CIRRELT

CIRRELT-2019-58

A Parallel Iterated Tabu Search Applied to Several Quadratic Assignment Problems

Allyson Silva
Leandro C. Coelho
Maryam Darvish

December 2019

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2019-020

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél. : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél. : 1-418-656-2073
Télécopie : 1-418-656-2624

A Parallel Iterated Tabu Search Applied to Several Quadratic Assignment Problems

Allyson Silva*, Leandro C. Coelho, Maryam Darvish

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, 2325 de la Terrasse, Québec, Canada G1V 0A6

Abstract. In the Quadratic Assignment Problem (QAP), facilities are assigned to sites in order to minimize interactions between pairs of facilities. Although easy to define, it is considered as one of the hardest problems in combinatorial optimization, due to its non-linear nature. After decades of research on the QAP, many variations of this problem arose to deal with different applications. Along with the QAP, we consider in this study four variants - the Quadratic Bottleneck Assignment Problem, the Biquadratic Assignment Problem, the Quadratic Semi-Assignment Problem, and the Generalized QAP - and develop a single framework to solve them. Our multi-start iterated tabu search (MS-ITS) is based on the most successful heuristics to solve the QAP. It combines the diversification phase of generating new local optima found after modifying a solution, using an operator that performs random swaps of facilities, with the intensification phase of a simple, yet effective, tabu search. The search is speed up through several solutions being improved concurrently using parallelism, and a convergence criteria determines whether the search stops according to the current best solutions in each parallel search. Computational experiments using the hardest benchmark instances from the literature attest the effectiveness of the MS-ITS, showing its competitiveness when compared to state-of-the-art methods to solve this problem. We also show that our MS-ITS significantly outperforms the best methods found for all four variants of the QAP, significantly improving their literature.

Keywords. Assignment, quadratic assignment, hybrid metaheuristic, tabu search, parallel computing.

Acknowledgements. The authors thank Leonidas Pitsoulis for kindly providing us their algorithm for comparative purposes. We also thank Compute Canada for providing high-performance parallel computing facilities. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 2019-00094. This support is gratefully acknowledged.

---

* Corresponding author: allyson.silva@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2019

1. Introduction

Consider an assignment problem where facilities have to be located in sites. Each facility can be located in one site and each site can hold exactly one facility. The distances between sites and the flows between facilities are given. The Quadratic Assignment Problem (QAP) consists in minimizing a cost function expressed by the distance and the flow between each pair of facilities located in their respective sites.

The QAP was introduced in 1957 by Koopmans and Beckmann [49]. It is a well-known and well-studied problem, being considered as one of the most difficult ones to solve in combinatorial optimization. It belongs to the class of NP-complete problems, meaning that no known polynomial time algorithm exists to solve it up to a proven optimality. Even finding an approximate solution to the QAP is hard [79]. This difficulty is mainly due to the quadratic form of its objective function in addition to its combinatorial nature.

Several problems related to the QAP appeared in literature in the following years. The Quadratic Bottleneck Assignment Problem (QBAP) was proposed in 1961 by Steinberg [86], and it aims to minimize the maximum cost between two facilities instead of the overall network cost. Two years later, the Biquadratic Assignment Problem (BiQAP) was proposed by Lawler [51]. It is essentially a QAP in which the interactions occur between four facilities simultaneously instead of two. The Quadratic Semi-Assignment Problem (QSAP), proposed in 1969 by Greenberb [40], relaxes the constraints that all sites have to be assigned to exactly one facility, allowing the number of potential sites to be different than the number of facilities. A relatively new problem is the Generalized QAP (GQAP), proposed in 2003 by Lee and Ma [53], which generalizes the QAP by considering that multiple facilities can be located in a site, if enough resources are available. Other less used and studied variants also exist [48, 76, 78, 85].

A number of practical problems can be formulated as a QAP or one of its variants. The design of typewriter keyboards [77] or keyboards on touchscreen devices [25] can be formulated as a QAP in order to arrange the keys to minimize the time needed to write some text. The campus planning problem [26] and the hospital layout problem [32] are essentially QAPs where buildings have to be located in the available sites in order to minimize the total walking distance between the buildings. The backboard wiring problem is solved using a QBAP to minimize the length of the longest wire that connects controls and displays [86]. A GQAP can be used to design a university curriculum [91].

Other applications include the molecular conformation problem [75], dartboard design [31], zoning in forests [11], computer motherboard design [65], and presidential campaign itinerary planning [52]. Perhaps the most popular applications arise in logistics and facility layout. Cordeau et al. [22] use the GQAP to manage containers in a storage yard. Fu and Kaku [36] and Benjaafar [6] show that the QAP can be used to design a facility layout in order to minimize work-in-process inventories. Chiang et al. [20] use a BiQAP variant as a quantitative attempt to minimize workflow interferences in facility layout design. Krarup and Pruzan [50] describe the QAP as being equivalent to the layout planning problem. The QAP can also be adapted to formulate other fundamental problems such as the traveling salesman problem, the graph partitioning problem and the maximal clique problem [73], while the QSAP is used for optimizing clustering [43] and scheduling [57] problems.

In their review, Abdel-Basset et al. [1] show that the majority of exact approaches for QAPs use some sort of branch-and-bound related algorithm. First schemes of this technique are found in Gilmore [38] and Lawler [51] and recent advances in Hahn et al. [41]. Other exact approaches use cutting plane methods [5, 65], branch-and-cut [33], and dynamic programming [21]. The calculation of lower bounds for QAPs is also a topic of interest in research. Some examples of techniques are the Gilmore-Lawler bound [38], eigenvalue related bound [34], bounds based on linear programming [2, 27], and bounds based on reformulations [82]. The presented methods are used for the QAP, but they can be easily adapted for its variants.

Exact methods usually fail to prove solution optimality within a reasonable time for instances of considerable size. An alternative approach to deal with these instances is the use of heuristics and metaheuristics, such as simulated annealing [66], genetic algorithms [29], scatter search [23], ant colony [59, 89], neural networks [12], particle swarm [54], greedy randomized adaptive search procedure (GRASP) [71], variable neighborhood search [37], and tabu search [30]. Recent trends in the development of heuristic procedures to solve QAPs are the combination of two or more metaheuristics, creating a hybrid algorithm, most of which use a diversification-intensification approach. Because of the good performance of tabu search in QAPs, several hybrid heuristics use it in this phase. For example, Chiang and Chiang [19] and Misevičius [67] incorporate tabu search to simulated annealing, Hasegawa et al. [45] use tabu search with a neural network, and Youssef et al. [92] hybridize tabu search, simulated annealing and fuzzy logic. A popular combination is tabu search with genetic algorithm [4, 28, 35]. When genetic algorithms are hybridized with local searches and have all solutions improved by them, they are known as memetic algorithms. Moreover, some methods based on

memetic algorithms are found in Merz and Freisleben [63] and Ostrowski and Ruoppila [72]. Memetic algorithms are highly parallelizable, which can potentially linearly reduce the running time of the algorithm with the number of available cores. Even when no genetic operators are applied, but solutions are evolved in parallel, parallelism has potential to speed up the search. Several studies applied parallelism to solve the QAP [44, 46, 69, 90].

The QAP has received considerable attention during the last decades, as shown by the several QAP books and reviews (see Abdel-Basset et al. [1], Burkard [14], Finke et al. [34] and Loiola et al. [55]). However, the literature for its variants is outdated, and modern methods were not yet tested for most of them.

In this paper, we present a multi-start iterated tabu search (MS-ITS) metaheuristic to solve the QAP, QBAP, BiQAP, QSAP and GQAP. The method iterates between diversification-intensification phases to evolve multiple solutions, performing random modifications to them followed by local searches and a tabu search to reach new local optima. Since the intensification phase is too time consuming, we use parallelism to apply multiple tabu searches simultaneously to the solutions evolving in parallel. Our MS-ITS has a different structure than similar approaches for this problem. We show that our framework is robust enough to be adapted to solve all five problems requiring no major modifications. Computational experiments using benchmark instances attest the effectiveness of the MS-ITS, significantly updating the state-of-the-art methods of the QAP variants. Our method has the best average performance compared to the other heuristics for a hard set of benchmark instances for the QAP, finding solutions on average at 0.03% from best known solutions (BKS). For the BiQAP, QSAP, and GQAP, we find the BKS in all instances tested performing better than the state-of-the-art methods used as comparison, either heuristics or exact methods. For the QBAP, we show that MS-ITS outperforms a state-of-the-art exact method especially in larger instances.

Another contribution of this paper is providing a literature review of the most relevant research related to the QAP and its variants. We survey the most relevant formulations, solution algorithms, instances and solutions for the five variants addressed in this paper.

The remainder of this paper is organized as follows. Section 2 gives a formal presentation of the five problems considered by presenting their mathematical formulations and a description of related studies to each one. Section 3 details the MS-ITS. The results of computational experiments for each of the problems are presented in Section 4. Finally, Section 5 presents our concluding remarks.

2. Mathematical formulations and literature review

The formulations for the QAP, QBAP, BiQAP, QSAP, and GQAP are presented next.

2.1. Quadratic Assignment Problem

Let n be the number of facilities to be assigned to n sites, where $\mathcal{N} = \{1, \ldots, n\}$ represents the set of such facilities and sites. Two $n \times n$ matrices are given. $\mathbf{F} = (f_{ij})$ represents the flow between facilities i and j, and $\mathbf{D} = (d_{kl})$ represents the distance between sites k and l. A QAP solution may be represented by a permutation of $\mathcal{N}$, where $p(i)$ is the position of the facility i in the permutation. The Koopmans-Beckman QAP formulation [49] is represented as:

$$\min_{p \in \Pi_{\mathcal{N}}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} f_{ij} d_{p(i)p(j)}, \tag{1}$$

where Π_N is the set of all permutations of $\mathcal{N}$. Each single product of f and d is the cost of assigning facilities i and j to sites $p(i)$ and $p(j)$, respectively. This formulation can be transformed to a quadratic 0–1 formulation using a permutation matrix $\mathbf{X} = (x_{ij})$, such that:

$$x_{ik} = \begin{cases} 1, & \text{if } p(i) = k, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The matrix $\mathbf{X}$ is characterized by the following assignment constraints:

$$\sum_{i \in \mathcal{N}} x_{ik} = 1, \quad \forall k \in \mathcal{N}, \tag{3}$$

$$\sum_{k \in \mathcal{N}} x_{ik} = 1, \quad \forall i \in \mathcal{N}, \tag{4}$$

$$x_{ik} \in \{0, 1\}, \quad \forall i, k \in \mathcal{N}. \tag{5}$$

Hence, the QAP is formulated as:

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{N}} \sum_{l \in \mathcal{N}} f_{ij} d_{kl} x_{ik} x_{jl}, \tag{6}$$

subject to constraints (3)–(5). Other formulations for the QAP exist [1, 14, 55, 70], and it can be solved in polynomial time for some special matrix structures [18]. Although the literature on exact approaches for the QAP is vast, benchmark instances with as few as 30 nodes are yet open to be solved [17].

