Applied Mathematics and Computation 219 (2013) 4800–4815

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Breakout local search for the quadratic assignment problem

Una Benlic, Jin-Kao Hao *

LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France

ARTICLE INFO

Keywords:
Quadratic assignment
Breakout local search
Adaptive diversification
Landscape analysis

ABSTRACT

The quadratic assignment problem (QAP) is one of the most studied combinatorial optimization problems with various practical applications. In this paper, we present breakout local search (BLS) for solving QAP. BLS explores the search space by a joint use of local search and adaptive perturbation strategies. Experimental evaluations on the set of QAPLIB benchmark instances show that the proposed approach is able to attain current best-known results for all but two instances with an average computing time of less than 4.5 hours. Comparisons are also provided to show the competitiveness of the proposed approach with respect to the best-performing QAP algorithms from the literature.

1. Introduction

The quadratic assignment problem (QAP) is a classical combinatorial optimization problem that is well known for its various applications. QAP can be considered as the problem of assigning at minimal cost a set of n locations to a given set of n facilities, given a flow f_{ij} from facility i to facility j for all $i,j \in \{1,\ldots,n\}$ and a distance d_{ab} between locations a and b for all $a,b \in \{1,\ldots,n\}$. Let Π denote the set of the permutation functions $\pi : \{1,\ldots,n\} \rightarrow \{1,\ldots,n\}$, QAP can mathematically be formulated as follows:

$$min_{\pi\in\Pi} C(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\pi_i\pi_j}, \tag{1}$$

问题的描述

where f and d are the flow and distance matrices respectively, and $\pi \in \Pi$ is a solution where π_i represents the location assigned to facility i. The problem objective is then to find a permutation π^* in Π that minimizes the sum of the products of the flow and distance matrices, i.e., $C(\pi^*) \leqslant C(\pi), \forall \pi \in \Pi$.

QAP is notable for its ability to formulate a great number of practical problems such as the backboard wiring in electronics, the design of typewriter keyboards, campus planning, analysis of chemical reactions for organic compounds, balancing turbine runners, and many others. Reviews on some significant applications of QAP can be found in [8,13,26,32].

问题的应用领域的介绍

* Corresponding author.
E-mail addresses: benlic@info.univ-angers.fr (U. Benlic), hao@info.univ-angers.fr (J.-K. Hao).

0096-3003/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.10.106

该问题的复杂描述和近年的成果，退火遗传等算法在该处的应用

QAP is known to be one of the hardest combinatorial optimization problems. Indeed, the computational complexity of QAP is such that even instances of size $20 \leqslant n \leqslant 30$ represent a real challenge for the current exact approaches [14,18]. Approximate methods are thus often indispensable to handle larger instances. In recent years, many efforts have been made in developing effective heuristic algorithms, which can find high quality solutions in reasonable computation time. Representative examples include simulated annealing [39], tabu search [2,16,19,20,29,33,36], (hybrid) genetic algorithms [1,11,12,15,28,38], ant colony algorithms [35,37], scatter search [10], iterated local search [34], etc. For a comprehensive survey of different solution methods, the reader is referred to [9,22,31].

In this work, we propose a new heuristic approach for QAP, called the breakout local search (BLS), which has been applied successfully to several hard combinatorial problems including maximum clique (both weighted and unweighted cases) [4], maximum cut [5], and minimum sum coloring [6]. Based on the framework of Iterated Local Search (ILS) [25], BLS uses local search (i.e., the steepest descent) to discover local optima and employes adaptive perturbations to continually move from one local optimum to another in the search space. The continual exploration of new search areas is achieved by dynamically determining the number of perturbation moves, and by adaptively choosing between three types of perturbation moves of different intensities, depending on the current search state. Despite its simplicity, BLS shows excellent performance on the set of well-known benchmark instances from the QAPLIB and is very competitive with the current best QAP approaches.

In the next section, we review some of the most popular heuristic approaches for solving QAP. In Section 3, we present our BLS approach and describe in detail its main components. Section 4 shows extensive computational results and comparisons on the QAPLIB benchmark instances. In Section 5, we perform a landscape analysis on different QAP instances and relate this analysis to the performance of BLS. Moreover, we investigate the importance of the adaptive perturbation strategy to the overall algorithm performance. Conclusions are provided in the last section.

根据局部适应性算子，对求解区域进行求解，对当前解进行扰动

2. State-of-art heuristic approaches for the QAP

This section provides a brief summary of some of the best-performing and most representative metaheuristic adaptations for solving QAP. We later use these as reference algorithms for our comparative study, except the popular robust tabu search [37] which is nowadays outperformed by much more recent approaches. It is important to mention that none of the existing QAP approaches can be considered as the most effective for all the different types of QAP instances, due to significant differences in structures of these instance (see Section 5).

The robust tabu search (Ro-TS) proposed by Taillard [37] is probably the best-known heuristic and the simplest to implement. Ro-TS uses the swap move which exchanges two elements of a solution. The tabu list forbids the reverse exchange of a swap move during the next h iterations, where the tabu tenure h varies randomly within a given interval. In addition to the classical aspiration criterion that overrides the tabu status of a move leading to an improved solution, a second criterion is also introduced. A move that has never been selected during a large number of iterations is performed regardless of the tabu status and its contribution to the objective function value. However, the most important new feature introduced in Ro-TS is that a complete swap neighborhood is explored in $O(n^2)$ instead of $O(n^3)$ as in previous algorithms. As many later QAP algorithms, we adopt this technique to speed up the performance of our BLS approach.

关于禁忌搜索的描述与BLS的搜索策略

Cooperative parallel tabu search algorithm (CPTS), proposed by James et al. [20], executes in parallel several tabu search (TS) operators on multiple processors. The TS operator is a modified version of Taillard's Ro-TS [37] obtained by changing the stopping criterion and the tabu tenure parameters for each processor participating in the algorithm. In order to accomplish cooperation between TS processes, CPTS maintains a global reference set which uses information exchange to promote both intensification and diversification in a parallel environment. CPTS globally obtains excellent results on the whole set of QAP instances and is used as a reference algorithm in our comparative study.

