to a certain location.

Consider the set $\boldsymbol{N}$={1,2,…,n}and three $n\times n$ matrices $\boldsymbol{F}$=(f_{ij}), $\boldsymbol{D}$=(d_{ij}) and $\boldsymbol{C}$=(c_{ij}). The QAP with coefficient matrices $\boldsymbol{F}$, $\boldsymbol{D}$ and $\boldsymbol{C}$ can be stated as follows:

$$\min_{x\in \boldsymbol{X}} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik}d_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\ , \quad (1)$$

where

$$\boldsymbol{X} = \left\{ \boldsymbol{x} \middle| \ \sum_{j=1}^{n} x_{ij} = 1, \quad i \in \boldsymbol{N}, \right. \quad (2)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in \boldsymbol{N}, \quad (3)$$

$$\left. x_{ij} \in \{0,1\}, \quad i, j \in \boldsymbol{N} \right\}. \quad (4)$$

f_{ik} denotes the amount of flow between facilities i and k, d_{jl} denotes the distance between locations j and l, and c_{ij} denotes the cost of locating facility i at location j. x_{ij} =1if facility i is assigned to location j, otherwise, x_{ij} =0.

In [8] a more general expression of (1) was introduced by using a four-dimensional array q_{ijkl} instead of the flow-distance products $f_{ik}d_{jl}$:

$$\min_{x\in X} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} q_{ijkl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\ . \quad (5)$$

The QAP has drawn the researcher's attention worldwide and extensive research has been done since it was proposed by Koopmans and Beckmann in 1957 [9]. For example, a complete study of valid inequalities, facets and lifting theorems can be found in [10, 11, 12], several linearizations of the objective function (1) are achieved by defining new variables and introducing new linear (and binary) constraints [13, 14, 15], and a number of bounding techniques, such as Gilmore-Lawler type bounds [16], convex quadratic programming bound [17], bounds based on dual approach and bounds based on semidefinite programming relaxation [18, 19], have also been developed. Another example is solution techniques, some of the most challenging QAP instances have been solved by exact methods and heuristic algorithms like simulated annealing [20], Tabu search [20, 21], migrating birds optimization algorithm [22], and so on. The readers can also refer to the review papers [23, 24] and books/book chapters [6, 7, 11, 25] for details of applications, theory and algorithms of the QAP.

Lagrangian relaxation is one of the most popular bounding techniques in combinatorial optimization [26] and it has also been used to solve the QAP by combining with branch and bound [27, 28]. Semi-Lagrangian relaxation (SLR), a new type of Lagrangian relaxation with zero duality gap, has been proposed in recent years and has allowed to solve large-scale combinatorial optimization problems with equality constraints by means of general purpose mixed integer programming solvers [29, 30]. However, as far as we know, the SLR applied to the QAP has not been reported in literature. In this paper, we will investigate whether the SLR can be applied to effectively solve the QAP:

Are the corresponding subproblems tractable? Can they be solved by standard mixed integer programming solvers as for example Cplex?

In order to implement the SLR to solve the QAP, the starting point is the reduced linearizations of the QAP proposed in [31], named QAP-I and QAP-II. These formulations have a reduced number of constraints and therefore seem well suited for the SLR in order to consider a reduced number of Lagrangian multipliers (dual variables). Furthermore, a dual ascent algorithm is proposed to solve the corresponding SLR dual problem. Finally, to investigate the effectivity of the SLR on solving the QAP, we use it to solve a set of 30 instances from the QAP library QAPLIB [32], which range from 12 to 32 facilities/locations.

This paper is organized as follows: In Section 2 we briefly review the main properties of the SLR. In Section 3 we introduce QAP-I and QAP-II, two reduced linearizations of the QAP. In Section 4, the SLR is applied to formulations QAP-I and QAP-II, and a dual ascent algorithm is proposed for solving the corresponding SLR dual problems. In Section 5, we report the results of the computational experiment intended to compare the plain use of Cplex versus the combined use of Cplex and SLR. Finally, in Section 6 we conclude.

2 Semi-Lagrangian relaxation

The concept of SLR was introduced in [29] and applied to the uncapacitated facility location problem in [30]. In this section, we summarize the main results of these two papers in Theorem 1. Consider the following problem, to be named "primal" henceforth:

$$\boldsymbol{z}^* = \min_{\boldsymbol{x}} \ \boldsymbol{c}^T \boldsymbol{x}, \tag{6a}$$

$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{6b}$$

$$\boldsymbol{x} \in \boldsymbol{X} \subset \mathbf{P} \cap \mathbf{N}^n, \tag{6c}$$

where $\boldsymbol{A} \in \mathbf{R}^m \times \mathbf{R}^n$, the components of $\boldsymbol{b} \in \mathbf{R}^m$ and $\boldsymbol{c} \in \mathbf{R}^n$ are nonnegative. $\mathbf{P}$ is a polyhedral set, $0 \in \boldsymbol{X}$ and problem (6) is feasible.

The SLR consists in substituting the constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by the equivalent pair of constraints $\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}$ and $\boldsymbol{A}\boldsymbol{x} \ge \boldsymbol{b}$, and then relaxing $\boldsymbol{A}\boldsymbol{x} \ge \boldsymbol{b}$ only. Thus we obtain the SLR dual problem:

$$\mathfrak{L}^* = \max_{\boldsymbol{u} \in \boldsymbol{U}} \boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u}), \tag{7}$$

where $\boldsymbol{u}$ is the SLR multiplier vector (dual variables), $\boldsymbol{U} = \mathbf{R}_+^{\boldsymbol{m}}$ and $\boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u})$ is the SLR dual function defined as

$$\boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u}) = \min_{\boldsymbol{x}} \ \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{u}^T (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}), \tag{8a}$$

$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}, \tag{8b}$$

$$\boldsymbol{x} \in \boldsymbol{X}, \tag{8c}$$

Note that to calculate $\boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u})$ we have to solve problem (8), which we call the oracle at $\boldsymbol{u}$. Also note that with our assumptions its feasible region is bounded. We also have that $\boldsymbol{x} = 0$ is feasible, hence problem (8) has an optimal solution. $\boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u})$ is well-defined, but the minimizer in (8) is not necessarily unique. With some abuse of notation, we write

$$\begin{aligned} \boldsymbol{x}(\boldsymbol{u}) = \arg\min_{\boldsymbol{x}} \{ \boldsymbol{c}^T \boldsymbol{x} + \boldsymbol{u}^T (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \,\big|\, \\ \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b},\ \boldsymbol{x} \in \boldsymbol{X} \}, \end{aligned} \tag{9}$$

to denote one such minimizer.

We denote $\boldsymbol{X}^*$, $\boldsymbol{U}^*$and $\boldsymbol{X(u)}$ the set of optimal solutions of problems (6), (7) and (8), respectively. We say that ($\boldsymbol{x}^*$, $\boldsymbol{u}^*$) is an optimal primal-dual point if $\boldsymbol{u}^* \in \mathrm{int}(\boldsymbol{U}^*)$ and $\boldsymbol{x}^* \in \boldsymbol{X}(\boldsymbol{u}^*)$ $\cap \boldsymbol{X}^*$. Given two sets $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, their addition corresponds to $\boldsymbol{G}_1 + \boldsymbol{G}_2 = \{\boldsymbol{g}_1 + \boldsymbol{g}_2 : \boldsymbol{g}_1 \in \boldsymbol{G}_1$ and $\boldsymbol{g}_2 \in \boldsymbol{G}_2\}$. For any set $\boldsymbol{G}$, $\mathrm{int}(\boldsymbol{G})$ stands for its interior.

Theorem 1 [29, 30] The following statements hold.