A better alternative to deal with larger problems is the development of heuristics. Those recent effective heuristics similar to our MS-ITS to solve the QAP are described next. Misevičius [68] presents an ITS where several cycles of diversification and intensification occur between a complex mutation operator mechanism and a tabu search. The memetic algorithm proposed by Benlic and Hao [8] integrates a special local search procedure, known as Breakout Local Search (BLS), with an evolutionary approach that uses genetic algorithm operators to create a pool of candidate solutions to be improved by the local search. Tosun [90] presents a parallel algorithm that hybridizes a genetic algorithm and a robust tabu search, using parallelism to speed up the latter. In Aksan et al. [3], a parallel BLS is presented where parallel solutions evolve and are improved using BLS. These methods are tested on several instances from the QAPLIB library [17], having very similar performances, and they are among the best methods found to solve the QAP.

2.2. Quadratic Bottleneck Assignment Problem

In the bottleneck version of the QAP, the same inputs for the original problem are given. The objective, however, is to find a permutation $p \in \Pi_{\mathcal{N}}$ such that:

$$\min_{p \in \Pi_{\mathcal{N}}} \max_{i,j \in \mathcal{N}} f_{ij} d_{p(i)p(j)}, \tag{7}$$

which can be reformulated with a linear objective function by introducing a slack variable Z as:

$$\min Z, \tag{8}$$

subject to (3)–(5), and to

$$Z - f_{ij} d_{kl} x_{ik} x_{jl} \geq 0, \quad \forall i, j, k, l \in \mathcal{N}. \tag{9}$$

The above formulation can yet be converted to a mixed integer linear programming (MILP) model using the standard linearization technique [39] to replace the quadratic terms in constraints (9) by

the linearization variables $z_{ikjl} = x_{ik}x_{jl}$, i.e., z_{ijkl} is equal to one if facility i is assigned to site k and facility j is assigned to site l, and zero otherwise. Thus, (9) is replaced by the linear constraints:

$$Z - f_{ij}d_{kl}z_{ikjl} \geq 0, \quad \forall i, j, k, l \in \mathcal{N}, \tag{10}$$

$$z_{ikjl} \leq x_{ik}, \quad \forall i, j, k, l \in \mathcal{N}, \tag{11}$$

$$z_{ikjl} \leq x_{jl}, \quad \forall i, j, k, l \in \mathcal{N}, \tag{12}$$

$$z_{ikjl} \geq x_{ik} + x_{jl} - 1, \quad \forall i, j, k, l \in \mathcal{N}. \tag{13}$$

The literature of the QBAP is very limited. Burkard [13] states that the QBAP is NP-hard since it contains the bottleneck travelling salesman problem as a special case. He also proves the asymptotic behavior of the problem. Although the QBAP can be solved using the same QAP algorithms, we could not find any study with computational results for the QBAP. Steinberg [86] introduces QBAP for a wiring problem, but unfortunately he does not solve it, only presenting the results for the given backboard wiring instance as a QAP.

2.3. Biquadratic Assignment Problem

In the BiQAP, we are given two four-dimensional arrays as input, $A = (a_{ijkl})$ and $B = (b_{p(i)p(j)p(k)p(l)})$, where a_{ijkl} is the QAP flow-equivalent parameter for the facilities i, j, k and l, and $b_{p(i)p(j)p(k)p(l)}$ is the distance-equivalent parameter for the sites $p(i)$, $p(j)$, $p(k)$ and $p(l)$. The BiQAP is formulated in the Koopmans-Beckmann form [16] as:

$$\min_{p \in \Pi_{\mathcal{N}}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{N}} \sum_{l \in \mathcal{N}} a_{ijkl} b_{p(i)p(j)p(k)p(l)}. \tag{14}$$

N-adic variants of the QAP, such as the cubic and quartic versions – as the BiQAP is also called – are generalizations of the QAP, therefore, they are also NP-hard. Although the BiQAP has been considered since the 1960s, it was formally studied for the first time several decades later in Burkard et al. [16], who present different formulations for the problem and show how to compute lower bounds for the optimal solution value, derived from lower bounds for the QAP. After computational tests for instances with known optimal solutions, they conclude that the quality of the bounds is not very good and deteriorates when the problem size increases. This suggests that exact methods will only be effective on very small instances. Finally, the asymptotic behavior of the BiQAP is proved by showing

that the ratio between the worst and optimal solution values tends to one as the size of the problem increases.

Due to the difficulty to solve the BiQAP exactly, some heuristics have been proposed. Burkard and Çela [15] develop and compare variants of simulated annealing, tabu search and other improvement methods, particularly using pairwise exchange algorithms (detailed in Section 3.3.1). Computational experiments using the same instances as in Burkard et al. [16] suggest that one version of the simulated annealing is the best among the variants tested. Later, Mavridou et al. [61] describe a GRASP for the BiQAP that iteratively constructs greedy solutions by making the first four assignments simultaneously using their cost of interaction, then assigning the other facilities one by one, and uses the pairwise exchange algorithm to improve the solution. This method was able to find the optimal solution for all the instances used in Burkard and Çela [15] improving the performance of the simulated annealing, however, requiring large CPU times.

2.4. Quadratic Semi-Assignment Problem

In the QSAP, we are given a possible distinct number of facilities and sites, where $\mathcal{N} = \{1, \ldots, n\}$ is the set of facilities and $\mathcal{M} = \{1, \ldots, m\}$ is the set of sites. Semi-assignment constraints state that only facilities should be assigned to sites, but sites can stay unassigned. Also, there is no constraint on the number of facilities assigned to each site. A location cost matrix $\mathbf{C} = (c_{ik})$, where each value represents the cost of assigning facility i to site k, is usually considered. The quadratic 0-1 formulation of the QSAP is given as:

$$\min \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{M}} c_{ik} x_{ik} + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{M}} \sum_{l \in \mathcal{M}} f_{ij} d_{kl} x_{ik} x_{jl}, \tag{15}$$

subject to

$$\sum_{k \in \mathcal{M}} x_{ik} = 1, \quad i \in \mathcal{N}, \tag{16}$$

$$x_{ik} \in \{0, 1\}, \quad i \in \mathcal{N}, k \in \mathcal{M}. \tag{17}$$

The QSAP NP-hardness is proven in Sahni and Gonzalez [79]. Malucelli [57] and Malucelli and Pretolani [58] provide lower bounds for the QSAP and present certain polynomial solvable cases. Milis and Magirou [64] propose a Lagrangean relaxation algorithm for this problem and show that

even for a QSAP of small size, it is hard to find optimal solutions. Reformulation techniques are also a known approach used to solve this problem. Saito et al. [80] study the polytope that arises from a MILP reformulation of the problem. The reformulation-linearization technique (RLT), presented in Billionnet and Elloumi [9], linearizes the QSAP. They show that the best reduction of the QSAP is obtained by solving a continuous relaxation of their RLT. Schüle et al. [81] provide RLT formulations with different levels for the QSAP and analyze their tightness. The level-1 RLT model from Schüle et al. [81] is presented next.

Consider linearization variables z_{ijkl} and a set $\mathcal{N}' = \{j \in \mathcal{N} : j > i\}$. The level-1 RLT model is as follows:

$$\min \sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{M}} c_{ik}x_{ik} + \sum_{i\in\mathcal{N}\setminus\{N\}} \sum_{j\in\mathcal{N}'} \sum_{k\in\mathcal{M}} \sum_{l\in\mathcal{M}} (f_{ij} + f_{ji})d_{kl}z_{ikjl}, \tag{18}$$

subject to (16), (17), and to

$$\sum_{l\in\mathcal{M}} z_{ikjl} = x_{ik}, \quad \forall i \in \mathcal{N} \setminus \{N\}, \forall k \in \mathcal{M}, \forall j \in \mathcal{N}', \tag{19}$$

$$\sum_{k\in\mathcal{M}} z_{ikjl} = x_{jl}, \quad \forall i \in \mathcal{N} \setminus \{N\}, \forall j \in \mathcal{N}', \forall l \in \mathcal{M}, \tag{20}$$

$$0 \leq z_{ikjl} \leq 1, \quad \forall i \in \mathcal{N} \setminus \{N\}, \forall k \in \mathcal{M}, \forall j \in \mathcal{N}', \forall l \in \mathcal{M}. \tag{21}$$

The QSAP has many applications in scheduling, clustering, and partitioning problems [43, 56, 84]. Some task-assignment problems can be formulated as QSAPs [10]. Also, the hub location [80] and the metric labeling [47] problems are special cases of the QSAP. Although heuristic methods for these particular problems can be adapted to solve the QSAP, we could not find any published work on it.

2.5. Generalized Quadratic Assignment Problem

The GQAP is the capacitated version of the QSAP. The arrays r_i, indicating the size of a facility i, and C_j, indicating the capacity of a site j, are given. The formulation of the GQAP, as presented in Lee and Ma [53], considers objective function (15) subject to (16), (17) and to capacity constraints given as:

$$\sum_{i\in\mathcal{N}} r_i x_{ik} \leq C_k, \quad k \in \mathcal{M}. \tag{22}$$

Once again, only small instances could be solved to optimality in the GQAP. Lee and Ma [53] present three linearization approaches and a branch-and-bound algorithm. Hahn et al. [42] describe a level-1 RLT with a dual ascent procedure in a branch-and-bound scheme to prove optimality for instances adapted from another problem. Pessoa et al. [74] find new lower bounds for the same instances using Lagrangean relaxations of the RLT formulation from Hahn et al. [42].