In [30], Misevicius presents an enhanced tabu search algorithm (ETS) which employs a modified version of Taillard's Ro-TS as an intensification procedure, and applies mutations to the best solution found so far to diversify the search. ETS can thus be considered as an adaptation of the ILS metaheuristic. To add more robustness to the mutation process, the number of mutation moves is varied in some interval during the execution of the algorithm. Beside a "mild mutation" which serves as a basic mutation, three additional mutation operators (called alternative mutation procedures) are used to add more diversity once a long-lasting stagnation is detected. Depending on the alternative mutation procedure implemented, these ETS variants are called ETS-1, ETS-2 and ETS-3. At the time it was issued, ETS appeared to be superior to other powerful QAP heuristics. Moreover, it improved the best known solutions for several hard QAP instances. We thus use all the three ETS versions for our experimental comparisons.

Misevicius later proposed in [31] another iterated tabu search (ITS) which is quite similar to its ETS approach. It as well employs a traditional tabu search to reach local optima and triggers a perturbation (reconstruction) phase in order to escape the attained local optimum. The "ruined" solution becomes a new starting point for the basic TS procedure. ITS uses a perturbation mechanism which varies adaptively the number of random perturbation moves in some interval. It obtains excellent results on the unstructured instances and the real-life like instances, and outperforms to some extent the ETS algorithm. We thus use it as another reference in our comparison.

In [12], Drezner performs extensive computational experiments on QAP using various variants of a memetic algorithm. The author compares the modified robust tabu search (MRT) and the simple tabu search as local optimization algorithms combined with a crossover operator. Moreover, different parent selection (distance modification, gender modification) and pool updating strategies are tested. The best version of the memetic algorithm is MRT60 which integrates the modified robust tabu search MRT for offspring improvement. MRT is identical to Ro-TS [37] except that the tabu tenure is generated in $[0.2n, 1.8n]$ rather than in $[0.9n, 1.1n]$. MRT60 is the best-performing algorithm for the grid-based instances and is used as another reference algorithm in our comparative study.

The improved hybrid genetic algorithm (IHGA) is proposed by Misevicius [29]. It incorporates a robust local improvement procedure as well as an effective restart mechanism based on shift mutations. The author slightly improved the classic scheme of a uniform like crossover (ULX) to get a new optimized crossover (OX). The optimized crossover is a crossover that (a) is ULX and (b) produces a child that has the best fitness value among the children created by M runs of ULX. The offspring is then improved with a local search mechanism which consists of a tabu search procedure and a solution reconstruction procedure. The reconstruction is achieved by performing a number μ of random swaps, where μ is varied according to the instance size. Once convergence of the algorithm is observed, all the individuals but the best undergo the shift mutation, which simply consists in shifting all the items in a wrap-around fashion by a predefined number of positions. IHGA is one of the best algorithms for the unstructured instances and the real-life like instances and is thus used as one of the references in our comparative study.

A popular approach for QAP is a particular population-based iterated local search (PILS) proposed by Stützle [35]. The underlying iterated local search (ILS) algorithm starts from a random assignment, and applies a first-improvement local search procedure based on the swap neighborhood. To speed up the search process, the algorithm uses the *do not look bit* strategy, previously proposed to accelerate local search algorithms for TSP. Once a local optimum is reached, ILS applies a perturbation that consists of exchanging k randomly chosen items, where k is varied between k_{min} and k_{max}. Stützle extends the described ILS to a population-based algorithm where no interaction between solutions takes place, and each single solution is improved by the standard ILS. In the proposed PILS, the population consists of μ solutions and in each iteration λ new solutions are generated. A selection strategy, based both on distance and quality of solutions, is then employed to form a new population of μ solutions from the set of $\mu + \lambda$ solutions. This algorithm showed excellent performance at the time it was published, though it is now outperformed by other more recent algorithms.

3. The breakout local search

3.1. General procedure

Our breakout local search (BLS) approach is conceptually rather simple. Following the general scheme of iterated local search (ILS) [25], it alternates iteratively between a local search phase (to reach local optima) and a dedicated perturbation phase (to discover new promising regions). More precisely, starting from an initial solution π_0, BLS applies to it the steepest descent procedure (see Section 3.2) to reach a local optimum π. Each iteration of the steepest descent procedure scans the whole neighborhood and selects the best improving neighboring solution. If such a solution does not exist in the neighborhood, local optimality is reached. At this point, BLS tries to escape from the current local optimum π by applying a suitable number L of dedicated perturbation moves to π (we say that π is perturbed). This perturbed solution becomes a new starting point for the next phase of the descent.

The success of any local search approach depends on the right degree of diversification introduced into search. Moreover, the optimal degree of diversification at a certain stage of the search is not necessarily optimal at another stage. The degree of diversification introduced by a perturbation mechanism depends both on the number of perturbation moves (also called "jump magnitude") and the type of moves used for perturbation. If the diversification is too weak, the local search has greater chances to end up cycling between two or more locally optimal solutions leading to search stagnation. On the other hand, a too strong diversification has the same effect as a random restart, which usually results in a low probability of finding better solutions in the following local search phase. The basic idea of the perturbation mechanism employed by BLS is to introduce the most suitable degree of diversification required at a certain stage of the search by dynamically determining the number L of perturbation moves, and by adaptively choosing between three types of perturbation moves of different intensities.

Algorithm 1 presents the BLS algorithm for QAP, whose components are detailed in the following sections. Starting from an initial random solution, BLS uses the descent procedure to reach a local optimum π (lines 10–16). The jump magnitude L is then determined depending on whether the search escaped or returned to the immediate previous local optimum, and whether the search is stagnating in a non-promising region (lines 24–34). BLS then applies L perturbation moves to π in order to get a new starting point for the descent procedure (line 37).