1. $\boldsymbol{L_{SLR}}(\boldsymbol{u})$ is concave and $\boldsymbol{b}$-$\boldsymbol{Ax}(\boldsymbol{u})$ is a subgradient at $\boldsymbol{u}$.

2. $\boldsymbol{L_{SLR}}(\boldsymbol{u})$ is monotone and $\boldsymbol{L_{SLR}}(\boldsymbol{u}') \geq \boldsymbol{L_{SLR}}(\boldsymbol{u})$ if $\boldsymbol{u}' \geq \boldsymbol{u}$, with strict inequality if $\boldsymbol{u}' > \boldsymbol{u}$ and $\boldsymbol{u}' \notin \boldsymbol{U}^*$.

3. $\boldsymbol{U}^* + \mathbf{R}_+^{\boldsymbol{m}} = \boldsymbol{U}^*$; thus $\boldsymbol{U}^*$ is an unbounded (convex) set.

4. If $\boldsymbol{x}(\boldsymbol{u})$ is such that $\boldsymbol{Ax}(\boldsymbol{u})$=$\boldsymbol{b}$, then $\boldsymbol{u} \in \boldsymbol{U}^*$ and $\boldsymbol{x}(\boldsymbol{u}) \in \boldsymbol{X}^*$.

5. Conversely, if $\boldsymbol{u} \in \mathrm{int}(\boldsymbol{U}^*)$, then any $\boldsymbol{x}(\boldsymbol{u}) \in \boldsymbol{X}^*$.

6. The SLR closes the duality gap for problem (6), that is, $\boldsymbol{z}^* = \mathfrak{L}^*$.

3 Reduced QAP linearizations

Here we briefly review the reduced QAP linearizations proposed in [31], QAP-I and QAP-II which are derived by eliminating constraints and variables of the Adams and Johnson linearization [14], respectively. First, the following formulation QAP-I is proposed in [31] by eliminating the symmetry constraints of the Adams and Johnson linearization.

QAP-I:

$$\min_{x,y} \sum_{1\le i<k\le n} \sum_{1\le j\ne l\le n} \tilde{q}_{ijkl} y_{ijkl} + \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} x_{ij}, \tag{10}$$

$$\textbf{s.t.} \quad \sum_{l=1, l\ne j}^{n} y_{ijkl} = x_{ij}, \quad i, j, k \in \boldsymbol{N}, i<k, \tag{11}$$

$$\sum_{j=1, j\ne l}^{n} y_{ijkl} \le x_{kl}, \quad i, k, l \in \boldsymbol{N}, i<k, \tag{12}$$

$$y_{ijkl} \in \{0, 1\}, \quad i, j, k, l \in \boldsymbol{N}, i<k, l\ne j \tag{13}$$

$$\boldsymbol{x} \in \boldsymbol{X}, \tag{14}$$

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

Second, formulation QAP-II is further presented in [31] by eliminating variables in the case of a sparse cost matrix.

QAP-II:

$$\min_{x,y} \sum_{\substack{1\le i<k\le n \\ ik\notin \boldsymbol{F}_0}} \sum_{1\le j\ne l\le n} \tilde{q}_{ijkl} y_{ijkl} + \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} x_{ij}, \tag{15}$$

$$\text{s.t.} \quad \sum_{l=1, l\ne j}^{n} y_{ijkl} = x_{ij}, \quad i, j, k \in \boldsymbol{N}, i<k, ik \notin \boldsymbol{F}_0, \tag{16}$$

$$\sum_{j=1, j\ne l}^{n} y_{ijkl} \le x_{kl}, \quad i, k, l \in \boldsymbol{N}, i<k, ik \notin \boldsymbol{F}_0, \tag{17}$$

$$y_{ijkl} \in \{0, 1\}, \quad i, j, k, l \in \boldsymbol{N}, i<k, l \ne j, ik \notin \boldsymbol{F}_0, \tag{18}$$

$$\boldsymbol{x} \in \boldsymbol{X}, \tag{19}$$

where $\boldsymbol{F}_0$ is the index set of zero flows, i.e.,

$$\boldsymbol{F}_0 = \{(i,k) \in \boldsymbol{N} \times \boldsymbol{N} \mid f_{ik} = 0\}.$$

The following result will be used in

Section 4 to derive a dual ascent algorithm intended to solve these two formulations.

Theorem 2 [31] If $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an optimal solution of QAP-I (or QAP-II), then $y^*_{ijkl} = x^*_{ij}x^*_{kl}$ $(i, j, k, l \in \boldsymbol{N},\ i < k,\ l \neq j)$.

As shown in [31], formulations QAP-I and QAP-II have a reduced number of constraints and therefore seem well suited for the SLR in order to consider a reduced number of Lagrangian multipliers (dual variables).

4 Semi-Lagrangian relaxation of the QAP linearization

Before applying the SLR method we stress the following facts:

- Formulations QAP-I and QAP-II are simpler compared with other QAP linearizations, such as Adams and Johnson linearization, Frieze and Yadegar linearization, Padberg and Rijal linearization [6, 25, 11].
- The number of y_{ijkl} variables is $\mathrm{O}(n^4)$. Hopefully many of these variables can be fixed to “0” in the SLR framework.
- It is unnecessary to fix variables x_{ij}, since the number of these variables is $\mathrm{O}(n^2)$.
- If $\boldsymbol{F}_0 = \Phi$ in QAP-II, then QAP-II is equivalent to QAP-I. Thus, in this section we only present the implementation of the SLR to solve QAP-I and the related theoretical results. Formulation QAP-II can be solved in a similar way by the SLR method.

4.1 Semi-Lagrangian dual problem

Following the ideas of the preceding Section 2, we apply the Semi-Lagrangian relaxation to QAP-I and obtain the SLR dual problem:

$$\max_{\boldsymbol{u}} \boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u}), \tag{20}$$

with the *oracle* (dual function)

$$\boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u}) := \min_{\boldsymbol{x},\boldsymbol{y}} \boldsymbol{L}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{y}), \tag{21}$$

$$\text{s.t.} \qquad \boldsymbol{x} \in \boldsymbol{X}, \tag{22}$$

$$\sum_{l=1, l\neq j}^{n} y_{ijkl} \le x_{ij}, \quad i, j, k \in \boldsymbol{N}, i < k, \tag{23}$$

$$\sum_{j=1, j\neq l}^{n} y_{ijkl} \le x_{kl}, \quad i, k, l \in \boldsymbol{N}, i < k, \tag{24}$$

$$y_{ijkl} \in \{0, 1\}, \quad i, j, k, l \in \boldsymbol{N}, i < k, l \neq j, \tag{25}$$

where

$$\begin{aligned}
\boldsymbol{L}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1,k>i}^{n}\sum_{l=1,l\neq j}^{n} \tilde{q}_{ijkl} y_{ijkl} + \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}x_{ij} + \\
&\quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1,k>i}^{n} u_{ijk}\left(x_{ij} - \sum_{l=1,l\neq j}^{n} y_{ijkl}\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1,k>i}^{n}\sum_{l=1,l\neq j}^{n} (\tilde{q}_{ijkl} - u_{ijk}) y_{ijkl} \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{k=1,k>i}^{n} u_{ijk} + p_{ij}\right)x_{ij} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1,k>i}^{n}\sum_{l=1,l\neq j}^{n} \bar{q}(u)_{ijkl}\, y_{ijkl} + \sum_{i=1}^{n}\sum_{j=1}^{n} \bar{u}_{ij}x_{ij}
\end{aligned}$$

with $\bar{q}(\boldsymbol{u})_{ijkl} := \tilde{q}_{ijkl} - u_{ijk}$ and $\bar{u}_{ij} := \sum_{k=1,k>i}^{n} u_{ijk} + p_{ij}$.