Heuristic approaches are also presented for the GQAP. Cordeau et al. [22] present a memetic heuristic that combines genetic algorithms and tabu search, which solves Lee's instances faster and introducing larger instances. Mateus et al. [60] propose several GRASP with path-relinking heuristics using different construction, local search, and path-relinking procedures. Later, McKendall and Li [62] propose a simple tabu search heuristic to solve Cordeau's instances, and compare its performance with the other heuristics.

3. Multi-start iterated tabu search

We now present our multi-start iterated tabu search (MS-ITS) metaheuristic. Following the successful diversification-intensification approach for QAPs, we integrate a diversification phase where the solution is modified until a new local optimum is found using local searches, then it is improved in an intensification phase using a tabu search adapted from the QAP for its variants. Multiple start points are created using a constructive heuristic based on the random assignment of facilities to sites, which are improved simultaneously, thus maximizing the potential benefits of the use of parallel computing.

3.1. Solution representation

A solution x for any of the five problems considered can be encoded as a vector $x = \{p(i)\}$, $\forall i \in \mathcal{N}$, where $p(i)$ represents the site where facility i is installed. The five QAP variants considered here are split into two sets. The permutation problems are those in which x is represented by a permutation of the sites indices. The QAP, QBAP and BiQAP belong to this set. Meanwhile, the non-permutation problems are those that a site index can appear multiple times in x. This set contains the QSAP and GQAP.

3.2. Constructive heuristics

Three different heuristics are implemented to create initial solutions. On the permutation problems, we simply generate a random permutation of the sites, assigning each one to a facility. On the QSAP,

every facility is assigned to a random site, until all facilities are located. To generate a feasible solution for the GQAP, additional steps are required to respect capacity constraints. The steps performed are as follows:

1. Generate a queue of unassigned facilities in a random order;
2. Assign the first facility in the queue to a randomly drawn site. If there is enough capacity available, the facility stays assigned to the site and is removed from the queue. Otherwise, draw another site, different than the ones already selected. Repeat this until the facility is successfully assigned to a site or all sites are checked;
3. If the facility cannot fit in any site, remove a random facility already assigned from its site and place it in the end of the queue. Then, recheck if the current facility fits in this site. Repeat this step until the current facility can be assigned to a site;
4. Repeat steps 2 and 3 until the queue of unassigned facilities is empty.

This procedure is always capable of generating a feasible solution, if it exists. In the first two, a feasible solution is generated in time $O(n)$ for an instance with n facilities. However, for the GQAP heuristic, if the feasible solution space is too restricted, such as in the case when capacity constraints are tight, the process of removing assigned facilities to give space to unassigned ones can take extra time to generate a solution for the problem. We present next local search heuristics to improve feasible solutions, which are embedded within the proposed MS-ITS metaheuristic.

3.3. Local searches

Local search heuristic is a technique used to improve a solution by searching in the neighborhood of its solution space. In the proposed MS-ITS, we implement two of the most common procedures used in local searches for the QAP and its variants. The first one is well suited for all problems studied here and is based on swapping the positions of two facilities. The second one is applicable only to the non-permutation problems (QSAP and GQAP), and consists of the movement of facilities between sites.

3.3.1. Pairwise exchange local search

Local search 1 (LS1) performs the *pairwise exchange* procedure [35, 83]. If solution $x = \{p(i)\}, \forall i \in \mathcal{N}$, is a starting point for LS1, the neighbor $x' = \{q(i)\}$, $\forall i \in \mathcal{N}$, obtained by the swap of facilities of

indices r and s is:

$$q(i) = p(i), \quad \forall i \in \mathcal{N} \setminus \{r, s\},$$
$$q(r) = p(s),$$
$$q(s) = p(r).$$

The neighborhood of LS1 ($\mathbf{N}_1(x)$) contains the solutions generated by all swaps between r and s such that $r < s$. The objective of LS1 is to find the best improving solution x^* contained in the neighborhood of x, i.e., $x^* = \{\min f(x') | f(x') < f(x), \forall x \in \mathbf{N}_1(x)\}$, also known as steepest descent. In our heuristic, an exhaustive search is performed, meaning that every time a solution is improved a new local search is performed starting from this new solution. The local search stops when no improved solution can be found in a solution neighborhood.

In the permutation problems, all swaps will result in a feasible and distinct solution, meaning that a neighborhood contains exactly $\frac{n(n-1)}{2}$ solutions. In the QSAP, when two different facilities are located in the same site, their swap does not change the current solution. So, LS1 neighborhood size in this problem is bounded by $\frac{n(n-1)}{2}$; the actual size depends on the number of distinct sites where facilities are located. For example, in the case that all facilities are assigned to the same site, then $\mathbf{N}_1(x) = \{\emptyset\}$. In the GQAP, the neighborhood size can be even smaller. Beyond distinct locations between swapped facilities, we should also consider if the swap results in a feasible solution. In a swap between any pair of facilities i and j with capacities $r_i \geq r_j$ the neighbor generated is feasible if $r_i - r_j \leq C_{p(j)} - \sum_{k|p(k)=p(j)} r_k$, i.e., if there is enough capacity left in the site to accommodate the extra space of the new facility it is receiving.

In each iteration of LS1, we are interested in calculating the difference $\Delta(x, r, s) = f(x') - f(x)$, where $x' \in \mathbf{N}_1(x)$ is obtained by swapping facilities r and s in the current solution x. The simplest way to calculate $\Delta(x, r, s)$ is by calculating separately $f(x)$ and $f(x')$, then performing the subtraction. For the QAP, this would take $O(n^2)$ because the calculation of $f(x)$ for any feasible solution x takes $O(n^2)$ time. However, when using the pairwise exchange local search for the QAP, one particular effect is that the cost of a neighbor solution can be computed in $O(n)$ operations, instead of $O(n^2)$, as follows

[7, 87]:

$$\begin{aligned}\Delta(x,r,s) =&(f_{rr}-f_{ss})(d_{p(s)p(s)}-d_{p(r)p(r)})+(f_{rs}-f_{sr})(d_{p(s)p(r)}-d_{p(r)p(s)})+\\ &\sum_{i\in\mathcal{N}\setminus\{r,s\}}((f_{is}-f_{ir})(d_{p(i)p(s)}-d_{p(i)p(r)})+(f_{si}-f_{ri})(d_{p(s)p(i)}-d_{p(r)p(i)})).\end{aligned} \tag{23}$$

Taillard [87] proposed to use memory to store the values $\Delta(x,r,s)$ so that if x' is the solution obtained after swapping the sites of facilities r and s in the solution x, then computing $\Delta(x',u,v)$ is even faster if pairs $\{r,s\}$ and $\{u,v\}$ are mutually exclusive, i.e., $\{r,s\}\cup\{u,v\}=\{\emptyset\}$. Thus, $\Delta(x',u,v)$ is calculated in $O(1)$ as:

$$\begin{aligned}\Delta(x',u,v) =&(f_{su}-f_{sv}+f_{rv}-f_{ru})(d_{q(s)q(u)}-d_{q(s)q(v)}+d_{q(r)q(v)}-d_{q(r)q(u)})+\\ &(f_{us}-f_{vs}+f_{vr}-f_{ur})(d_{q(u)q(s)}-d_{q(v)q(s)}+d_{q(v)q(r)}-d_{q(u)q(r)})+\\ &\Delta(x,u,v).\end{aligned} \tag{24}$$

Equations (23) and (24) can be reduced even further if the flow matrix $\mathbf{F}$ and/or the distance matrix $\mathbf{D}$ are symmetric. For example, when both matrices $\mathbf{F}$ and $\mathbf{D}$ are symmetric and have a null diagonal, (23) is reduced to:

$$\Delta(x,r,s)=\sum_{i\in\mathcal{N}\setminus\{r,s\}}2(f_{ri}-f_{si})(d_{p(r)p(i)}-d_{p(s)p(i)}). \tag{25}$$

Finally, (24) is reduced in the symmetric case to:

$$\Delta(x',u,v) =2(f_{su}-f_{sv}+f_{rv}-f_{ru})(d_{q(s)q(u)}-d_{q(s)q(v)}+d_{q(r)q(v)}-d_{q(r)q(u)})+\Delta(x,u,v). \tag{26}$$

The same reductions are applicable to the BiQAP, as shown in Burkard and Çela [15]. The main difference is that the BiQAP objective function is reduced from $O(n^4)$ to $O(n^3)$ time by calculating the difference between a solution and its neighbor, and to $O(n^2)$ by storing the values $\Delta(x,r,s)$ to calculate neighbor objective functions after the first iteration. We refer the reader to Burkard and Çela [15] to see the full detailed reduced equations for the BiQAP.

3.3.2. Drop/add local search

Local search 2 (LS2) performs drop and add procedures to generate neighbor solutions for the non-permutation problems. This represents the exchange of the location of a facility. The neighborhood $\mathbf{N}_2(x)$ of a solution x explored by LS2 contains all solutions where facility $i \in \mathcal{N}$ is moved from its current site $p(i)$ to another site $k \in \mathcal{M} \setminus \{p(i)\}$.

In the QSAP, every facility movement will result in a feasible and distinct solution. Since each facility can be moved to $m-1$ different sites, LS2 neighborhood of a QSAP solution contains exactly $n(m-1)$ solutions. In the GQAP, it is necessary to verify capacity restrictions to ensure the procedure results in a feasible solution.

3.4. Tabu search

The tabu search (TS) is undoubtedly the most popular and successful metaheuristic to solve the QAP. It consists in allowing solutions to move to the least degrading solution when no improving neighbor exists. In order to avoid cycling, the return to the previous solution is forbidden. A data structure called *tabu list* is maintained indicating all forbidden moves. This list contains elements defining forbidden moves. In the QAP, a forbidden move can be defined by the pairs of facilities recently swapped [83] or by prohibiting moves where both facilities would be assigned to sites they had occupied in recent iterations [87]. In our tabu list, we simply save all solutions previously visited during the current call of the search, thus eliminating parameters such as the tabu list size and the number of iterations to keep a solution as tabu, as used in other works. Note that this might increase the complexity order to verify if a solution is in the tabu list, but it prevents the same solution being visited more than once during a single search. The only parameter to be set is the stopping criterion, chosen as the maximum number of iterations I performed by the TS without improvement in the best solution. The improving mechanism used is the pairwise exchange local search (LS1). Therefore, we can summarize the steps of our TS as:

1. Search for the best solution that is not in the tabu list using LS1 and change the current solution to it;
2. Add the new current solution to the tabu list;
3. If the current solution is the best one found so far, restart the iterations counter;
4. Repeat steps 1–3 until the iterations counter reaches I.