Algorithm 1. Breakout local search for QAP

Require: Distance and flow matrices d and f of size $n \times n$.
Ensure A permutation π over a set of facility locations.
1: $\pi \leftarrow$ *random permutation of* $\{1, \ldots, n\}$
2: $c \leftarrow C(\pi)$ /* *c records the objective value of the current solution* */
3: Compute the initial $n \times n$ matrix δ of move gains /* *Section 3.3* */
4: $\pi_{best} \leftarrow \pi$ /* π_{best} *records the best solution found so far* */
5: $c_{best} \leftarrow c$ /* c_{best} *records the best objective value reached so far* */
6: $c_p \leftarrow c$ /* c_p *records the best objective value of the last descent* */
7: $\omega \leftarrow 0$ /* ω *is the counter for consecutive non-improving local optima* */
8: $L \leftarrow L_0$ /* *set the number of perturb. moves L to its default value* L_0 */
9: **while** stopping condition not reached **do**
10: **while** $\exists$ $swap(u, v)$ such that $(c + \delta(\pi, u, v)) < c$ **do**
11: $\pi \leftarrow \pi \oplus swap(u, v)$ /* *Perform the best-improving move* */
12: $c \leftarrow c + \delta(\pi, u, v)$
13: $H_{uv} \leftarrow Iter$ /* *Update iter. number when move uv was last performed* */
14: Update matrix δ
15: $Iter \leftarrow Iter + 1$
16: **end while**
17: **if** $c < c_{best}$ **then**
18: $\pi_{best} \leftarrow \pi; c_{best} \leftarrow c$ /* *Update the recorded best solution* */
19: $\omega \leftarrow 0$ /* *Reset counter for consecutive non-improv. local optima* */
20: **else if** $c \neq c_p$ **then**
21: $\omega \leftarrow \omega + 1$
22: **end if**
23: /* *Determine the perturbation strength L to be applied to* π */
24: **if** $\omega > T$ **then**
25: /* *Search seems to be stagnating, set L to a large value* */
26: $L \leftarrow L_{Max}$
27: $\omega \leftarrow 0$
28: **else if** $c = c_p$ **then**
29: /* *Search returned to the previous local optimum, increase jump magnitude by one* */
30: $L \leftarrow L + 1$
31: **else**
32: /* *Search escaped from the previous local optimum, reinitialize jump magnitude* */
33: $L \leftarrow L_0$
34: **end if**
35: /* *Perturb the current local optimum* π *with L perturb. moves* */
36: $c_p \leftarrow c$ /* *Uptade the objective value of the previous local optimum* */
37: $\pi \leftarrow Perturbation(\pi, L, H, Iter, \delta, \omega)$ /* *Section 3.4* */
38: **end while**

3.2. The neighborhood relation and its exploitation

As previously explained, a candidate QAP solution can be encoded as a permutation π of $\{1, \ldots, n\}$, such that π_i denotes the location assigned to facility $i \in \{1, \ldots, n\}$. Like many existing local search methods for QAP, we employ the swap move to π which consists in exchanging locations of two facilities.

The neighborhood $N(\pi)$ of a solution π is then defined as the set of all the permutations that can be obtained by exchanging any two values π_u and π_v, i.e., $N(\pi) = \{\pi' | \pi'_u = \pi_v,\ \pi'_v = \pi_u,\ u \neq v \text{ and } \pi'_i = \pi_i,\ \forall i \neq u, v\}$. The size of $N(\pi)$ is thus equal to $n(n-1)/2$.

BLS uses the steepest descent to explore the whole neighborhood $N(\pi)$. In other words, each iteration of the steepest descent identifies the best swap and applies it to π to obtain a new solution. This descent process is repeated until a local optimum is reached (see lines 10–16 of Algorithm 1). Note that the cost of a solution obtained with a swap move can be computed in time $O(n^2)$ based on the objective function given in Eq. 1, leading to a total time of $O(n^4)$ for evaluating all the solutions from $N(\pi)$. Obviously, such an explicit calculation becomes very expensive as the instance size increases. In [37], Taillard suggests a very effective way to incrementally update the objective variation of each move. In the next section, we examine this idea which we adopt in our BLS algorithm.

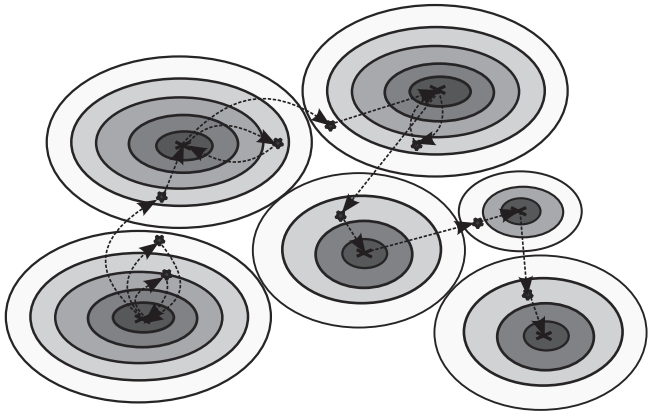

Fig. 1. The search trajectory of the BLS algorithm.