The oracle problem $\boldsymbol{L}_{\boldsymbol{SLR}}(\boldsymbol{u})$ contains the assignment constraints over $\boldsymbol{x}$, thus $\boldsymbol{x}(\boldsymbol{u})$ computed by solving (21)-(25) is a feasible solution for (5) and can be used to calculate an upper bound.

From (21)-(25), it is easy to show that there exist optimal solutions ($\boldsymbol{x}(\boldsymbol{u})$,$\boldsymbol{y}(\boldsymbol{u})$) of the SLR oracle $\boldsymbol{L_{SLR}}(\boldsymbol{u})$ such that $y(\boldsymbol{u})_{ijkl}=0$ if $\bar{q}(\boldsymbol{u})_{ijkl}\ge 0$. Therefore, in the oracle $\boldsymbol{L_{SLR}}(\boldsymbol{u})$ some $\boldsymbol{y}(\boldsymbol{u})_{\boldsymbol{ijkl}}$ can be fixed to "0" in advance if their $\bar{q}(\boldsymbol{u})_{ijkl}\ge 0$. This operation, which reduces the size of the oracle, is quite common in Lagrangian relaxation applied to combinatorial optimization. There, using some appropriate argument, one fixes some of the oracle variables and obtains a reduced-size oracle called the *core problem*. Usually we have (much) fewer variables $y(\boldsymbol{u})_{ijkl}$ in the core problem. Roughly speaking, if the size of the core problem is small enough, it will be possible to solve it by an Integer Programming solver (e.g. Cplex, etc.), and this is the main advantage of the core problem.

To control the number of variables $y(\boldsymbol{u})_{ijkl}$ in the core problem, we introduce the sorted costs. For each (i,j,k) ($1\le i,j\le n$, $k>i$), we sort the costs $\left\{\tilde{q}_{ijkl}\middle| l\in \boldsymbol{N}\right\}$, and get the sorted costs:

$$\tilde{q}^1_{ijk}\le \tilde{q}^2_{ijk}\le\cdots\le \tilde{q}^n_{ijk}.$$

For a given u and a triple (i,j,k) ($1\le i<k\le n$, $j\in \boldsymbol{N}$), if u is such that $u_{ijk}=\tilde{q}^r_{ijk}$ ($r\in\{1,2,\cdots,n\}$), then the triple (i,j,k) will have at most r-1 related variables y_{ijkl} with strictly negative reduced cost.

On the other hand, by applying Theorem 1, Statement 1, we have that $\boldsymbol{L_{SLR}}(\boldsymbol{u})$ is concave and vector $\boldsymbol{s}(\boldsymbol{u})$ with

$$s(\boldsymbol{u})_{ijk}=x(\boldsymbol{u})_{ij}-\sum_{l=1,l\ne j}^{n} y(\boldsymbol{u})_{ijkl}, \qquad (26)$$
$$1\le i,j,k\le n, k>i,$$

is a subgradient of $\boldsymbol{L_{SLR}}(\boldsymbol{u})$ at $\boldsymbol{u}$.

The following result gives a sufficient optimality condition for the SLR dual problem (20).

Theorem 3 $\boldsymbol{u}\ge\tilde{\boldsymbol{q}}^{\boldsymbol{n}}\Rightarrow \boldsymbol{u}\in \boldsymbol{U}^*$ and $\boldsymbol{u}>\tilde{\boldsymbol{q}}^{\boldsymbol{n}}\Rightarrow$ $\boldsymbol{u}\in \mathrm{int}(\boldsymbol{U}^*)$.

Proof: Consider the oracle (21)-(25), and assume $\boldsymbol{u}\ge\tilde{\boldsymbol{q}}^{\boldsymbol{n}}$. If there exists an optimal solution of the oracle such that $\sum_{l=1,l\ne j}^{n} y(\boldsymbol{u})_{ijkl}$ $=x(\boldsymbol{u})_{ij}$, $\forall\ (i,j,k)$ and $k>i$, then, by Theorem 1 Statement 4, this solution is optimal for the original problem. Assume that we have an oracle solution with $\sum_{l=1,l\ne j}^{n} y(\boldsymbol{u})_{ijkl}=0\ne x(\boldsymbol{u})_{ij}=1$ for some (i,j,k) and $k>i$, and let l' be such that $x(\boldsymbol{u})_{kl'}=1$ for the given k. By hypothesis, $\tilde{q}_{ijkl'}-u_{ijk}\le \tilde{q}_{ijkl'}-\tilde{q}^n_{ijk}\le 0$. Thus, one can set $y(\boldsymbol{u})_{ijkl'}=x(\boldsymbol{u})_{ij}=1$ without increasing the objective value. The modified solution is also optimal. Hence, there exists an optimal oracle solution with $\sum_{l=1,l\ne j}^{n} y(\boldsymbol{u})_{ijkl}=x(\boldsymbol{u})_{ij}$, $\forall\ (i,j,k)$ ($k>i$) and $\boldsymbol{u}\in \boldsymbol{U}^*$. Furthermore, according to the Statement 3 of Theorem 1, we can show that $\boldsymbol{u}\in \mathrm{int}(\boldsymbol{U}^*)$ if $\boldsymbol{u}>\tilde{\boldsymbol{q}}^{\boldsymbol{n}}$.

■

4.2 Solving the QAP Semi-Lagrangian dual problem

Similar with Lagrangian relaxation, it is a common approach to develop a dual ascent method to solve the combinatorial optimization problem when the SLR is implemented. However, the SLR solving procedure based on dual ascent usually takes a number of time consuming iterations for large-scale problems. Initializing and updating the SLR multiplier vector are the key techniques to reduce the number of iterations. We now present a specialized version of the dual ascent method [30] for formulation QAP-I, which involves a pre-processing phase of initializing the SLR multiplier vector with a near optimal one, and a process of updating the multiplier vector based on the Theorem 2.

4.2.1 Initializing the multiplier vector

By Theorem 1, Statement 3, we know that the set of optimal dual solutions $\boldsymbol{U}^*$ is convex and unbounded such that $U^* + \mathrm{R}_+^{\boldsymbol{m}} = U^*$. In Fig.1 we have a possible representation of this set. We can define the set of non-dominated optimal solutions as the Pareto frontier of the optimal set $\boldsymbol{U}^*$. Fig.1 pictures a possible trajectory of multiplier from the origin to the set. It is easy to observe that an optimal solution $(\boldsymbol{x}(\boldsymbol{u}), \boldsymbol{y}(\boldsymbol{u}))$ of the oracle (21)-(25) has the property that $\boldsymbol{y}(\boldsymbol{u})_{\boldsymbol{ijkl}} = 0$ if the corresponding reduced cost $(\tilde{q}_{ijkl} - u_{ijk})$ is nonnegative. At the origin $\boldsymbol{O}$ ($\boldsymbol{u}$=0) of Fig.1, the oracle is easy and has the trivial solution $\boldsymbol{y}(\boldsymbol{u}) = 0$, but a number of time consuming iterations will be needed to obtain an optimal primal-dual solution $((\boldsymbol{x}(\boldsymbol{u}^*), \boldsymbol{y}(\boldsymbol{u}^*)), \boldsymbol{u}^*)$. At point $\boldsymbol{C}$, far inside the optimal set, all reduced costs are negative, the oracle is essentially equivalent to the original problem QAP-I and is difficult to be solved. The difficulty in solving the oracle increases as one progresses along the path $\mathbf{O}$-$\mathbf{C}$, that is, as one increases $\boldsymbol{u}$. A key issue is how to initialize the multiplier vector $\boldsymbol{u}$ near to the Pareto frontier of $\boldsymbol{U}^*$, point $\mathbf{A}$ in Fig.1, where the oracle problem involves only a few variables. At point $\mathbf{A}$ the oracle may be easy and one may only need to solve a short sequence of moderately difficult oracles in order to get an optimal primal-dual solution.