The steps described are for a single start point run. Tabu searches for QAPs can also be run in parallel. A concurrent and independent search starting from different initial solutions is proposed by Taillard [87]. In his approach, no information is exchanged during the search of the parallel TS algorithm and the best solution is determined at the end of the execution. An alternative to the independent search strategy is the cooperative multi-thread heuristic where the exchanging of information between solutions run in parallel is allowed [46, 88]. Recent parallel TS for the QAP, either independent or cooperative, are found in Aksan et al. [3], Czapiński [24], and Tosun [90].

The MS-ITS proposed in this study consists of the parallel execution of independent TS in multiple start points. The pseudocode of the MS-ITS is presented in Algorithm 1. For each of the P parallel start points, we are interested in three solutions: the current solution (x), the previous local minimum (x'), and the best solution found so far (x^*). The first step of MS-ITS consists in creating a feasible solution for each start point using the constructive heuristics (line 5). Then, the initial solution is improved using the tabu search represented in the algorithm as $TS(x)$, where x is the initial solution for the search (lines 6–8). In this first iteration, all three solutions are equal, i.e., $x = x' = x^*$ (line 9). The perturbation phase comes next. A diversification operator is adapted from Talbi and Bachelet [88] where a controlled number of random swaps of facilities is performed in the current solution to define the starting point of the next search. For this reason, a $Swaps(x, s)$ operator is defined (line 13). It modifies the solution x by performing s distinct swaps of facility locations. The new solution is improved using the local searches LS1 and LS2, respectively and whenever applicable, represented in the pseudocode as $LS(x)$, to reach a local minimum (line 14). If the current local minimum x is different than the previous local minimum x' found in the previous iteration, then the tabu search $TS(x)$ is applied to improve x (line 16). Otherwise, we increase the number of random swaps performed to move the current solution further away from the previous local minimum (line 18), repeating the perturbation process. The best solution found in the thread x^* is updated whenever an improved solution is found (lines 20–22). In our MS-ITS, we consider up to three stopping criteria, depending on the resources available to find the solution (line 10). The first is the maximum number of iterations, i.e., the number of times the perturbation process is repeated. The second is the maximum running time. Finally, we observed in preliminary experiments that when multiple solutions converge from different start points to a similar local optimum, there is a very high chance that this solution is the global optimum. Thus, we defined a third stopping criteria stating that if at least 50% of the solutions in $\mathcal{X}^*$ are equal to the best solution found so far, then the search stops and the algorithm returns the best solution in $\mathcal{X}^*$. This last criterion is best suited to use when P is high enough so that more

solutions need to converge to avoid a false positive.

Algorithm 1 Multi-start iterative tabu search (MS-ITS)

```
Number of parallel start points: P;
Set of best solutions: X* = {x*_p | p = {1,...,P}};
Set of current local optimum solutions: X' = {x'_p | p = {1,...,P}};
Set of modified solutions: X = {x_p | p = {1,...,P}};
Create X* using the constructive heuristics;
for all parallel p ∈ P do
    x*_p ← TS(x*_p);                                   // Perform TS on each parallel solution
end for
X' ← X* and X ← X*;
while stopping criteria are not satisfied do
    s_p = 1, ∀p = {1,...,P}                            // Reset number of random swaps
    for all parallel p ∈ P do
        x_p ← Swaps(x'_p, s_p);                        // Perturb solution using s_p random swaps of facilities
        x_p ← LS(x_p);                                 // Search for local minimum
        if f(x_p) ≠ f(x'_p) then
            x'_p ← TS(x'_p);                           // Perform TS to each solution
        else
            s_p ← s_p + 1;                             // Number of swaps is increased
        end if
        if f(x'_p) < f(x*_p) then
            x*_p ← x'_p;                               // Update best solution
        end if
    end for
end while
return best solution in X*.
```

4. Computational experiments

This section provides the results obtained from extensive computational experiments performed to setup our MS-ITS and to compare its performance with state-of-the-art methods to solve the QAP and its variants. We used computers equipped with an Intel Gold 6148 Skylake CPU with a 2.4 GHz clock, 8 GB of RAM and 40 threads. The MS-ITS was implemented in C++, while the exact models were solved using CPLEX 12.5. The parallelism was implemented using OpenMP. We note that the MS-ITS is not a memory-intensive algorithm, but we set the memory limit for CPLEX. All instances and results are available from `https://www.leandro-coelho.com/quadratic-assignment-problems/`.

4.1. Test instances

The QAP is solved using three sets of classical instances from the QAPLIB benchmark library. They are the *Taia* (unstructured, randomly generated), *Taib* (real-life-like instances) and *Sko* (instances with grid-based distances) sets. Each instance is labeled using the name of the set and the number of facilities contained in it. Together, the three sets consist of 31 problem instances with up to 100

facilities and sites. Although the library provides more instances, the others are easily solvable by many existing methods, thus they are not considered here. We did not find any instance set specifically designed for the QBAP. Thus, we used the same QAP sets for this problem. For the BiQAP, the original instances used in Mavridou et al. [61] are reported to be lost after contact with one of the authors. Thus, we generated 12 new instances using a random uniform distribution between 1 and 9 to determine each distance between sites and each flow between facilities considering both as symmetric matrices. The new instances have similar sizes as those from Mavridou et al. [61], i.e., 10 to 36 sites and facilities. The GQAP is solved using a set of 21 instances taken from Cordeau et al. [22], with up to 50 facilities and 20 sites. These instances are labeled with a code consisting of three parts: the number of facilities, the number of sites, and the tightness of the capacity constraints. Other smaller instances for this problem are found in Mateus et al. [60], but they are easily solved using the existing methods, thus they also are not considered here. For the QSAP, we consider two sets of instances. Set 1 is an adaptation from the 21 instances of Cordeau et al. [22] for the GQAP with no facility capacities. In these instances, distances between a facility to itself is zero. Since sites in the QSAP are uncapacitated, this would result in a solution with all facilities located in a single site, which gives a solution equal to zero. To prevent this, we set d_{jj} for $j \in \mathcal{M}$ to be equal to 50% of the average graph distance. Set 2 has 20 instances generated as follows. Points in a Euclidean space of size 100×100 are generated where 70% of them represent facilities and the remaining represent sites. Instances are generated for $p = \{50, 75, 100, 125, 150\}$ points. The flow matrix $\mathbf{F}$ is generated uniformly from $\{q, \ldots, 100\}$, where $q = \{1, 10, 25, 50\}$. The distance matrix $\mathbf{D}$ is the Euclidean distance matrix. Distances d_{jj} for $j \in \mathcal{M}$ are set in the same way as for the set 1. The assignment costs c_{ij} are set to $\delta_{ik} d_{ik}$, where δ_{ik} is randomly generated using a uniform distribution from $\{q, \ldots, 100\}$ for $q = \{1, 10, 25, 50\}$. The instances are identified by the number of points p and class q.

4.2. Parameter setting

Two rounds of experiments were performed to tune our MS-ITS. We first set the parameters P (number of multi-start solutions) and I (number of tabu search iterations) using a selected number of instances. Three instances for each of the five problems were selected to perform the experiments using the criteria of picking those instances that are neither too easy nor too hard to solve in a short run according to observations during preliminary experiments. These are: *tai40a*, *tai50a*, and *sko56* for the QAP; *tai10a*, *tai12a*, and *tai15a* for the QBAP; $|\mathcal{N}| = \{18, 20, 22\}$ for the BiQAP; *35-15-95*, *40-09-95*, and *50-10-95* for the GQAP; and *100-C1*, *125-C10*, and *125-C25* for the QSAP. Then, we

analyze our metaheuristic speedup when using the parallelism by presenting a scalability analysis to determine the number of threads to use in the search.

4.2.1. Setting the number of multi-start solutions and tabu search iterations

We performed a grid search to determine which combination of the parameters P and I leads to a faster convergence of the best solution found by our MS-ITS. We increased P from 40 to 1,000 and I from 0 to 500, resulting in 16 combinations of parameters values. Tests were replicated 20 times for each instance with each of the combinations tested. Only the time limit was considered as a stopping criterion, being 10 seconds for the GQAP, one minute for the QAP, QBAP, and QSAP, and five minutes for the BiQAP. The results are presented in Table 1. The gaps are computed as the average relative distance between the average solution of the 20 replications and the best solution found.

Table 1: Parameter setting of MS-ITS

Problem	*Time*	P	$I = 0$ Gap (%)	$I = 50$ Gap (%)	$I = 200$ Gap (%)	$I = 500$ Gap (%)
QAP	1 min	40	0.47	0.17	**0.12**	0.14
		100	0.45	0.16	0.15	0.13
		400	0.43	0.14	**0.12**	0.15
		1000	0.42	0.14	**0.12**	0.13
QBAP	1 min	40	3.32	4.61	6.21	7.59
		100	3.15	4.96	5.96	7.29
		400	2.62	4.31	5.86	6.84
		1000	**2.43**	4.33	6.04	7.08
BiQAP	5 min	40	0.16	0.09	0.09	0.08
		100	0.15	0.08	0.07	0.08
		400	0.15	0.07	0.07	0.08
		1000	0.15	0.08	0.07	**0.06**
QSAP	1 min	40	0.04	0.01	0.01	0.02
		100	0.06	0.01	0.02	0.01
		400	0.07	**0.00**	**0.00**	**0.00**
		1000	0.04	**0.00**	**0.00**	**0.00**
GQAP	10 sec	40	0.16	**0.00**	0.07	0.16
		100	0.13	0.01	0.14	0.47
		400	0.06	0.03	0.48	0.79
		1000	0.05	0.18	0.70	0.81

The results show that solution quality highly depends on the parameters chosen for the algorithm. The best observed value for I is different for different problems, being 200 for QAP, 0 for QBAP, meaning that the tabu search should not be used at all during the search, between 200 and 500 for BiQAP, and 50 for GQAP. Interestingly, for QSAP any $I > 0$ resulted in similar solutions. The number of

tabu search iterations is related to the time spent on the tabu search. In a time-constrained run, for a fixed P, longer tabu searches mean that the algorithm will spend more time searching around the region of the initial solution, while in shorter tabu searches, more calls of the perturbation phase are made during the run, which allows a broader exploration of different regions of the solution space. Nonetheless, by increasing P the effect is the opposite. More start points mean a higher diversification in the search since our perturbation phase moves solutions to a limited distance, while the random solutions generated in the beginning of the algorithm are potentially more diverse. The results show that, in a short run, convergence is faster for higher values of P in the QBAP and QSAP. The opposite was observed for the GQAP, while for the QAP and BiQAP results are quite similar for any P.