3.3. Fast computation and update of move gains

Let π be a solution, $swap(r,s)$ a move, and $\pi' = \pi \oplus swap(r,s)$ the neighboring solution of π obtained by applying $swap(r,s)$ to π. Then the cost variation between the two solutions $\delta(\pi,r,s) = C(\pi') - C(\pi)$, also called the move gain of $swap(r,s)$, can be efficiently calculated as follows [37].

$$\delta(\pi,r,s) = f_{rr}(d_{\pi_s\pi_s} - d_{\pi_r\pi_r}) + f_{rs}(d_{\pi_s\pi_r} - d_{\pi_r\pi_s}) + f_{sr}(d_{\pi_r\pi_s} - d_{\pi_s\pi_r}) + f_{ss}(d_{\pi_r\pi_r} - d_{\pi_s\pi_s}) + \sum_{i=1,i\neq r,s}^{n} (f_{ir}(d_{\pi_i\pi_s} - d_{\pi_i\pi_r}) + f_{is}(d_{\pi_i\pi_r} - d_{\pi_i\pi_s}) + f_{ri}(d_{\pi_s\pi_i} - d_{\pi_r\pi_i}) + f_{si}(d_{\pi_r\pi_i} - d_{\pi_s\pi_i})). \quad (2)$$

Computing $\delta(\pi,r,s)$ with Eq. 2 thus requires only $O(n)$ time instead of $O(n^2)$ when calculated with Eq. 1. Furthermore, if μ is the solution obtained after swapping locations of facilities r and s in the solution π, it is possible to compute the value $\delta(\mu,u,v)$ even faster if the pairs $\{r,s\}$ and $\{u,v\}$ are mutually exclusive $\{r,s\} \cup \{u,v\} = \emptyset$. Indeed, the formula given in Eq. 3 computes the change in the value of the objective function $\delta(\mu,u,v)$ in time $O(1)$, given that the value $\delta(\pi,u,v)$ from the previous permutation π is known.

$$\delta(\mu,u,v) = (f_{ru} - f_{rv} + f_{sv} - f_{su})(d_{\mu_s\mu_u} - d_{\mu_s\mu_v} + d_{\mu_r\mu_v} - d_{\mu_r\mu_u}) + (f_{ur} - f_{vr} + f_{vs} - f_{us})(d_{\mu_u\mu_s} - d_{\mu_v\mu_s} + d_{\mu_v\mu_r} - d_{\mu_u\mu_r}) + \delta(\pi,u,v) \quad (3)$$

Note that only $2n-3$ pairs are not mutually exclusive and Eq. 2 is then used to evaluate $\delta(\mu,u,v)$. Therefore, calculating the change in the evaluation function C for all the $n(n-1)/2$ possible moves from the swap neighborhood requires $O(n^2)$ instead of $O(n^4)$ when the calculation of each C is done directly with Eq. 1, or $O(n^3)$ when using Eq. 2.

3.4. Adaptive perturbation mechanism

The perturbation mechanism plays a crucial role within BLS since the steepest descent alone cannot escape from a local optimum. As described in the following sections, BLS thus tries to move to the next local optimum by varying the number of perturbation moves (Section 3.4.1) and and the type of perturbation moves (Section 3.4.2) depending on the search state. The pseudo-code of this adaptive perturbation-based diversification procedure is given in Algorithms 2 and 3.

3.4.1. Jump magnitude

The idea of BLS is to first explore neighboring local optima, and move to a new distant area only if the search seems to be stagnating. Therefore, after each local search phase, BLS performs most of the time a small number L of directed or random moves (see Section 3.4.2) that is hopefully just strong enough to escape the attraction of the current local optimum and to fall under the influence of a neighboring local optimum. If the jump was not sufficient to escape the current local optimum, the jump magnitude L is incremented and the perturbation is applied again to the current local optimum (see lines 28–30 of Algorithm 1). Otherwise, the number of perturbation moves is set to its default value, i.e., $L = L_0$ (see lines 31–33 of Algorithm 1). A strong perturbation (with a large number L_{Max} of moves) is carried out only after consecutively visiting a certain number T of local optima without any improvement on the quality of the best solution found (see lines 24–27 of Algorithm 1). This ensures that the search trajectory is not confined in a localized portion of the search space. The described process of moving from one local optimum to another is illustrated in Fig. 1.

Algorithm 2. Adaptive perturbation procedure *perturbation* $(\pi, L, H, Iter, \delta, \omega)$

Require. Local optimum π, perturbation strength L, matrix H, global iteration counter *Iter*, move gain matrix δ, number of consecutive non-improving local optima visited ω
Ensure. A perturbed solution π.
1: Determine probability P according to Formula (4) /* *Section 3.4.2* */
2: With probability P, $\pi \leftarrow Perburb(\pi, L, H, Iter, \delta, \omega, A)$
/* *Directed perturbation* */
3: With probability $(1-P)\cdot Q$, $\pi \leftarrow Perturb(\pi, L, H, Iter, \delta, \omega, B)$
/* *Recency perturbation* */
4: With probability $(1-P)\cdot(1-Q)$, $\pi \leftarrow Perturb(\pi, L, H, Iter, \delta, \omega, C)$
/* *Random perturbation* */
5: *Return* π

Algorithm 3. Perturbation operator $Perburb(\pi, L, H, Iter, \delta, \omega, M)$

Require. Identical to $Perturbation(\pi, L, H, Iter, \delta, \omega)$ of Algorithm 2, together with the set of perturbation moves M
Ensure. A perturbed solution π
1: **for** $i := 1$ to L
2: Take $swap(u, v) \in M$
3: $\pi \leftarrow \pi \oplus swap(u, v)$
4: $c \leftarrow c + \delta(\pi, u, v)$
5: $H_{uv} \leftarrow Iter$
6: Update the move gain matrix δ and M
7: $Iter \leftarrow Iter + 1$
8: **if** $c < c_{best}$ **then**
9: $\pi_{best} \leftarrow \pi; c_{best} \leftarrow c$ /* *Update the recorded best solution* */
10: $\omega \leftarrow 0$ /* *Reset counter for consecutive non-improv. local optima* */
11: **end if**
12: **end for**
13: *Return* π

3.4.2. Three types of perturbation moves

Instead of making random jumps all the time, BLS alternates between three types of perturbations: directed, recency-based, and random.