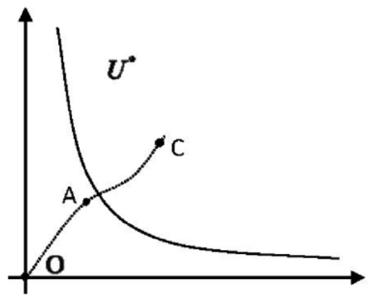

Fig.1 Path from the origin multiplier vector $\boldsymbol{O}$ to optimal set $\boldsymbol{U}^*$

Here we initialize the SLR multiplier vector by using the formula

$$\begin{aligned} &u_{ijk}^1 = \tilde{q}_{ijk}^{\lceil \lambda r' + (1-\lambda)(n+1) \rceil}, \\ &1 \le i, j, k \le n, k > i, \end{aligned} \tag{27}$$

where $0 \le \lambda \le 1$, $\lceil \lambda r' + (1-\lambda)(n+1) \rceil$ denotes the smallest integer which is larger than $\lambda r'$ + $(1-\lambda)(n+1)$, $r' = \min\{r \,|\, \tilde{q}_{ijk}^r > u_{ijk}^0, r \in \{1, 2, \cdots, n+1\}\}$, and $u_{ijk}^0 \le \tilde{q}_{ijk}^{n+1}$ is the initial guess for

u^*_{ijk}. Obviously, initial multiplier vector $\boldsymbol{u}^1$ is a point of the line connecting points $\tilde{\boldsymbol{q}}^{r'}$ and $\tilde{\boldsymbol{q}}^{n+1}(\tilde{q}^{n+1}_{ijk} = \tilde{q}^{n}_{ijk} + \varepsilon, 0 < \varepsilon < 1)$ which is an interior point of $\boldsymbol{U}^*$.

4.2.2 Updating the multiplier vector

Suppose that $(\boldsymbol{x}(\boldsymbol{u^t}), \boldsymbol{y}(\boldsymbol{u^t}))$ is the optimal solution of the oracle $\boldsymbol{L_{SLR}}(\boldsymbol{u^t})$. For any given triple (i, j, k) $(1 \le i, j, k \le n,\ k > i)$, if $s(\boldsymbol{u}^t)_{ijk} = x(\boldsymbol{u}^t)_{ij} - \sum_{l=1,l\neq j}^{n} y(\boldsymbol{u}^t)_{ijkl} = 1$, then there exists $\boldsymbol{l'} \in \boldsymbol{N}$ such that $x(\boldsymbol{u}^t)_{kl'} = x(\boldsymbol{u}^t)_{ij} = 1$ and $y(\boldsymbol{u}^t)_{ijkl'} = 0$. Furthermore, we can conclude that $y(\boldsymbol{u}^t)_{ijkl'}$ was fixed to "0" at the *t*th iteration because of $\tilde{q}(\boldsymbol{u}^t)_{ijkl'} = \tilde{q}_{ijkl'} - \boldsymbol{u}^t_{ijk} \ge 0$ (assume that $\tilde{q}(\boldsymbol{u}^t)_{ijkl'} < 0$, we can define a new feasible solution for the oracle $\boldsymbol{L_{SLR}}(\boldsymbol{u}^t)$, say $(\bar{\boldsymbol{x}}(\boldsymbol{u}^t), \bar{\boldsymbol{y}}(\boldsymbol{u}^t))$, which is equal to $(\boldsymbol{x}(\boldsymbol{u^t}), \boldsymbol{y}(\boldsymbol{u^t}))$ except for component with the given (i, j, k, l'). For this component, we set $\bar{\boldsymbol{y}}(\boldsymbol{u^t})_{\boldsymbol{ijkl'}} = 1$, and therefore $\boldsymbol{L}(\boldsymbol{u}^t, \bar{\boldsymbol{x}}(\boldsymbol{u}^t), \bar{\boldsymbol{y}}(\boldsymbol{u}^t)) < \boldsymbol{L}(\boldsymbol{u}^t, \boldsymbol{x}(\boldsymbol{u}^t), \boldsymbol{y}(\boldsymbol{u}^t))$, which contradicts the fact that $(\boldsymbol{x}(\boldsymbol{u^t}), \boldsymbol{y}(\boldsymbol{u^t}))$ is the optimal solution of the oracle $\boldsymbol{L_{SLR}}(\boldsymbol{u^t})$). In order to make $\boldsymbol{y}(\boldsymbol{u}^{t+1})_{\boldsymbol{ijkl'}} = \boldsymbol{x}(\boldsymbol{u}^{t+1})_{\boldsymbol{ij}}\, \boldsymbol{x}(\boldsymbol{u}^{t+1})_{\boldsymbol{kl'}}$ at the (*t*+1)th iteration, the variable $\boldsymbol{y}(\boldsymbol{u}^{t+1})_{\boldsymbol{ijkl'}}$ should not be fixed to "0". Here we update the SLR multiplier $u^{t+1}_{ijk} = \tilde{q}_{ijkl'} + \varepsilon$ $(0 < \varepsilon < 1)$ for obtaining negative $\tilde{q}(\boldsymbol{u}^{t+1})_{ijkl'}$ $(\tilde{q}(\boldsymbol{u}^{t+1})_{ijkl'} = \tilde{q}_{ijkl'} - u^{t+1}_{ijk} = \tilde{q}_{ijkl'} - \tilde{q}_{ijkl'} - \varepsilon < 0)$.

To maintain the number of variables as low as possible, at each iteration we add at most one variable for per triple (i, j, k) if $s(\boldsymbol{u}^t)_{ijk} = x(\boldsymbol{u}^t)_{ij} - \sum_{l=1,l\neq j}^{n} y(\boldsymbol{u}^t)_{ijkl} = 0$, that is, we update u^t_{ijk} from $\tilde{q}^r_{ijk} + \varepsilon$ to $u^{t+1}_{ijk} = \tilde{q}^{r+1}_{ijk} + \varepsilon$. See the Step 6 of Algorithm 1 in section 4.2.3.

4.2.3 Dual ascent algorithm

Combining with the idea of subsection 4.2.1 and 4.2.2, we develop the following Algorithm 1 for solving the QAP-I.

Algorithm 1

- Input: Data of the QAP-I instance, and $\boldsymbol{u}^0 \ge 0$ initial guess for an optimal point for the SLR dual problem (20).
- Output: $(\boldsymbol{x}(\boldsymbol{u}^*), \boldsymbol{y}(\boldsymbol{u}^*), \boldsymbol{u}^*)$ optimal primal-dual point or $(\boldsymbol{x}(\boldsymbol{u}^*), \boldsymbol{y}(\boldsymbol{u}^*))$ optimal primal point for the QAP-I instance.

1. Initialization:

- For each (i, j, k) ($1 \le i, j \le n,\ k > i$), sort the costs $\{\tilde{q}_{ijkl} \,|\, l \in \boldsymbol{N}\}$ to obtain the sorted costs:

$$\tilde{q}^1_{ijk} \le \tilde{q}^2_{ijk} \le \cdots \le \tilde{q}^n_{ijk}.$$

- For each (i, j, k) $(1 \le i, j \le n,\ k > i)$, set $\tilde{q}^{n+1}_{ijk} = \tilde{q}^n_{ijk} + \varepsilon$ for a $0 < \varepsilon < 1$.
- Set $\tilde{\boldsymbol{N}} = \{1, 2, \cdots, n+1\}$.
- Set upper bound **UB**=inf and the best upper bound **BestUB**=inf.
- Solve the linear programming (LP)

relaxation of the QAP-I to obtain the optimal objective value (LP Cost), and set lower bound **LB=LP Cost**.