For the remaining experiments, we chose the following parameter values based on these results:

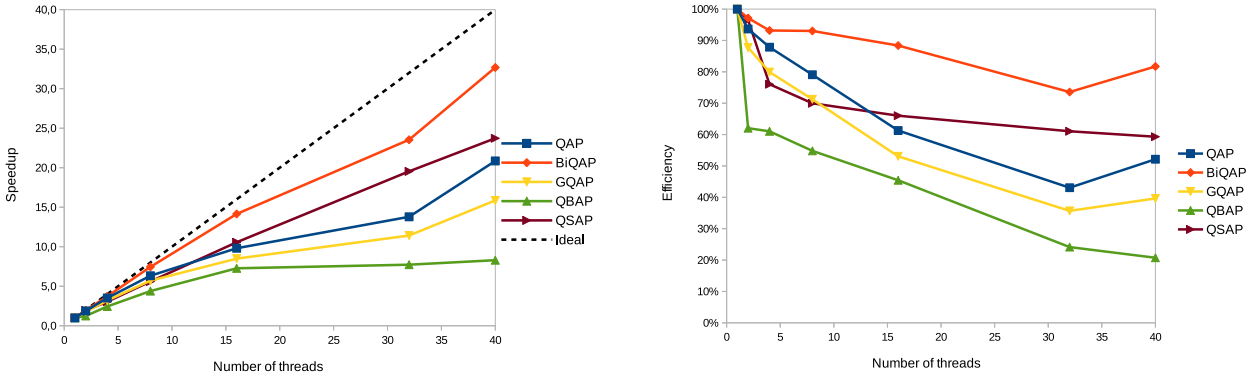
- QAP: $P = 40$ and $I = 200$;
- BiQAP: $P = 1000$ and $I = 500$;
- GQAP: $P = 40$ and $I = 50$;
- QSAP: $P = 400$ and $I = 50$;
- QBAP: $P = 1000$ and $I = 0$.

4.2.2. Setting the number of threads

The parallelism in our MS-ITS is used to speed up the search by allowing multiple solutions to improve in parallel. We run a scalability analysis to verify the performance of the parallel algorithm by computing the speedup efficiency of the parallelism. The speedup obtained when running with T threads is calculated as $S_T = r_1/r_T$, where r_T is the run time for the algorithm with T threads. Ideally, a parallel algorithm would speed up the search the same number of times as the number of parallel threads used. In practice, S_T is upper bounded by T due to the time lost with the parallel overheads and the tasks performed outside of the parallelized code. A measure of efficiency in the scalability analysis is thus calculated as $E_T = S_T/T$, where E_T is the percentage of speedup obtained by the run with T threads compared to the ideal case.

To perform equivalent runs, we used the maximum number of cycles as the sole stopping criterion represented by the number of iterations performed by MS-ITS (lines 10–24 in Algorithm 1). The number of cycles was set so that run time using 40 threads are approximately those in column *Time* in Table 1. We run for $T = \{1, 2, 4, 8, 16, 32, 40\}$ threads, where the maximum value is the number

of cores supported by the CPU used. Figure 1a presents the speedup obtained in each problem, and Figure 1b shows the efficiency of the parallelism.



(a) Speedup **(b)** Efficiency

Figure 1: Scalability analysis for the parallelism in the MS-ITS

The results demonstrate that in all problems, even considering their different parameter settings and instance sizes, speedup increases when the number of threads increases up to 40 threads. This is an indication that the performance could be further improved by using a larger number of threads with a computer with more cores available. Since the tabu search is the most time-consuming task performed in the MS-ITS, note that speedup is higher in the BiQAP where the number of iterations in the TS is the highest among all problems, while it is lower in the QBAP since TS is not applied at all. The speedup is also high in the QSAP probably due to the large instances used in this experiment. It is also interesting to observe the rise in efficiency when increasing the number of threads from 32 to 40 in some problems. A possible explanation lies on the management done by the OpenMP when assigning tasks to threads. Taking the QAP as an example, as we consider 40 parallel solutions, it is more efficient to assign each one to each thread in a run with 40 threads, where the run time will be the longest time to perform all operations in one of the threads, than to assign to only 32 threads, where the eight remaining solutions will have to wait until threads are available to be processed, which can potentially double the time taken to run a cycle in the MS-ITS. Since the observed speedup is the best with 40 threads, we used this as the number of threads for the next experiments.

4.3. Comparison of MS-ITS against state-of-the-art methods

We present next the performance of our MS-ITS for all instance sets described in Section 4.1. In all but the GQAP, as stopping criteria, we used a time limit of one hour or a convergence criteria of 50% of the parallel solutions being equal to the best solution found so far. The maximum number of cycles

is unlimited. MS-ITS is executed 10 times for each instance. All state-of-the-art methods used as comparison are already introduced in Section 2.

4.3.1. Comparison for the QAP

The MS-ITS is compared to the ITS by Misevičius [68], the memetic algorithm (BMA) by Benlic and Hao [8], the parallel hybrid algorithm (PHA) by Tosun [90], and the parallel BLS (BLS-OpenMP) by Aksan et al. [3]. Table 2 shows the results obtained by our heuristic compared to these four state-of-the-art heuristics. Results are reported as the average gap to the best known solution (BKS), calculated as the average of $gap(x) = \frac{f(x) - f(x^*)}{f(x)}$ where $f(x)$ is the solution obtained by the heuristic and $f(x^*)$ is the BKS. Overall, except for the ITS, all heuristics, including our MS-ITS, present very similar performance. MS-ITS slightly outperforms the other heuristics by finding solutions on average at 0.03% from BKS. In 22 of the 31 instances, our method was capable of finding the BKS in all 10 runs. Other heuristics performed better in this metric, like the BLS-OpenMP and the PHA. However, they seem to have difficulties in finding good solutions for larger *Taia* instances. The similar performance between all methods is expected since they all rely on a similar procedure to improve solutions, which is based on the pairwise exchange local search. The differences observed are due to the way that the diversification phase is designed in each of the methods. Since all heuristics were run in different computation environments, the running times cannot be directly compared and are shown only for illustrative purposes.

4.3.2. Comparsion for the QBAP

The QBAP is solved by our MS-ITS and by CPLEX using the MILP model presented in Section 2.2. The results are shown in Table 3. We show the best solution found by our heuristic and the average solution for the 10 runs. CPLEX is run with a time limit of one hour using all 40 threads. Therefore, each solution shown in the table found by the solver is the optimal solution for the problem when the time is under 60 minutes, or the best upper bound found when the run stopped. Note that run times can be larger than 60 minutes due to the time spent to build the model before CPLEX starts, which can be considerably high because when the model is linearized using the standard linearization technique, the number of variables and constraints increases significantly with the problem size. No lower bound is found by CPLEX for the non-solved instances, except for *tai20a*, for which a lower bound of 92 is given. In most of the larger instances, *n.s.* is displayed in the table meaning that no solution is found within the time limit. MS-ITS finds the BKS in 27/31 instances. We note that in

Table 2: Comparison of MS-ITS with state-of-the-art heuristics for the QAP

Instance	*BKS*	*ITS* [68]		*BMA* [8]		*PHA* [90]		*BLS-OpenMP* [3]		*MS-ITS*	
		Gap (%)	*Time (min)*	*Gap (%)*	*Time (min)*	*Gap (%)*	*Time (min)*	*Gap (%)*	*Time (min)*	*Gap (%)*	*Time (min)*
tai20a	703,482	**0.00**	0.0	**0.00**	0.0	**0.00**	0.4	**0.00**	0.1	**0.00**	0.0
tai25a	1,167,256	**0.00**	0.1	**0.00**	0.0	**0.00**	0.5	**0.00**	0.1	**0.00**	0.0
tai30a	1,818,146	**0.00**	0.2	**0.00**	0.0	**0.00**	1.0	**0.00**	0.2	**0.00**	0.1
tai35a	2,422,002	**0.00**	0.5	**0.00**	0.0	**0.00**	1.3	**0.00**	0.3	**0.00**	0.4
tai40a	3,139,370	0.22	1.3	0.06	8.1	**0.00**	10.6	**0.00**	32.2	0.04	30.5
tai50a	4,938,796	0.41	5.5	0.13	42.0	**0.00**	12.7	**0.00**	68.2	0.04	60.0
tai60a	7,205,962	0.45	12.5	0.14	67.5	**0.00**	19.6	**0.00**	107.9	0.21	60.0
tai80a	13,499,184	**0.36**	60.0	0.43	65.8	0.64	40.0	0.50	235.9	0.37	60.0
tai100a	21,052,466	**0.30**	200.0	0.40	44.1	0.54	71.9	0.62	448.5	0.31	60.0
tai20b	122,455,319	**0.00**	0.0	**0.00**	0.0	**0.00**	0.4	**0.00**	0.1	**0.00**	0.0
tai25b	344,355,646	**0.00**	0.0	**0.00**	0.0	**0.00**	0.6	**0.00**	0.0	**0.00**	0.0
tai30b	637,117,113	0.01	0.0	**0.00**	0.0	**0.00**	0.8	**0.00**	0.2	**0.00**	0.2
tai35b	283,315,445	0.02	0.1	**0.00**	0.0	**0.00**	1.1	**0.00**	0.3	**0.00**	0.2
tai40b	637,250,948	0.01	0.2	**0.00**	0.0	**0.00**	1.6	**0.00**	0.3	**0.00**	0.1
tai50b	458,821,517	0.02	0.5	**0.00**	1.2	**0.00**	5.8	**0.00**	0.7	**0.00**	0.9
tai60b	608,215,054	0.04	1.1	**0.00**	5.2	**0.00**	9.5	**0.00**	18.6	**0.00**	7.6
tai80b	818,415,043	0.23	3.0	**0.00**	31.3	**0.00**	27.7	**0.00**	218.1	**0.00**	60.0
tai100b	1,185,996,137	0.14	6.7	**0.00**	13.6	**0.00**	42.5	**0.00**	160.8	**0.00**	60.0
sko42	15,812	**0.00**	0.2	**0.00**	0.0	**0.00**	1.6	**0.00**	0.4	**0.00**	0.1
sko49	23,386	0.01	0.3	**0.00**	0.0	**0.00**	4.0	**0.00**	0.6	**0.00**	9.2
sko56	34,458	0.02	0.7	**0.00**	0.0	**0.00**	16.2	**0.00**	0.8	**0.00**	9.7
sko64	48,498	0.01	1.5	**0.00**	0.0	**0.00**	23.1	**0.00**	1.2	**0.00**	3.1
sko72	66,256	0.06	3.0	**0.00**	3.5	**0.00**	33.6	**0.00**	1.8	**0.00**	60.0
sko81	90,998	0.06	6.0	**0.00**	4.3	**0.00**	39.9	**0.00**	2.4	0.01	60.0
sko90	115,534	0.07	12.0	**0.00**	15.3	**0.00**	40.5	**0.00**	3.2	**0.00**	60.0
sko100a	152,002	0.09	30.0	**0.00**	22.3	**0.00**	41.7	**0.00**	29.8	0.01	60.0
sko100b	153,890	0.06	30.0	**0.00**	6.5	**0.00**	42.3	**0.00**	8.5	**0.00**	60.0
sko100c	147,862	0.06	30.0	**0.00**	12.0	**0.00**	42.2	**0.00**	4.3	**0.00**	60.0
sko100d	149,576	0.09	30.0	0.01	20.9	**0.00**	41.9	**0.00**	12.9	0.02	60.0
sko100e	149,150	0.07	30.0	**0.00**	11.9	**0.00**	42.5	**0.00**	4.3	**0.00**	60.0
sko100f	149,036	0.08	30.0	**0.00**	23.0	**0.00**	42.0	**0.00**	17.1	0.01	60.0
Average		0.09	16.0	0.04	12.9	0.04	21.3	0.04	44.5	**0.03**	31.0