The *directed perturbation* is based on the tabu search principles [18]. It uses a selection rule that favors swap moves that minimize the cost degradation, under the constraint that the move has not been applied during the last γ iterations (γ is the tabu tenure that takes a random value from a given range). The tabu status of a move is neglected only if the move leads to a new solution better than the best solution found so far. The *directed perturbation* relies thus both on (1) history information which keeps track, for each move, of the last iteration (time) when it was performed and (2) the quality of the moves to be applied for perturbation in order not to deteriorate too much the perturbed solution. The eligible moves for the directed perturbation are identified by the set A such that:

$$A = \{swap(u, v) | min\{\delta(\pi, u, v)\}, (H_{uv} + \gamma) < Iter \text{ or } (\delta(\pi, u, v) + c) < c_{best}, u \neq v \quad and\ 1 \leqslant u, v \leqslant n\}$$

where H is the matrix that keeps track of the iteration number when a move was last performed, *Iter* the current iteration number, c the cost of the current solution, and c_{best} the cost of the best solution discovered so far. A larger value of γ implies a stronger diversification.

The *recency-based perturbation* relies only on the piece of history information provided by the H matrix. It favors the least recently performed moves regardless of the cost degradation caused by the move. More formally, these moves are identified by the set B such that:

$$B = \{swap(u, v) | min\{H_{uv}\}, u \neq v \ and\ 1 \leqslant u, v \leqslant n\}$$

Finally, the *random perturbation* simply performs moves that are selected uniformly at random. Those moves are identified as follows.

$$C = \{swap(u, v) | u \neq v \quad \text{and} \quad 1 \leqslant u, v \leqslant n\}$$

It should be clear that these perturbation types have an increasingly stronger diversification effect.

In order to insure the best balance as possible between an intensified and a diversified search, BLS takes turns probabilistically between these three types of perturbations. The probability of applying a particular perturbation is determined dynamically depending on the search state, i.e., the current number ω of consecutive non-improving local optima visited. ω is reset to zero each time the best solution found during the search is improved, or when the number of consecutive non-improving local optima exceeds a given threshold T. The idea is to apply more often the directed perturbation (with a higher probability) at the beginning of the search, i.e., as the search progresses towards improved new local optima (ω is small). With the increase of ω, the probability of using the directed perturbation progressively decreases while the probability of applying the random and the recency-based moves increases for the purpose of a stronger diversification.

Additionally, it has been observed from an experimental analysis that it is useful to guarantee a minimum of applications of the directed perturbation. Therefore, we constrain the probability P of applying the directed perturbation to take values no smaller than a threshold P_0:

$$P = \begin{cases} e^{-\omega/T}, & \text{if,} \quad e^{-\omega/T} > P_0 \\ P_0 & \text{otherwise} \end{cases} \tag{4}$$

where T is the maximum number of non-improving local optima visited before triggering a stronger perturbation.

Given probability P of applying the directed perturbation, the probability of applying the recency-based and the random perturbation is determined respectively by $(1-P)\cdot Q$ and $(1-P)\cdot(1-Q)$ where Q is a constant from $[0,1]$ (see Algorithm 2).

Once the type of perturbation is determined, we apply accordingly the corresponding moves identified by the sets A, B or C to the current solution L times (see Algorithm 3). The resulting solution is used by the next round of the steepest descent local search as its new starting point.

An analysis on the impact of different perturbation types is presented in Section 5.

3.5. Discussion

BLS combines some features from several well-established metaheuristics: iterated local search (ILS) [25], tabu search (TA) [18] and simulated annealing (SA) [22]. We briefly discuss the similarities and differences between our BLS approach and these methods.

Like ILS, BLS uses local search to discover local optima and perturbation to diversify the search. However, BLS has a particular focus on the importance of perturbation, and uses an adaptive and multi-type perturbation mechanism. The three types of perturbations are triggered according to the search status, leading to variable levels of diversification. The directed perturbation of BLS is based on the notion of tabu list that is borrowed from TS. The recency-based perturbation also finds its sources in TS where recency related information may be used for the purpose of both intensification and diversification. Nevertheless, BLS does not consider the tabu list during its descent phases while each iteration of TS is constrained by the tabu list. As such, BLS and TS may explore different trajectories during their respective search, leading to different local optima. Since the local search phase of BLS is a simple descent procedure which bases its moves only on the quality criterion, the diversification is completely eliminated during this phase, unlike TS and SA for which the intensification and diversification are always intertwined. In fact, we strongly believe that this constitutes one of the keys to BLS effectiveness.

A further distinction of BLS is the way an appropriate perturbation type is selected at a certain stage of the search. As explained in Section 3.4.2, BLS adaptively changes the probability of applying a weak perturbation according to the current state of search. The idea for this adaptive change of probability finds its inspiration from SA, and enables a better balance between an intensified and diversified search.

As we will see in the next section, BLS is able to attain highly competitive results on the set of well-known QAP benchmarks.

4. Experimental results

4.1. Experimental protocol

This section is dedicated to an extensive experimental evaluation of the proposed BLS algorithm on a wide range of benchmark instances from the QAPLIB.[1] The instance size n varies from 12 to 150, and is indicated in the instance name. The QAPLIB archive comprises 135 instances that can roughly be classified into four types:

I. Real-life instances obtained from practical applications of QAP;
II. Unstructured, randomly generated instances for which the distance and flow matrices are randomly generated based on a uniform distribution;
III. Randomly generated instances with structure that is similar to that of real-life instances;
IV. Instances in which distances are based on the Manhattan distance on a grid.

[1] http://www.seas.upenn.edu/qaplib/inst.html.

Table 1
Settings of important parameters.