2. Initial dual point:

Set $t=1$. For each (i,j,k), set $r' = \min\{r \,|\, \tilde{q}^{r}_{ijk} > u^{0}_{ijk},\ r \in \tilde{\boldsymbol{N}}\}$, and initialize u^{t}_{ijk} by using the formula (27).

3. Oracle call:

- Variable fixing: For each (i,j,k,l), fix $y(\boldsymbol{u}^t)_{ijkl}$ to "0" if $\bar{q}(\boldsymbol{u})_{ijkl} \geq 0$.
- Solve the core problem (21)-(25) to obtain $\boldsymbol{L_{SLR}}(\boldsymbol{u}^t)$ as well as $(\boldsymbol{x}(\boldsymbol{u}^t), \boldsymbol{y}(\boldsymbol{u}^t))$.
- Compute the subgradient $\boldsymbol{S}(\boldsymbol{u}^t)$ such that

$$s(\boldsymbol{u}^t)_{ijk} = x(\boldsymbol{u}^t)_{ij} - \sum_{l=1,l\neq j}^{n} y(\boldsymbol{u}^t)_{ijkl}, \qquad (28)$$
$$1 \leq i,j,k \leq n,\ k > i.$$

- Update the lower bound

$$\mathbf{LB} = \boldsymbol{L_{SLR}}(\boldsymbol{u}^t). \qquad (29)$$

4. Heuristic call: Compute **UB** and **BestUB**.

$$\mathbf{UB} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} q_{ijkl}\, x(\boldsymbol{u}^t)_{ij}\, x(\boldsymbol{u}^t)_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x(\boldsymbol{u}^t)_{ij} \qquad (30)$$

$$\mathbf{BestUB} = \min\{\mathbf{UB}, \mathbf{BestUB}\} \qquad (31)$$

5. Stopping criterion:

If $\boldsymbol{S}(\boldsymbol{u}^t) = 0$, stop with an optimal primal-dual point $(\boldsymbol{x}(\boldsymbol{u}^*), \boldsymbol{y}(\boldsymbol{u}^*), \boldsymbol{u}^*) = (\boldsymbol{x}(\boldsymbol{u}^t), \boldsymbol{y}(\boldsymbol{u}^t), \boldsymbol{u}^t)$. If **LB=BestUB**, update $y(\boldsymbol{u}^t)_{ijkl} = x(\boldsymbol{u}^t)_{ij}\, x(\boldsymbol{u}^t)_{kl}$ $(1 \leq i < k \leq n, 1 \leq j \neq l \leq n)$ and stop with an optimal primal point $(\boldsymbol{x}(\boldsymbol{u}^*), \boldsymbol{y}(\boldsymbol{u}^*)) = (\boldsymbol{x}(\boldsymbol{u}^t), \boldsymbol{y}(\boldsymbol{u}^t))$. Otherwise, proceed to Step 6.

6. Dual point updating:

For per triple (i,j,k), if $\boldsymbol{s}(\boldsymbol{u}^t)_{\boldsymbol{ijk}} = 1$, find l' such that $\boldsymbol{x}(\boldsymbol{u}^t)_{\boldsymbol{kl'}} = 1$ and update $u^{t+1}_{ijk} = \tilde{q}_{ijkl'} + \varepsilon$, otherwise update $u^{t+1}_{ijk} = \min\{\tilde{q}^{r}_{ijk} \,|\, \tilde{q}^{r}_{ijk} > u^{t}_{ijk},\ 1 \leq r \leq n+1\} + \varepsilon$.

7. Set $t = t+1$ and return to Step 3.

Theorem 4 Algorithm 1 is a dual ascent method and it converges to an optimal dual point $\boldsymbol{u}^* \in \boldsymbol{U}^*$ ($\boldsymbol{U}^* \neq \Phi$) or an optimal primal point $(\boldsymbol{x}(\boldsymbol{u}^*), \boldsymbol{y}(\boldsymbol{u}^*))$ after finitely many iterations.

Proof: Let $(\boldsymbol{x}(\boldsymbol{u}^t), \boldsymbol{y}(\boldsymbol{u}^t))$ be the optimal solution of the oracle $\boldsymbol{L_{SLR}}(\boldsymbol{u}^t)$ (t is a finite positive integer). We have four exclusive cases:

Case 1 $\mathbf{LB} \neq \mathbf{BestUB}$. At least for one component of $\boldsymbol{S}(\boldsymbol{u}^t)$, say (i',j',k'), there exists $\boldsymbol{s}(\boldsymbol{u}^t)_{\boldsymbol{i'j'k'}} = 1$ and $u^{t}_{i'j'k'} < \tilde{q}^{n+1}_{i'j'k'}$. In this case, there exist $x(\boldsymbol{u}^t)_{i'j'} = 1$ and $x(\boldsymbol{u}^t)_{k'l'} = 1$ ($k' > i'$, $l' \neq j'$), and $y(\boldsymbol{u}^t)_{i'j'k'l'} = 0$ because $\tilde{q}_{i'j'k'l'} \geq u^{t}_{i'j'k'}$. Then $u^{t}_{i'j'k'}$ can be updated by the Step 6 of Algorithm 1, and $u^{t}_{i'j'k'} < \tilde{q}_{i'j'k'l'} + \varepsilon = u^{t+1}_{i'j'k'}$.

By Theorem 1, Statement 2, we have that $\boldsymbol{L_{SLR}}(\boldsymbol{u}^{t+1}) > \boldsymbol{L_{SLR}}(\boldsymbol{u}^t)$.

Case 2 $\mathbf{LB} \neq \mathbf{BestUB}$. All of the nonzero components of $S(u^t)$ have the associated multipliers $u^t_{ijk} = \tilde{q}^{n+1}_{ijk}$. We will prove by contradiction that this case cannot happen.

For convenience, we can define $\boldsymbol{T} = \{(i,j,k) | \sum_{l=1,l\neq j}^{n} y(\boldsymbol{u}^t)_{ijkl} = 0 \neq x(\boldsymbol{u}^t)_{ij} = 1, 1 \le i,j,k \le n \text{ and } k > i\}$. In this case, let l' be such that $x(\boldsymbol{u}^t)_{kl'} = 1$ for any $(i,j,k) \in \boldsymbol{T}$, then one can modify a solution by setting $y(\boldsymbol{u}^t)_{ijkl'} = x(\boldsymbol{u}^t)_{ij} x(\boldsymbol{u}^t)_{kl'} = 1$ with decreasing the objective value because of $\tilde{q}_{ijkl'} - u^t_{ijk} = \tilde{q}_{ijkl'} - \tilde{q}^{n+1}_{ijk} < 0$, which contradicts $(\boldsymbol{x}(\boldsymbol{u}^t), \boldsymbol{y}(\boldsymbol{u}^t))$, the optimal solution of the oracle $\boldsymbol{L}_{SLR}(\boldsymbol{u}^t)$.

Case 3 $\mathbf{LB}$=$\mathbf{BestUB}$ and $\boldsymbol{S}(\boldsymbol{u}^t) \neq \mathbf{0}$. In this case, $\boldsymbol{x}(\boldsymbol{u}^t)$ is the optimal solution for formulation (5). At Step 5, $(\boldsymbol{x}(\boldsymbol{u}^t), \boldsymbol{y}(\boldsymbol{u}^t))$ updated by using $y(\boldsymbol{u}^t)_{ijkl} = x(\boldsymbol{u}^t)_{ij} x(\boldsymbol{u}^t)_{kl}$ $(1 \le i < k \le n, 1 \le j \neq l \le n)$ is optimal to QAP-I (see Theorem 2).