very small instances (*tai10a*, *tai12a*, *tai12b*, *tai15b*, and *tai20b*), MS-ITS finds the optimal solution in all 10 runs. In mid-size ones (*tai15a* to *tai25a*), the heuristic cannot find the optimal solution, being outperformed by the exact method, which is reversed when instances are larger.

4.3.3. Comparsion for the BiQAP

We compare our MS-ITS for the Biquadratic QAP to the GRASP by Mavridou et al. [61], which was kindly provided by the authors so that both methods could be run in the same computation environment. In our experiments, both algorithms were run 10 times. GRASP parameters were set as in the original paper. We limited it to a maximum of 100 iterations per run. The results using both methods are presented in Table 4. All BKS are found only by the MS-ITS. The gaps reported are from the average solution of the 10 runs to the BKS. For the MS-ITS, we show two different gaps. The first is from the solution found after the algorithm stopped using its regular stopping criteria (convergence and time), while the second is the gap at the moment that MS-ITS spent the same time as the GRASP for a fair comparison between both methods. The results show that MS-ITS outperforms GRASP by finding solutions at 0.06% from BKS compared to 0.39% for the GRASP. We also notice that by

Table 3: Comparison of MS-ITS with the MILP model for the QBAP

Instance	*MILP*		*MS-ITS*		
	Solution	*Time (min)*	*Best sol.*	*Avg. sol.*	*Time (min)*
tai10a	**4,256**	0.1	**4,256**	4,256	12.5
tai12a	**4,756**	0.2	**4,756**	4,756	60.0
tai15a	**4,757**	0.9	4,980	5,081.2	60.0
tai17a	**4,704**	18.5	5,084	5,279.1	60.0
tai20a	**5,236**	64.3	5,828	5,940.7	60.0
tai25a	**5,940**	60.3	6,370	6,511.9	60.0
tai30a	7,134	60.7	**7,055**	7,127.3	60.0
tai35a	8,613	61.3	**7,332**	7,508.7	60.0
tai40a	9,360	63.5	**7,663**	7,737.7	60.0
tai50a	9,604	80.9	**8,118**	8,169.8	60.0
tai60a	*n.s.*		**8,342**	8,417.9	60.0
tai80a	*n.s.*		**8,811**	8,823.9	60.0
tai100a	*n.s.*		**8,930**	9,000.1	60.0
tai12b	**4,371,380**	0.4	**4,371,380**	4,371,380	1.4
tai15b	**22,204,329**	5.3	**22,204,329**	22,204,329	0.0
tai20b	**17,132,916**	61.7	**17,132,916**	17,132,916	1.7
tai25b	36,151,400	60.3	**17,878,110**	18,176,040.1	60.0
tai30b	62,752,438	60.4	**26,597,690**	26,989,418.2	60.0
tai35b	16,153,962	61.2	**8,328,912**	8,851,863.2	60.0
tai40b	24,261,283	64.0	**14,617,192**	16,283,570.1	60.0
tai50b	*n.s.*		**7,758,079**	8,002,736.1	60.0
tai60b	*n.s.*		**7,759,900**	8,021,782.9	60.0
tai80b	*n.s.*		**5,885,460**	6,147,192.4	60.0
tai100b	*n.s.*		**7,336,101**	7,559,628.3	60.0
sko42	90	65.5	**60**	60	1.2
sko49	*n.s.*		**70**	70	1.1
sko56	*n.s.*		**80**	84	11.5
sko64	*n.s.*		**90**	93	3.2
sko72	*n.s.*		**100**	100	4.5
sko81	*n.s.*		**110**	112	7.5
sko90	*n.s.*		**120**	120	11.7
sko100a	*n.s.*		**120**	129	14.8
sko100b	*n.s.*		**130**	131	33.1
sko100c	*n.s.*		**120**	129	10.3
sko100d	*n.s.*		**120**	130	14.8
sko100e	*n.s.*		**130**	134	16.2
sko100f	*n.s.*		**120**	129	12.3

letting the algorithm run longer the average gap is reduced to 0.02%, which indicates that solutions keep improving by our heuristic. In fact, only instances with up to 14 facilities are stopped by the convergence criteria – with one exception run for $|\mathcal{N}| = 16$ – and instances with up to 18 facilities have the BKS found in all 10 runs. This shows how difficult it is to solve the BiQAP for the conditions considered in the set of instances tested.

4.3.4. Comparison for the QSAP

We conducted tests on the two sets of instances for the QSAP solving them using our MS-ITS heuristic and CPLEX with the level-1 RLT model as presented in Section 2.4. Time limit was set to one hour for both methods. The results are presented in Tables 5 and 6. For set 1, which contains smaller instances, the exact method proved the optimal solution for 19 out of 21 instances. The MS-ITS not

Table 4: Comparison of MS-ITS with GRASP by Mavridou et al. [61] for the BiQAP

Instance	*BKS*	*GRASP* [61]		*MS-ITS*			
		Gap (%)	*Time (min)*	*Gap (%)*	*Time (min)*[1]	*Gap (%)*	*Time (min)*[2]
$\|\mathcal{N}\| = 10$	239,980	0.15	0.0	0.00	0.0	0.00	0.0
$\|\mathcal{N}\| = 12$	478,278	0.63	0.0	0.00	0.0	0.00	0.3
$\|\mathcal{N}\| = 14$	915,566	0.46	0.1	0.00	0.1	0.00	2.4
$\|\mathcal{N}\| = 16$	1,572,146	0.58	0.1	0.05	0.1	0.00	59.5
$\|\mathcal{N}\| = 18$	2,523,592	0.53	0.3	0.06	0.3	0.00	60.0
$\|\mathcal{N}\| = 20$	3,846,588	0.61	0.5	0.17	0.6	0.04	60.0
$\|\mathcal{N}\| = 22$	5,684,042	0.51	1.0	0.14	1.0	0.01	60.0
$\|\mathcal{N}\| = 24$	8,084,696	0.42	1.6	0.16	1.8	0.06	60.0
$\|\mathcal{N}\| = 26$	11,150,786	0.34	2.7	0.07	2.9	0.03	60.1
$\|\mathcal{N}\| = 28$	15,093,624	0.32	4.2	0.07	4.7	0.03	60.1
$\|\mathcal{N}\| = 30$	19,837,908	0.28	6.3	0.06	7.0	0.03	60.1
$\|\mathcal{N}\| = 32$	25,773,108	0.25	9.6	0.05	10.1	0.03	60.2
$\|\mathcal{N}\| = 34$	32,893,030	0.18	14.2	0.02	15.8	0.01	60.3
$\|\mathcal{N}\| = 36$	41,318,986	0.18	19.6	0.04	21.3	0.03	60.4
Average		0.39	4.3	0.06	4.7	0.02	47.4

[1] Stopping criteria is the running time from GRASP

[2] Regular stopping criteria (50% of solutions converged or one-hour time limit)

only finds all BKS, including the two instances without proven optimality, but it also presents a very high consistency by finding the BKS in all runs for all instances. All that in an average time of less than one minute, compared to nine minutes for the level-1 RLT model. The set 2 is harder to solve due to the larger number of facilities and sites. None of the 20 instances has the optimality proven by the exact method, although in three occasions the BKS are equally found in both methods. For instances with up to $|\mathcal{N}| = 100$, the MS-ITS always stops when reaching the convergence stopping criteria. In larger instances, although the heuristic has not converged, in most of them the same solution is found in all 10 runs. The exceptions are the two largest instances of this set, *150-C25* and *150-C50*. Nevertheless, MS-ITS outperforms the exact approach by finding better solutions in average for all instances.