Para.	Description	Value
L_0	Initial jump magnitude	$0.05n$ (II), $0.15n$ (I, III, IV)
L_{Max}	Maximal jump magnitude	$rand[0.4n, 0.6n]$
T	Max. number of non-improving attractors visited before strong perturb	2500
γ	Tabu tenure	$rand[0.9n, 1.1n]$
P_0	Smallest probability for applying directed perturbation	0.75
Q	Probability for applying random over recency-based perturb.	0.7

Among the 135 instances, 101 instances (including all the real-life instance) can be considered as easy since BLS (and many other state-of-art QAP methods) can solve them to optimality in every single trial within a very short computation time (often less than a second). Since these easy instances do not represent any challenge for BLS, they are not considered in our experimentation.

Our BLS algorithm[2] is programmed in C++, and compiled with GNU gcc on a Xeon E5440 with 2.83 GHz and 8 GB.

Except for the initial jump magnitude L_0, we apply the same setting of parameters to all the tested problem instances. Parameter L_0 is particularly sensitive to the different structures of QAP instances. Therefore, for unstructured instances (type II) $L_0 = 0.05n$, while for other types $L_0 = 0.15n$. A justification for these values of L_0 is provided in Section 5.2. The setting of all the parameters is determined by a preliminary experiment and is given in Table 1. The stopping condition is the elapsed time which we set to 2 hours for all the instance of size $n \leqslant 100$, and to 10 hours for the two largest instances (*tai*150*b* and *tho*150) of size $n = 150$. As shown below, the best-known solutions are very often attained long before these time limits.

According to the literature [12,29,35], we employ several criteria for evaluation and comparison of BLS with other QAP approaches.

(1) The number of instances for which an optimal or best-known solution is reached within a reasonable computing time;
(2) The success rate of reaching an optimal or best-known solution;
(3) The percentage deviation $\bar{\delta}_{avg}$ of the average solution from the published best-known result over a certain number of runs. The percentage deviation between solutions is computed as $\bar{\delta} = 100(z - \bar{z})/\bar{z}[\%]$, where z is the best or average solution over a given number of runs and $\bar{z}$ the best-known objective value.

Beside the aforementioned criteria, computing times are also mentioned for indicative purposes.

4.2. Computational results and comparisons

We compare our BLS approach to some of the leading algorithms from the literature. The following algorithms are considered (see Section 2 for a short description):

(1) Cooperative parallel tabu search (CPTS) algorithm [20];
(2) Three tabu search variants (ETS1, ETS2, ETS3) [30];
(3) Iterated tabu search (ITS) [31];
(4) Improved hybrid genetic algorithm (IHGA) [29];
(5) Population-based iterated local search (PILS) [35];
(6) A variant of a hybrid genetic tabu search algorithm (MRT60) [12].

The results of the reference algorithms are extracted from the corresponding papers. A comparative analysis of such results is not a straightforward task because of the differences in computing hardware, result reporting methodology and termination criterion. Indeed, some papers only report the maximum time limit per execution (e.g. [29–31,12]) while others (e.g. [20,35]) report the average time needed to reach the best results over a number of trials. Moreover, some algorithms terminate after a fixed number of iterations, while others terminate after a fixed amount of CPU time. This comparison is thus presented only for indicative purposes and should be interpreted with caution. Nevertheless, this experimental study shows to some extent the performance of the proposed algorithm relative to these state-of-art algorithms.

Table 2 reports comparative results with CPTS, ETS-1, ETS-2, ETS-3, ITS, PILS and IHGA for unstructured instances (type II) and real-life like instances (type III). The best results are indicated in bold. Note that MRT60 is not used in this comparison because it only reports results for instances of type IV. The second column 'BKS' shows for each instance the best-known objective value ever reported in the literature. For each algorithm, column $\bar{\delta}_{avg}$ indicates the percentage deviation of the average solution, obtained with the given approach over a certain number of trials, from the best-known solution. If known, the success rate for reaching the best-known solution over 10 trials is given in parentheses next to the value of $\bar{\delta}_{avg}$. The CPU time (in minutes) is

[2] The source code of the BLS algorithm will be available online at: http://www.info.univ-angers.fr/pub/hao/BLS_code.cpp.

Table 2
Comparative results between BLS and some of the best performing QAP approaches on unstructured instances (type II) and real-life like instances (type III). The times are given in minutes. If available, the success rate of reaching the best-known result over 10 executions is indicated between parentheses.