Case 4 $\mathbf{LB}$=$\mathbf{BestUB}$ and $\boldsymbol{S}(\boldsymbol{u}^t) = \mathbf{0}$. By Theorem 1, Statement 4, we have $\boldsymbol{u}^t \in \boldsymbol{U}^*$.

■

5 Computational experiments

As we have already stated, the objective of this paper is to study whether the SLR can be implemented to solve the Quadratic Assignment Problem (QAP) effectively, when one uses a general purpose mixed integer linear programming solver combined with a standard PC. To study the performance of the SLR solution procedure, a set of instances from QAPLIB [32] was solved by combining the developed dual ascent method, Algorithm 1 in the previous Section, with Cplex (default settings).

In our numerical experiments, the CPU time limit was set to 14400 seconds. The experiments were conducted on a laptop with a processor Intel Core Duo 2.80GHz and with 3.95 GB of RAM. Cplex 12.5 interfaced with Matlab R2010a was used to solve the QAP instances. On the one hand, we used plain Cplex to assess the difficulty of the QAP instances and on the other hand, we also used Cplex as the integer programming solver to compute $\boldsymbol{L}_{SLR}(\boldsymbol{u})$ at each iteration of Algorithm 1. For convenience, we set $\boldsymbol{u}^0$ equal to the optimal vector of dual variables associated with constraints (11) of the LP relaxation of QAP-I (constraints (16) for QAP-II) in Algorithm 1. The tuning parameters of Algorithm 1, λ and ε were set to 0.5.

In Table 1 we describe the 30 instances used in our test, including, the optimal cost (Opt. Cost), the size of the instances n, the number of variables x_{ij} and y_{ijkl}, and the number of constraints (Nb. of Cons.). These instances can be divided into two groups: sparse instances (Chr18a-Scr20) and dense ones (Nug12-Had20). As mentioned in Section 4, the integer programming (IP) formulations QAP-I and QAP-II were applied to the dense instances and sparse ones, respectively. In Table 1, we also report the *density of the flow matrix* $\boldsymbol{F} = (\boldsymbol{f}_{ij})$ (*DFM*), the proportion of non-zero elements except the diagonal ones in the flow matrix F, which is defined by using the following formula:

$$DFM = \frac{\text{the number of the nonzero elements } - \ n}{n^2 - n} \times 100 \cdot$$

In Table 2 we report the results obtained with Cplex 12.5 with default settings and 14400 seconds of CPU time limit. Column Cost reports the best objective function value computed by Cplex which is also an upper bound of the optimal cost. Column $Opt.gap$ reports its optimal gap, $Opt.gap = \left|\frac{\text{Cost-Opt.Cost}}{\text{Opt.Cost}}\right| \times 100\%$, which provides a relative measure of proximity of the objective function value to the optimal value. Columns LP Cost and $LB\ gap$ report the linear programming (LP) relaxation value (lower bound) and the related gap between the lower bound and the optimal cost, $LB\ gap = \frac{\text{Opt.Cost-LPCost}}{\text{Opt.Cost}} \times 100$, respectively. The corresponding CPU time (in seconds) for solving the QAP-I, QAP-II and their LP relaxation is presented in columns IP (sec.) and LP (sec.), respectively.

Within the allowed time limit, 12 of the 30 instances were solved to optimality by plain Cplex, which are indicated by "0" optimality gap. The other 18 instances which were not proved their optimality are marked with (*). Note that, although Cplex computed the optimal costs for instances Nug12, Nug15 and Had14, they were not proved optimally (namely, Cplex did not exit optimally) and are also marked with (*). We observe that Cplex solved the LP relaxation for all of the cases in a few seconds. However, as expected, the bounds given by the LP relaxation of formulations QAP-I and QAP-II are weaker than the SLR dual bound presented in Table 3. The $LB\ gap$ is greater than 10% for most of the cases, even equal to 100% for the cases Esc32a-Esc32h and Nug12-Nug18.

In Table 3 we report the results obtained with the SLR solution procedure based on the dual ascent method, Algorithm 1. Columns LB, $LB\ gap$, UB and $Opt.gap$ report the dual lower bound, the lower bound gap (in %), the upper bound and the optimality gap (in %), respectively. If Algorithm 1 converged or gave a reasonable lower bound within 14400 seconds, then we report the convergent/ reasonable lower bound. For example, Algorithm1 took a total CPU time of 2.03 seconds to solve Chr18a and obtained the optimal lower/upper bound 11098, but Algorithm 1 did not converge and only gave the reasonable lower bound[1] 636 for the instance Esc32c. Otherwise, the symbol "-" is presented instead of the unreasonable bound,

[1] As discussed in previous section, the optimal solution of the oracle problem should be the lower bound to the QAP, but if the SLR did not solve one oracle problem optimally within 14400 seconds, the cost outputed can be greater or less than the optimal (best known) cost presented in Table 1. We call it as reasonable bound if the cost outputed is less than the optimal one, otherwise, we call it as unreasonable bound.

see cases Esc32a, Esc32b, Esc32d and Esc32h. Columns T. CPU time, CPU time of last iter. and Nb. of iters. report the total CPU time, the CPU time of the last iteration and the number of iterations, respectively. There are four cases. Case 1: we report the convergent time if the algorithm converged within 14400 seconds. Case 2: the algorithm converged and the total CPU time after the last finished iteration was longer than 14400 seconds. For example, Algorithm 1 took a total CPU time of 18712.82 seconds to perform 3 iterations in instance Nug12 and converged to an optimal solution. Case 3: the algorithm did not converge and the total CPU time after the last finished iteration was longer than 14400 seconds. For example, Algorithm 1 took 20902.76 seconds to perform 3 complete iterations in instance Had12 but did not converge to an optimal solution. Case 4: the first iteration went beyond 14400 seconds. In this case the algorithm was stopped before finishing even the first iteration, we report 1 (unfinished) iteration and 14400 seconds of CPU time. Note that in Case 3 and 4 we indicate the non-convergent cases by the symbol (*) in columns T. CPU time (sec.) and CPU time of last iter. . We observe that 10 of the 30 instances were solved to optimality by Algorithm 1. Note that, the $Opt.gap$ of instance Scr15 is equal to 0.00%, but instance Scr15 belongs to case 3 because its lower bound 50944 is not equal to its optimal upper bound 51140. Although Algorithm 1 computed the optimal costs for instances Esc32e and Esc32g, it did not prove its optimality. Thus, instances Esc32e and Esc32g are also marked with (*). The other instances, except Esc32c, Had12 and Had20, have an $Opt.gap$ greater than the one obtained by using plain Cplex. These greater $Opt.gaps$ indicate that Algorithm 1 cannot always solve the large-scale (or hard) QAP instances efficiently within the limited CPU time.

An advantage of Algorithm 1 is that one can reduce the number of variables, namely, we can fix many y_{ijkl} variables to “0”. Notice that the number of unfixed y_{ijkl} variables is different for each SLR iteration and therefore we give average figures in percent corresponding to all the SLR iterations (APY stands for “Average percentage of unfixed y_{ijkl} variables” in Table 3). Comparing the CPU time in Table 2 and Table 3, we can draw a conclusion: the solution time needed by Cplex to solve the oracle (one iteration of Algorithm 1) is, in general, shorter than the Cplex time to solve the original QAP. This is not surprising since the oracle has a reduced number of variables. However this is not always the case. For example, Cplex took 4.51 and 34.21 seconds to solve the instances Esc32e and Esc32g optimally, however, Algorithm 1 did not optimally solve one oracle problem (reduced problem) for these two instances within 14400 seconds.