4.3.5. Comparison for the GQAP

We compare next the proposed MS-ITS with other heuristics available in the literature for the QSAP. They are the memetic heuristic (Memetic) by Cordeau et al. [22], the GRASP (GRASP-PR) by Mateus et al. [60], and the tabu search (TS) by McKendall and Li [62]. Following the procedure used in the two latter papers, we added a new stopping criterion so that the MS-ITS stops the search right after finding the BKS reported in the other works. Table 7 shows the results obtained for each instance tested. The times for the other heuristics are those reported in their works. We highlight that the TS by McKendall and Li [62] was stopped in the instance *30-20-95* in a different solution, 0.27% far from the BKS. For this same instance, our MS-ITS found the BKS in 6 out of 10 runs within an one-hour

Table 5: Comparison of MS-ITS and level-1 RLT for the set 1 of QSAP instances

Instance	*Level-1 RLT*		*MS-ITS*		
	Sol.	*Time (min)*	*Best sol.*	*Avg. sol.*	*Time (min)*
20-15-35	**1,599,473**	0.7	**1,599,473**	1,599,473	0.0
20-15-55	**1,427,052**	0.3	**1,427,052**	1,427,052	0.0
20-15-75	**1,648,679**	0.5	**1,648,679**	1,648,679	0.1
30-06-95	**5,486,902**	0.2	**5,486,902**	5,486,902	0.0
30-07-75	**4,834,397**	0.1	**4,834,397**	4,834,397	1.3
30-08-55	**4,484,813**	0.3	**4,484,813**	4,484,813	0.0
30-10-65	**3,649,165**	0.4	**3,649,165**	3,649,165	0.0
30-20-35	**3,351,755**	17.8	**3,351,755**	3,351,755	0.1
30-20-55	**3,247,260**	12.0	**3,247,260**	3,247,260	0.1
30-20-75	**3,301,384**	60.3	**3,301,384**	3,301,384	0.1
30-20-95	**2,941,907**	5.5	**2,941,907**	2,941,907	0.0
35-15-35	**4,533,539**	5.2	**4,533,539**	4,533,539	0.4
35-15-55	**4,220,924**	3.7	**4,220,924**	4,220,924	0.1
35-15-75	**5,620,789**	60.3	**5,620,789**	5,620,789	0.1
35-15-95	**4,555,240**	8.0	**4,555,240**	4,555,240	0.1
40-07-75	**8,347,601**	0.8	**8,347,601**	8,347,601	0.1
40-09-95	**7,107,977**	1.9	**7,107,977**	7,107,977	0.1
40-10-65	**7,509,269**	2.5	**7,509,269**	7,509,269	0.1
50-10-65	**11,795,583**	5.0	**11,795,583**	11,795,583	12.8
50-10-75	**10,107,391**	2.3	**10,107,391**	10,107,391	0.4
50-10-95	**11,882,812**	8.1	**11,882,812**	11,882,812	1.6
Average	5,316,852.9	9.3	5,316,852.9	5,316,852.9	0.8

Table 6: Comparison of MS-ITS and level-1 RLT for the set 2 of QSAP instances

Instance	*Level-1 RLT*		*MS-ITS*		
	Sol.	*Time (min)*	*Best sol.*	*Avg. sol.*	*Time (min)*
50-C1	**1,022,084**	60.9	**1,022,084**	1,022,084	2.4
50-C10	1,124,895	60.7	**1,123,607**	1,123,607	2.1
50-C25	**1,292,223**	60.8	**1,292,223**	1,292,223	1.6
50-C50	**1,573,474**	60.5	**1,573,474**	1,573,474	2.6
75-C1	2,030,066	61.9	**1,993,829**	1,993,829	0.3
75-C10	2,254,452	61.2	**2,195,019**	2,195,019	0.4
75-C25	2,575,083	61.3	**2,530,699**	2,530,699	0.5
75-C50	3,118,480	61.4	**3,089,736**	3,089,736	0.7
100-C1	6,539,252	60.1	**3,490,341**	3,490,341	12.3
100-C10	7,122,728	60.1	**3,836,299**	3,836,299	11.9
100-C25	8,094,043	60.1	**4,412,117**	4,412,117	6.7
100-C50	9,712,705	60.1	**5,370,205**	5,370,205	3.5
125-C1	10,279,741	60.9	**4,881,972**	4,881,972	60.1
125-C10	11,194,743	60.9	**5,375,051**	5,375,051	60.1
125-C25	12,719,555	60.8	**6,196,362**	6,196,362	60.1
125-C50	15,260,785	60.9	**7,564,714**	7,564,714	60.1
150-C1	14,631,247	63.1	**6,930,942**	6,930,942	60.2
150-C10	15,934,010	63.3	**7,625,357**	7,625,357	60.2
150-C25	18,104,789	63.3	**8,782,100**	8,782,105.9	60.1
150-C50	21,721,652	63.2	**10,707,271**	10,707,313.5	60.2
Average	8,315,300.3	61.3	4,499,670.1	4,499,672.5	26.3

limit. However, the time reported is for the 10 runs. The average solution gap for this instance is 0.11% from BKS. The difficulty to solve this instance comes from the tight capacity available in the sites, which makes the search for feasible solutions longer using our constructive heuristic. Although a direct comparison of running time between the different heuristics is not possible since they were run in different environments, we can see how effective the proposed MS-ITS is. Excluding the hard

30-20-95 instance, the average time for our heuristic to find the BKS is less than one second.

Table 7: Comparison of MS-ITS with state-of-the-art heuristics for the GQAP

Instance	*BKS*	*Memetic* [22] *Time (s)*	*GRASP-PR* [60] *Time (s)*	*TS* [62] *Time (s)*	*MS-ITS* *Time (s)*
20-15-35	1,471,896	96.0	7.1	0.1	0.1
20-15-55	1,723,638	102.0	2.9	0.1	0.0
20-15-75	1,953,188	102.0	2.0	251.0	0.1
30-06-95	5,160,920	114.0	2.6	0.1	0.3
30-07-75	4,383,923	156.0	7.8	0.1	0.1
30-08-55	3,501,695	96.0	1.6	0.1	0.1
30-10-65	3,620,959	210.0	121.9	4.1	0.5
30-20-35	3,379,359	564.0	79.0	1970.2	0.3
30-20-55	3,593,105	462.0	25.2	1474.0	0.2
30-20-75	4,050,938	522.0	41.4	0.7	0.2
30-20-95	5,710,645	5232.0	543019.0	10.0[1]	972.4[2]
35-15-35	4,456,670	456.0	306.1	1605.7	0.6
35-15-55	4,639,128	384.0	21.1	0.2	0.4
35-15-75	6,301,723	396.0	68.2	0.5	0.3
35-15-95	6,670,264	864.0	1454.0	1.0	2.8
40-07-75	7,405,793	180.0	59.4	0.6	0.3
40-09-95	7,667,719	1140.0	417.0	4.6	3.5
40-10-65	7,265,559	240.0	17.9	0.2	0.4
50-10-65	10,513,029	504.0	24.6	1.4	0.5
50-10-75	11,217,503	606.0	1352.4	3.3	1.1
50-10-95	12,845,598	1254.0	89.4	26.0	1.5
Average[3]		422.4	205.1	267.2	0.7

[1] Avg. time to find the solution 5,726,530 (gap of 0.27% from BKS)
[2] BKS is found in 6 out of 10 runs. The average gap is 0.11%
[3] Excluding the instance 30-20-95

5. Conclusions

In this paper, we presented a simple yet effective multi-start iterated tabu search (MS-ITS) algorithm, adapted to solve the Quadratic Assignment Problem (QAP) and four of its variants, i.e., the Quadratic Bottleneck Assignment Problem (QBAP), the Biquadratic Assignment Problem (BiQAP), the Quadratic Semi-Assignment Problem (QSAP), and the Generalized Quadratic Assignment Problem (GQAP). Although these variants have many practical applications, some of their literature is outdated compared to the advances recently achieved for the QAP.

The MS-ITS was developed following the successful diversification-intensification approach for QAPs, where solutions are modified using a controlled number of random swaps, then being improved by one or two local searches, depending on the problem, and by a tabu search whenever new local optima are found. The algorithm was set for each individual problem through computational experiments using sets of benchmark instances found in literature, among other instances created here. The results of the experiments show that the MS-ITS is competitive against other heuristics for the QAP and significantly outperforms the state-of-the-art methods for the other four problems.

References

[1] M. Abdel-Basset, G. Manogaran, H. Rashad, and A. N. H. Zaied. A comprehensive review of quadratic assignment problem: variants, hybrids and applications. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–24, 2018.

[2] W. P. Adams and T. A. Johnson. Improved linear programming-based lower bounds for the quadratic assignment problem. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 43–75. American Mathematical Society, Rhode Island, 1994.

[3] Y. Aksan, T. Dokeroglu, and A. Cosar. A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering*, 103:105–115, 2017.

[4] J. Balakrishnan, C. H. Cheng, D. G. Conway, and C. M. Lau. A hybrid genetic algorithm for the dynamic plant layout problem. *International Journal of Production Economics*, 86(2):107–120, 2003.

[5] M. S. Bazaraa and H. D. Sherali. On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. *Journal of the Operational Research Society*, 33(11):991–1003, 1982.

[6] S. Benjaafar. Modeling and analysis of congestion in the design of facility layouts. *Management Science*, 48(5):679–704, 2002.

[7] U. Benlic and J.-K. Hao. Breakout local search for the quadratic assignment problem. *Applied Mathematics and Computation*, 219(9):4800–4815, 2013.

[8] U. Benlic and J.-K. Hao. Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, 42(1):584–595, 2015.

[9] A. Billionnet and S. Elloumi. Best reduction of the quadratic semi-assignment problem. *Discrete Applied Mathematics*, 109(3):197–213, 2001.

[10] A. Billionnet, M.-C. Costa, and A. Sutter. An efficient algorithm for a task allocation problem. *Journal of the Association for Computing Machinery*, 39(3):502–518, 1992.

[11] J. Bos. Zoning in forest management: a quadratic assignment problem solved by simulated annealing. *Journal of Environmental Management*, 37(2):127–145, 1993.

[12] C. Bousono-Calzon and M. R. W. Manning. The hopfield neural network applied to the quadratic assignment problem. *Neural Computing & Applications*, 3(2):64–72, 1995.

[13] R. E. Burkard. Selected topics on assignment problems. *Discrete Applied Mathematics*, 123(1-3): 257–302, 2002.

[14] R. E. Burkard. Quadratic assignment problems. In P. M. Pardalos, D.-Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 2741–2814. Springer, London, 2013.

[15] R. E. Burkard and E. Çela. Heuristics for biquadratic assignment problems and their computational comparison. *European Journal of Operational Research*, 83(2):283–300, 1995.

[16] R. E. Burkard, E. Çela, and B. Klinz. On the biquadratic assignment problem. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 117–146. American Mathematical Society, 1994.

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

[18] E. Cela, V. Deineko, and G. J. Woeginger. New special cases of the quadratic assignment problem with diagonally structured coefficient matrices. *European Journal of Operational Research*, 267 (3):818–834, 2018.