Problem	BKS	BLS % $\bar{\delta}_{avg}$	$t(m)$	CPTS [20]% $\bar{\delta}_{avg}$	$t(m)$	ETS-1 [30]% $\bar{\delta}_{avg}$	ETS-2 [30]% $\bar{\delta}_{avg}$	ETS-3 [30]% $\bar{\delta}_{avg}$	$t(m)$	PILS [35]% $\bar{\delta}_{avg}$	$t(m)$	ITS [31]% $\bar{\delta}_{avg}$	$t(m)$	IHGA [29]% $\bar{\delta}_{avg}$	$t(m)$
Random instances (type II)															
tai20a	703482	**0.000**(10)	0.0	**0.000**(10)	0.1	**0.000**(10)	**0.000**(10)	**0.000**(10)	0.0	**0.000**(10)	0.0	0.060(8)	0.0	**0.000**(10)	0.0
tai25a	1167256	**0.000**(10)	0.0	**0.000**(10)	0.3	0.037	**0.000**(10)	0.015	0.1	**0.000**(10)	0.2	**0.000**(10)	0.0	**0.000**(10)	0.1
tai30a	1818146	**0.000**(10)	0.0	**0.000**(10)	1.6	0.003	0.041	**0.000**(10)	0.2	**0.000**(10)	0.6	**0.000**(10)	0.1	**0.000**(10)	0.3
tai35a	242002	**0.000**(10)	0.2	**0.000**(10)	2.3	**0.000**(10)	**0.000**(10)	**0.000**(10)	3.7	**0.000**(10)	2.3	**0.000**(10)	0.3	**0.000**(10)	0.6
tai40a	3139370	**0.022**(7)	38.9	0.148(1)	3.5	0.167	0.130	0.173	28.3	0.280(0)	12.0	0.210(1)	0.8	0.209(1)	1.4
tai50a	4938796	**0.157**(2)	45.1	0.440(0)	10.3	0.375(0)	0.407(0)	0.441(0)	116.7	0.663(0)	11.2	0.373(0)	3.0	0.262(0)	5.0
tai60a	7205962	**0.251**(1)	47.9	0.476(0)	26.4	0.606(0)	0.639(1)	0.713(0)	116.7	0.820(0)	7.4	0.330(1)	9.7	0.583(0)	12
tai80a	13499184	0.517(0)	47.3	0.691(0)	94.8	0.757(0)	0.826(0)	0.841(0)	200.0	0.927(0)	12.7	**0.494**(0)	25.0	0.756(0)	53.3
tai100a	21052466	0.430(0)	39.0	0.589(0)	261.2	0.717(0)	0.718(0)	0.790(0)	666.7	1.027(0)	9.8	**0.427**(0)	60.0	0.606(0)	200.0
Average		0.153	24.3	0.260	44.5	0.296	0.307	0.330	125.8	0.413	6.24	0.210	11.0	0.268	30.3
Real-life like instances (type III)															
tai20b	122455319	**0.000**(10)	0.0	**0.000**(10)	0.1	**0.000**(10)	**0.000**(10)	**0.000**(10)	0.0	**0.000**(10)	0.0	**0.000**(10)	0.0	**0.000**(10)	0.0
tai25b	344355646	**0.000**(10)	0.0	**0.000**(10)	0.4	**0.000**(10)	**0.000**(10)	**0.000**(10)	0.0	**0.000**(10)	0.0	**0.000**(10)	0.0	**0.000**(10)	0.0
tai30b	637117113	**0.000**(10)	0.0	**0.000**(10)	1.2	**0.000**(10)	**0.000**(10)	**0.000**(10)	0.1	**0.000**(10)	0.0	**0.000**(10)	0.0	**0.000**(10)	0.0
tai35b	283315445	**0.000**(10)	0.0	**0.000**(10)	2.4	**0.000**(10)	0.019	**0.000**(10)	0.3	**0.000**(10)	0.0	**0.000**(10)	0.1	**0.000**(10)	0.0
tai40b	637250948	**0.000**(10)	0.0	**0.000**(10)	4.5	**0.000**(10)	**0.000**(10)	**0.000**(10)	0.3	**0.000**(10)	0.0	**0.000**(10)	0.2	**0.000**(10)	0.1
tai50b	458821517	**0.000**(10)	2.8	**0.000**(10)	13.8	**0.000**(10)	0.003	**0.000**(10)	4.5	**0.000**(10)	0.1	**0.000**(10)	0.9	**0.000**(10)	0.3
tai60b	608215054	**0.000**(10)	5.6	**0.000**(10)	30.4	**0.000**(10)	0.001	0.003	22.5	**0.000**(10)	0.2	**0.000**(10)	2.2	**0.000**(10)	0.7
tai80b	818415043	**0.000**(10)	11.4	**0.000**(10)	110.9	0.008	0.036	0.016	116.7	**0.000**(10)	1.3	**0.000**(10)	5.8	**0.000**(10)	2.5
tai100b	1185996137	**0.000**(10)	16.0	0.001(8)	241.0	0.072	0.123	0.034	333.3	**0.000**(10)	2.3	**0.000**(9)	23.3	**0.000**(10)	7.3
tai150b	498896643	**0.075**(1)	243.6	0.076(0)	7377.8	–	–	–	–	0.095(0)	36.7	0.100(1)	60.0	0.111(2)	38.3
Average		0.008	27.9	0.008	778.3	0.009	0.020	0.006	53.1	0.010	4.1	0.010	9.3	0.011	4.9

Table 3
Comparative results between BLS and some of the best performing QAP approaches on grid-based (type IV) instances. The times are given in minutes. The success rate of reaching the best-known result over 10 executions is indicated between parentheses.

Problem	BKS	BLS % $\bar{\delta}_{avg}$	$t(m)$	CPTS [20]% $\bar{\delta}_{avg}$	$t(m)$	PILS [35]% $\bar{\delta}_{avg}$	$t(m)$	MRT60 [12]% $\bar{\delta}_{avg}$	$t(m)$
sko42	15812	**0.000**(10)	1.7	**0.000**(10)	5.3	**0.000**(10)	2.8	–	–
sko49	23386	**0.000**(10)	0.5	**0.000**(10)	11.4	**0.000**(10)	0.8	**0.000**(10)	4.3
sko56	34458	**0.000**(10)	1.1	**0.000**(10)	21.0	**0.000**(10)	0.5	**0.000**(10)	7.2
sko64	48498	**0.000**(10)	1.3	**0.000**(10)	42.9	**0.000**(10)	0.2	**0.000**(10)	12.4
sko72	66256	**0.000**(10)	4.1	**0.000**(10)	69.6	0.001(8)	6.7	**0.000**(10)	19.9
sko81	90998	**0.000**(10)	13.9	**0.000**(10)	121.4	0.007(5)	9.6	**0.000**(10)	31.9
sko90	115534	**0.000**(10)	16.6	**0.000**(10)	193.7	0.006(4)	10.6	**0.000**(10)	48.5
sko100a	152002	0.001(9)	20.8	**0.000**(10)	304.8	0.012(3)	7.9	**0.000**(10)	73.6
sko100b	153890	0.**000**(10)	10.8	**0.000**(10)	309.6	0.007(5)	7.3	**0.000**(10)	73.6
sko100c	147862	**0.000**(10)	15.5	**0.000**(10)	316.1	0.002(6)	11.5	**0.000**(10)	73.6
sko100d	149576	0.001(5)	38.9	**0.000**(10)	309.8	0.021(0)	11.8	**0.000**(10)	73.6
sko100e	149150	**0.000**(10)	42.5	**0.000**(10)	309.1	0.001(7)	6.8	**0.000**(10)	73.6
sko100f	149036	**0.000**(10)	17.3	0.003(4)	310.3	0.037(0)	11.7	0.000(9)	43.5
wil100	273038	**0.000**(10)	18.9	**0.000**(10)	316.6	0.004(1)	6.3	**0.000**(10)	73.6
tho150	8133398	0.023(1)	268.8	0.013(0)	1991.7	0.068(0)	36.2	**0.003**(3)	1223.6
Average		0.002	31.5	0.001	308.8	0.011	8.7	0.000	130.9

only given for indicative purposes. The last line shows the averaged information. From the results in Table 2, we can make the following observations.