Table1. Instances description

Ins.	**Opt. Cost**	n	DFM(%)	**Nb. of** x	**Nb. of** y	**Nb. of Cons.**
Chr18a	11098	18	11.11	324	5202	648
Chr18b	1534	18	11.11	324	5202	648
Chr20a	2192	20	10.00	400	7220	800
Chr20b	2298	20	10.00	400	7220	800
Chr20c	14142	20	10.00	400	7220	800
Chr22a	6156	22	9.09	484	9702	968
Chr22b	6194	22	9.09	484	9702	968
Chr25a	3796	25	8.00	625	14400	1250
P.ave.[2]	5926.25	20.63	9.80	430.13	8233.50	860.25
Esc32a	130	32	14.92	1024	73408	4800
Esc32b	168	32	21.77	1024	107136	6976
Esc32c	642	32	26.41	1024	129952	8448
Esc32d	200	32	18.15	1024	89280	5824
Esc32e	2	32	1.21	1024	5952	448
Esc32g	6	32	1.81	1024	8928	640
Esc32h	438	32	28.43	1024	139872	9088
P.ave.	226.57	32.00	16.10	1024.00	79218.29	5174.86
Scr12	31410	12	42.42	144	3696	696
Scr15	51140	15	40.00	225	8820	1290
Scr20	110030	20	32.63	400	23560	2520
P.ave.	64193.33	15.67	38.35	256.33	12025.33	1502.00
Nug12	578	12	100.00	144	8712	1608
Nug14	1014	14	100.00	196	16562	2576
Nug15	1150	15	100.00	225	22050	3180
Nug16a	1610	16	100.00	256	28800	3872
Nug16b	1240	16	100.00	256	28800	3872
Nug17	1732	17	100.00	289	36992	4658
Nug18	1930	18	100.00	324	46818	5544
P.ave.	1322.00	15.43	100.00	241.43	26962.00	3615.71
Had12	1652	12	100.00	144	8712	1608
Had14	2724	14	100.00	196	16562	2576
Had16	3720	16	100.00	256	28800	3872
Had18	5358	18	100.00	324	46818	5544
Had20	6922	20	100.00	400	72200	7640
P.ave.	4075.20	16.00	100.00	264.00	34618.40	4248.00
G.ave.[3]	9040.20	20.80	50.21	479.60	33943.27	3138.73

[2] In this paper, P.ave. is the abbreviation for partial average.
[3] G.ave. is the abbreviation for global average.

Table2. Cplex performance

Ins.	Cost	*Opt.gap* (%)	LP Cost	*LB gap* (%)	Cplex time LP(sec.)	IP(sec.)
Chr18a	11098	0.00	9515.31	14.26	0.06	1.22
Chr18b	1534	0.00	1534.00	0.00	0.02	0.53
Chr20a	2192	0.00	2156.00	1.64	0.06	1.08
Chr20b	2298	0.00	2242.92	2.40	0.05	2.18
Chr20c	14142	0.00	8816.59	37.66	0.06	67.96
Chr22a	6156	0.00	5993.60	2.64	0.09	1.78
Chr22b	6194	0.00	6099.02	1.53	0.08	1.76
Chr25a	3796	0.00	3272.01	13.80	0.13	49.28
P.ave.	5926.25	0.00	4953.68	9.24	0.07	15.72
Esc32a	194	49.23	0.00	100.00	0.91	14400(*)
Esc32b	244	45.24	0.00	100.00	3.29	14400(*)
Esc32c	716	11.53	0.00	100.00	8.16	14400(*)
Esc32d	232	16.00	0.00	100.00	2.54	14400(*)
Esc32e	2	0.00	0.00	100.00	0.03	4.51
Esc32g	6	0.00	0.00	100.00	0.05	34.21
Esc32h	492	12.33	0.00	100.00	8.41	14400(*)
P.ave.	269.43	19.19	0.00	100.00	3.34	10293.76
Scr12	31410	0.00	25474.00	18.90	0.05	32.78
Scr15	51140	0.00	40026.00	21.73	0.11	2816.24
Scr20	113126	2.81	75420.00	31.46	0.52	14400(*)
P.ave.	65225.33	0.94	46973.33	24.03	0.22	5750.03
Nug12	578	0.00	0.00	100.00	0.27	14400(*)
Nug14	1060	4.54	0.00	100.00	0.80	14400(*)
Nug15	1150	0.00	0.00	100.00	0.78	14400(*)
Nug16a	1722	6.96	0.00	100.00	1.19	14400(*)
Nug16b	1318	6.29	0.00	100.00	0.98	14400(*)
Nug17	1886	8.89	0.00	100.00	1.87	14400(*)
Nug18	2174	12.64	0.00	100.00	2.39	14400(*)
P.ave.	1412.57	5.62	0.00	100.00	1.18	14400.00(*)
Had12	1654	0.12	894.00	45.88	0.27	14400(*)
Had14	2724	0.00	1300.50	52.26	0.87	14400(*)
Had16	3784	1.72	1530.78	58.85	1.44	14400(*)
Had18	5474	2.16	2144.88	59.97	4.15	14400(*)
Had20	7266	4.97	2827.08	59.16	9.58	14400(*)
P.ave.	4180.40	1.80	1739.45	55.22	3.26	14400(*)
G.ave	9192.07	6.18	6308.22	60.74	1.64	8741.94

Table3. The performance of the SLR solving procedure

Ins.	UB	*Opt.gap* (%)	LB	*LB gap* (%)	APY(%)	T.CPU time (sec.)	Nb. of iters.	CPU time of last iter. (sec.)
Chr18a	11098	0.00	11098	0.00	33.28	2.03	2	0.81
Chr18b	1534	0.00	1534	0.00	21.39	0.76	1	0.76
Chr20a	2192	0.00	2192	0.00	22.32	1.67	1	1.67
Chr20b	2298	0.00	2298	0.00	22.32	2.25	1	2.25
Chr20c	14142	0.00	14142	0.00	24.69	58.95	1	58.95
Chr22a	6156	0.00	6156	0.00	24.33	2.03	1	2.03
Chr22b	6194	0.00	6194	0.00	24.33	3.11	1	3.11
Chr25a	3796	0.00	3796	0.00	22.53	21.46	1	21.46
P.ave.	5926.25	0.00	5926.25	0.00	24.40	11.53	1.13	11.38
Esc32a	204	56.92	-	-	24.55	14400(*)	1	14400(*)
Esc32b	260	54.76	-	-	24.44	14400(*)	1	14400(*)
Esc32c	690	7.48	636	0.93	24.40	14400(*)	1	14400(*)
Esc32d	256	28.00	-	-	24.49	14400(*)	1	14400(*)
Esc32e	2	0.00	2	0.00	27.98	14400(*)	1	14400(*)
Esc32g	6	0.00	6	0.00	26.85	14400(*)	1	14400(*)
Esc32h	524	19.63	-	-	24.38	14400(*)	1	14400(*)
P.ave.	277.43	23.83	-	-	25.30	14400(*)	1.00	14400(*)
Scr12	31410	0.00	31410	0.00	35.56	66.74	2	44.61
Scr15	51140	0.00	50944	0.38	30.47	14551.9(*)	5	3465.56
Scr20	116272	5.67	107170	2.60	17.91	14400(*)	1	14400(*)
P.ave.	66274.00	1.89	62810.67	0.99	27.98	9673.90	2.67	5067.57
Nug12	578	0.00	578	0.00	49.10	18712.82	3	7039.89
Nug14	1210	19.33	554	45.36	20.13	14400(*)	1	14400(*)
Nug15	1354	17.74	476	58.61	17.00	14400(*)	1	14400(*)
Nug16a	1822	13.17	948	41.12	21.09	14400(*)	1	14400(*)
Nug16b	1554	25.32	552	55.48	19.23	14400(*)	1	14400(*)
Nug17	2154	24.36	754	56.47	17.53	14400(*)	1	14400(*)
Nug18	2422	25.49	942	51.19	19.65	14400(*)	1	14400(*)
P.ave.	1584.86	17.92	686.29	44.03	23.39	15083.74	1.29	13415.46
Had12	1654	0.12	1602	3.03	43.22	20902.76	3	7438.45
Had14	2824	3.67	2414	11.38	17.96	14400(*)	1	14400(*)
Had16	3786	1.77	3256	12.47	19.03	14400(*)	1	14400(*)
Had18	5880	9.74	4860	9.29	19.65	14400(*)	1	14400(*)
Had20	7202	4.05	6174	10.81	19.90	14400(*)	1	14400(*)
P.ave.	4269.20	3.87	3661.20	9.40	23.95	15741.21	1.40	12930.07
G.ave.	9353.80	10.57	-	-	24.66	10473.71	1.33	9155.25