[19] W.-C. Chiang and C. Chiang. Intelligent local search strategies for solving facility layout problems with the quadratic assignment problem formulation. *European Journal of Operational Research*, 106(2-3):457–488, 1998.

[20] W.-C. Chiang, P. Kouvelis, and T. L. Urban. Incorporating workflow interference in facility layout design: The quartic assignment problem. *Management Science*, 48(4):584–590, 2002.

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

[22] J.-F. Cordeau, M. Gaudioso, G. Laporte, and L. Moccia. A memetic heuristic for the generalized quadratic assignment problem. *INFORMS Journal on Computing*, 18(4):433–443, 2006.

[23] V.-D. Cung, T. Mautor, P. Michelon, and A. Tavares. A scatter search based approach for the quadratic assignment problem. In *IEEE International Conference on Evolutionary Computation and Evolutionary Programming*, pages 165–170, 1997.

[24] M. Czapiński. An effective parallel multistart tabu search for quadratic assignment problem on CUDA platform. *Journal of Parallel and Distributed Computing*, 73(11):1461–1468, 2013.

[25] M. Dell'Amico, J. C. D. Diaz, M. Iori, and R. Montanari. The single-finger keyboard layout problem. *Computers & Operations Research*, 36(11):3002–3012, 2009.

[26] J. W. Dickey and J. W. Hopkins. Campus building arrangement using TOPAZ. *Transportation Research*, 6(1):59–68, 1972.

[27] Z. Drezner. Lower bounds based on linear programming for the quadratic assignment problem. *Computational Optimization and Applications*, 4(2):159–165, 1995.

[28] Z. Drezner. A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, 15(3):320–330, 2003.

[29] Z. Drezner. Compounded genetic algorithms for the quadratic assignment problem. *Operations Research Letters*, 33(5):475–480, 2005.

[30] Z. Drezner. The extended concentric tabu for the quadratic assignment problem. *European Journal of Operational Research*, 160(2):416–422, 2005.

[31] H. A. Eiselt and G. Laporte. A combinatorial optimization problem arising in dartboard design. *Journal of the Operational Research Society*, 42(2):113–118, 1991.

[32] A. N. Elshafei. Hospital layout as a quadratic assignment problem. *Journal of the Operational Research Society*, 28(1):167–179, 1977.

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

[34] G. Finke, R. E. Burkard, and F. Rendl. Quadratic assignment problems. *Annals of Discrete Mathematics*, 31(1):61–82, 1987.

[35] C. Fleurent and J. A. Ferland. Genetic hybrids for the quadratic assignment problem. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems, DIMACS*

Series in Discrete Mathematics and Theoretical Computer Science, volume 16, pages 173–187. American Mathematical Society, 1994.

[36] M. C. Fu and B. K. Kaku. Minimizing work-in-process and material handling in the facilities layout problem. *IIE Transactions*, 29(1):29–36, 1997.

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

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

[39] F. Glover and E. Woolsey. Converting the 0–1 polynomial programming problem to a 0–1 linear program. *Operations Research*, 22(1):180–182, 1974.

[40] H. Greenberg. A quadratic assignment problem without column constraints. *Naval Research Logistics Quarterly*, 16(3):417–421, 1969.

[41] 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.

[42] P. M. Hahn, B.-J. Kim, M. Guignard, J. M. Smith, and Y.-R. Zhu. An algorithm for the generalized quadratic assignment problem. *Computational Optimization and Applications*, 40(3): 351, 2008.

[43] P. Hansen and K.-W. Lih. Improved algorithms for partitioning problems in parallel, pipelined, and distributed computing. *IEEE Transactions on Computers*, 41(6):769–771, 1992.

[44] M. Harris, R. Berretta, M. Inostroza-Ponta, and P. Moscato. A memetic algorithm for the quadratic assignment problem with parallel local search. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 838–845, 2015.

[45] M. Hasegawa, T. Ikeguchi, K. Aihara, and K. Itoh. A novel chaotic search for quadratic assignment problems. *European Journal of Operational Research*, 139(3):543–556, 2002.

[46] T. James, C. Rego, and F. Glover. A cooperative parallel tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research*, 195(3):810–826, 2009.

[47] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the Association for Computing Machinery*, 49(5):616–639, 2002.

[48] J. Knowles and D. Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In C. M. Fonseca, Zitzler E. Fleming, P. J., K. Deb, and L. Thiele, editors, *Proceedings of the Evolutionary Multi-Criterion Optimization (EMO 2003), Second International Conference*, pages 295–310, 2003.

[49] T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica: Journal of the Econometric Society*, pages 53–76, 1957.

[50] J. Krarup and P. M. Pruzan. Computer-aided layout design. In M. L. Balinski and C. Lemarechal, editors, *Mathematical Programming in Use*, pages 75–94. Springer, Berlin, 1978.

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

[52] E. L. Lawler. The quadratic assignment problem: a brief review. In B. Roy, editor, *Combinatorial Programming: Methods and Applications, NATO Advanced Study Institutes Series (Series C – Mathematical and Physical Sciences)*, volume 19, pages 351–360, 1975.

[53] C.-G. Lee and Z. Ma. The generalized quadratic assignment problem. Technical Report MIE-OR TR2005-01, Departament of Mechanical and Industrial Engineering, University of Toronto, Toronto, 2004.

[54] H. Liu, A. Abraham, and J. Zhang. A particle swarm approach to quadratic assignment problems. In A. Saad, K. Dahal, M. Sarfraz, and R. Roy, editors, *Soft Computing in Industrial Applications, Advances in Soft Computing*, volume 39, pages 213–222, Berlin, 2007. Springer.

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

[56] V. F. Magirou and J. Z. Milis. An algorithm for the multiprocessor assignment problem. *Operations Research Letters*, 8(6):351–356, 1989.

[57] F. Malucelli. A polynomially solvable class of quadratic semi-assignment problems. *European Journal of Operational Research*, 91(3):619–622, 1996.

[58] F. Malucelli and D. Pretolani. Lower bounds for the quadratic semi-assignment problem. *European Journal of Operational Research*, 83(2):365–375, 1995.

[59] V. Maniezzo and A. Colorni. The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge & Data Engineering*, 11(15):769–778, 1999.

[60] G. R. Mateus, M. G. C. Resende, and R. M. A. Silva. GRASP with path-relinking for the generalized quadratic assignment problem. *Journal of Heuristics*, 17(5):527–565, 2011.

[61] T. Mavridou, P. M. Pardalos, L. S. Pitsoulis, and M. G. C. Resende. A GRASP for the biquadratic assignment problem. *European Journal of Operational Research*, 105(3):613–621, 1998.

[62] A. McKendall and C. Li. A tabu search heuristic for a generalized quadratic assignment problem. *Journal of Industrial and Production Engineering*, 34(3):221–231, 2017.

[63] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation*, 4(4):337–352, 2000.

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

[65] G. Miranda, H. P. L. Luna, G. R. Mateus, and R. P. M. Ferreira. A performance guarantee heuristic for electronic components placement problems including thermal effects. *Computers & Operations Research*, 32(11):2937–2957, 2005.

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

[67] A. Misevičius. An improved hybrid optimization algorithm for the quadratic assignment problem. *Mathematical Modelling and Analysis*, 9(2):149–168, 2004.

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

[69] D. Munera, D. Diaz, and S. Abreu. Hybridization as cooperative parallelism for the quadratic assignment problem. In *International Workshop on Hybrid Metaheuristics*, pages 47–61. Springer, 2016.

[70] A. Nyberg and T. Westerlund. A new exact discrete linear reformulation of the quadratic assignment problem. *European Journal of Operational Research*, 220(2):314–319, 2012.

[71] C. A. S. Oliveira, P. M. Pardalos, and M. G. C. Resende. GRASP with path-relinking for the quadratic assignment problem. In *International Workshop on Experimental and Efficient Algorithms*, pages 356–368. Springer, 2004.

[72] T. Ostrowski and V. T. Ruoppila. Genetic annealing search for index assignment in vector quantization. *Pattern Recognition Letters*, 18(4):311–318, 1997.

[73] P. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. In P. M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 1–42. American Mathematical Society, 1994.

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

[75] A. T. Phillips and J. B. Rosen. A quadratic assignment formulation of the molecular conformation problem. *Journal of Global Optimization*, 4(2):229–241, 1994.

[76] W. P. Pierskalla. Letter to the editor–the multidimensional assignment problem. *Operations Research*, 16(2):422–431, 1968.

[77] M. A. Pollatschek, H. Gershoni, and Y. T. Radday. Optimization of typewriter keyboard by computer-simulation. *Angewandte Informatik*, (10):438–439, 1976.

[78] A. P. Punnen and Y. Wang. The bipartite quadratic assignment problem and extensions. *European Journal of Operational Research*, 250(3):715–725, 2016.

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

[80] H. Saito, T. Fujie, T. Matsui, and S. Matuura. A study of the quadratic semi-assignment polytope. *Discrete Optimization*, 6(1):37–50, 2009.

[81] I. Schüle, H. Ewe, and K.-H. Küfer. Finding tight RLT formulations for quadratic semi-assignment problems. In S. Cafieri, A. Mucherino, G. Nannicini, F. Tarissan, and L. Liberti, editors, *Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, pages 109–112. University of Twente, 2009.

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

[83] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, 2(1):33–45, 1990.

[84] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the Association for Computing Machinery*, 48(2):206–242, 2001.

[85] J. M. Smith and W.-J. Li. Quadratic assignment problems and M/G/C/C/state dependent network flows. *Journal of Combinatorial Optimization*, 5(4):421–443, 2001.

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

[87] É. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17 (4-5):443–455, 1991.

[88] E.-G. Talbi and V. Bachelet. Cosearch: A parallel cooperative metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 5(1):5–22, 2006.

[89] E.-G. Talbi, O. Roux, C. Fonlupt, and D. Robillard. Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, 17(4):441–449, 2001.

[90] U. Tosun. On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, 39:267–278, 2015.

[91] Y. Z. Ünal and Ö. Uysal. A new mixed integer programming model for curriculum balancing: Application to a turkish university. *European Journal of Operational Research*, 238(1):339–347, 2014.

[92] H. Youssef, S. M. Sait, and H. Ali. Fuzzy simulated evolution algorithm for VLSI cell placement. *Computers & Industrial Engineering*, 44(2):227–247, 2003.