6 Conclusion

In this paper the Semi-Lagrangian relaxation (SLR) was implemented to solve the quadratic assignment problem (QAP) for the first time. A dual ascent algorithm with finite convergence, intended to solve the QAP, was proposed based on the theoretical properties of the SLR and the QAP.

The results presented in this paper show that Cplex outperformed the proposed approach in most of the instances in terms of solution quality and CPU time. However, this paper shows some contributions. First contribution: in Table 3 we observe that the average percentage of unfixed variables y_{ijkl} (APY) is around 25%. Roughly speaking, this implies that one should be able to solve the QAP by eliminating around 75% of its variables by using the SLR method. Thus, the CPU time spent by Cplex in solving the reduced oracle problem (one iteration of Algorithm 1) is in general shorter than the one spent by Cplex in solving the original QAP as one would expect. However, we have observed in a few cases that to solve the oracle problem, generated via fixing some variables in advance, may take longer than to solve the original QAP. Second contribution: comparing the lower bounds reported in Tables 2 and 3 under labels LP Cost and LB, respectively, we observe that the reported SLR lower bounds are much stronger than the LP bounds as expected (under convergence the SLR lower bound closes the duality gap).

Therefore, it is possible that these two contributions were useful for future SLR approaches to solve the QAP problem. Finally, some aspects in the SLR approach here presented deserve further research in order to analyze possible improvements. For example, it is possible to improve the computational efficiency of the Algorithm 1 by choosing better initial dual point and efficiently updating it.

References

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

[2] L. Steinberg. The backboard wiring problem: A placement algorithm. *Siam Reviews*, 1961, 3(1):37–50.

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

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

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

[6] R. E. Burkard. Quadratic assignment problem. *Handbook of Combinatorial Optimization*, New York: Springer, 2013, 2741–2814.

[7] H.-Z. Zhang, L. Ma, C. Beltran-Royo. *The Quadratic Assignment Problem & Its Linearization Techniques (in Chinese)*. Shanghai: Shanghai People’s Publishing House, 2013.

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

[9] T. C. Koopmans, M. J. Beckmann.

Assignment problems and the location of economic activities. *Econometrica*, 1957, 25(1):53–76.

[10] M. Jünger, V. Kaibel. The QAP-polytope and the star transformation. *Discrete Applied Mathematics*, 2001, 111(3):283–306.

[11] M.W. Padberg, M. P. Rijal. *Location, Scheduling, Design and Integer Programming*. Boston: Kluwer Academic Publishers, 1996.

[12] M. Jünger, V. Kaibel. On the SQAP-Polytop. *Siam Journal on Optimization*, 2000, 11 (2):444–463.

[13] W. P. Adams, M. Guignard, P. M. Hahn, et al. A level-2 reformulation-linearization technique bound for the quadratic assignment problem. *European Journal of Operational Research*, 2007, 180(3):983–996.

[14] W. P. Adams, 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*, 1994, 16:43–75.

[15] B. Rostami, F. Malucelli. A revised reformulation-linearization technique for the quadratic assignment problem. *Discrete Optimization*, 2014, 14:97–103.

[16] Y. Xia. Gilmore-lawler bound of quadratic assignment problem. *Frontiers of Mathematics in China*, 2008, 3(1):109–118.

[17] Y. Xia, W. Gharibi. On improving convex quadratic programming relaxation for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2015, 30(3):647–667.

[18] R. Sotirov E. de Klerk. Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry. *Mathematical Programming*, 2012, 133(1-2):75–91.

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

[20] M. S. Hussin, T. Stutzle. Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances. *Computers & Operations Research*, 2014, 43(1):286–291.

[21] A. Misevicius. An implementation of the iterated tabu search algorithm for the quadratic assignment problem. *OR Spectrum*, 2012, 34(3):665–690.

[22] E. Duman, M. Uysal, A. F. Alkaya. Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences*, 2012, 217(1):65–77.

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

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

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

[26] M. Guignard. Lagrangean relaxation. *Top*, 2003, 11(2):151–228.

[27] I. Z. Milis, V. F. Magirou. A lagrangian relaxation algorithm for sparse quadratic assignment problems. *Operations Research Letters*, 1995, 17(2):69–76.

[28] A. A. Pessoa, P. M. Hahn, M. Guignard, et al. Algorithms for the generalized quadratic assignment problem combining lagrangean decomposition and the reformulation-linearization technique. *European Journal of Operational Research*, 2010, 206(1):54–63.

[29] C. Beltran, C. Tadonki, J.-Ph. Vial. Solving the p-median problem with a semi-Lagrangian relaxation. *Computational Optimization and Applications*, 2006, 35(2):239–260.

[30] C. Beltran-Royo, J.-P. Vial, A. Alonso-Ayuso. Semi-lagrangian relaxation applied to the uncapacitated facility location problem. *Computational Optimization and Applications*, 2012, 51(1):387–409.

[31] H.-Z. Zhang, C. Beltran-Royo, M. Constantino. Effective formulation reductions for the quadratic assignment problem. *Computers & Operations Research*, 2010, 37(11):2007–2016.

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

Huizhen Zhang was born in 1979. She received her B.S. degree from Shanxi University of Finance and Economics in 2002, and M.S. degree from Chengdu University of Technology in 2005, and Ph.D degree from University of Shanghai for Science and Technology (USST) in 2009. Now she is an associate professor of USST. Her research interests include operations research and intelligent optimization algorithms.
Email: zhzhzywz@gmail.com, zhzzywz@163.com

Cesar Beltran-Royo was born in 1968. He received his B.S. degree from University of Valencia in 1991, and Ph.D. degree from Polytechnic University of Catalonia in 2001. Now he is a professor of Rey Juan Carlos University, Spain. His main research interests include operations research and stochastic optimization.
Email: cesar.beltran@urjc.es

Bo Wang was born in 1960. He received Ph.D. degree from Beihang University in 2000. Now he is a professor in the department of system science at the University of Shanghai for Science and Technology. His main research interests include systems engineering, operations management, environmental management and decision analysis.
Email: Toddwang2000@126.com

Liang Ma was born in 1964. He received his B.S. degree from Fudan University in 1985, and Ph.D. degree from Shanghai Jiao Tong University in 2000. Now he is a professor and Ph.D. supervisor in the department of system science at the University of Shanghai for Science and Technology. His main research interests include systems engineering, operations research and intelligent optimization.
Email: maliang@usst.edu.cn

Ziying Zhang was born in 1978. He received his B.S. degree from Hubei Engineering University, and M.S. degree from Zhengzhou University in 2007, and the Ph.D. degree from Fudan University in 2013. Now he is working at Shanghai University of Engineering Science as an associate professor. His main research interests include computational simulation of structure and physical properties of crystal material.
Email: yingzz25@126.com