For the unstructured instances (type II), BLS finds the best-known solution for 7 out of the 9 instances, with an average deviation $\bar{\delta}_{avg}$ of 0.153 over the 9 instances. For three hard instances *tai*40*a*, *tai*50*a* and *tai*60*a*, it reaches the best-known solution in 70%, 20% and 10% of the trials respectively. BLS thus outperforms ITS which is probably the current most effective approach for this type of instances. Indeed, ITS is able to attain the best-known result for 6 out of the 9 instances with an average deviation $\bar{\delta}_{avg}$ of 0.210 over the 9 instances. For instances *tai*40*a* and *tai*60*a*, ITS reaches the best-known result with a success rate of 10% but is unable to find the best-known solution for *tai*50*a* as it is the case with the five other reference QAP approaches.

For the real-life like instances (type III), BLS is able to attain the best-known solution for all the instances in every single trial, except for the largest instance *tai*150*b* where the success rate is 10%. BLS reports an average deviation $\bar{\delta}_{avg}$ of 0.008 over the 10 instances. Two reference approaches, ITS and IHGA, are also able to attain the best-known result for all the 10 real-life like instances with an average deviation $\bar{\delta}_{avg}$ of 0.010 and 0.111 respectively. Notice that the average deviations of ETS cannot be used for comparisons since the ETS algorithms do not report results for the hardest real-life like instance *tai*150*b* of this type. In summary, BLS is very competitive with the best performing QAP approaches for real-life like instances.

Table 3 reports comparative results of BLS with CPTS, PILS and MRT60 for instances with grid distances (type IV). The other reference approaches are not included in this comparison since they do not report results for these instances. From the results in Table 3, we observe that BLS is able to reach the best-known results at each trial for 12 out of 15 instances. It is even capable of attaining the best-known solution for the hardest grid-based instance *tho*150 with a success rate of 10%. The current best algorithm on instances of type IV is MRT60. Indeed, MRT60 attains the best-known result for instance *tho*150 in 30% of the trials and reports an average deviation $\bar{\delta}_{avg}$ of approximately 0 (against 0.002 for BLS) over the 15 instances. Beside MRT60 and BLS, the other reference algorithms are unable to reach the best-known result for *tho*150 although CPTS attains a low average deviation of 0.001. Notice that the performance of this genetic tabu hybrid MRT60 is unknown for the instances of types II and III.

Finally, as to the computing times, BLS requires on average 27.9 min to reach its best solution. Although it is impossible to make a precise comparison of computing times, our BLS algorithm does not seem to require more computing time than some other reference approaches (e.g. [12,20,30]).

5. Experimental analysis and discussion

It is well known that the performance of any heuristic algorithm depends on the balance between intensification and diversification. However, the desired balance is influenced by the structure of the given instance, such as the distribution of local optima, the correlation between solutions, the number of global optima, etc. An analysis of correlation between solution quality and distance to global optimum has shown to provide useful indications about the problem hardness and the distribution of local optima in the search space. Such an analysis on QAP instances has already been performed in [27,35] but using different local search procedures to sample candidate solutions and global optima which later turned out to be only locally optimal solutions. Therefore, for the sake of greater accuracy, we carry out the same analysis based on solutions sampled with our BLS approach and the current best-known solutions (global optima). In addition, we analyze the distribution of local optima in search space by investigating minimal distances between two medium or high quality local optima. These solutions may be viewed as 'strong' attractors since they may be more likely visited during the search than a low

quality local optimum. Based on the observations made from these analyses, we justify our choice for the used parameter settings. Moreover, we investigate the impact of the proposed multi-type perturbation strategy on BLS performance.

5.1. Landscape analysis

5.1.1. Analysis protocol

We perform the landscape analysis on 16 QAPLIB instances, based on a set of distinct solutions obtained after 2000 independent runs of the BLS approach. The number of iterations per run is set to 200000. As the distance between solutions, we calculate the number of locations that are allocated to distinct facilities in two solutions π and π', i.e., $d(\pi,\pi') = |\{i|\pi_i \neq \pi'_i\}|$. Since global optima for the analysed instances are not known, we use instead the best-known local optima to compute fitness-distance correlation and refer to them as global optima. We take into account for this analysis that some QAP instances have multiple global optima. This is the case with grid-based instances (type IV) and one random instance *tai*60*a* for which we have found two different global optima (i.e., best-known solutions).

5.1.2. FDC and distribution of local optima

The fitness-distance correlation (FDC) coefficient ρ [24] captures the correlation between the fitness (i.e., quality) of a solution and its distance to the nearest global optimum (or best-known solution if global optimum is not available). A value of $\rho = 1$ indicates perfect correlation between fitness and distance to the optimum, implying that the better the fitness of a solution the closer the solution is to the optimum. For correlation of $\rho = -1$, the evaluation function is completely misleading. FDC can also be visualized with a FD plot, where the same data used for estimating ρ is displayed graphically. Such plots have been used to estimate the distribution of local optima for a number of problems including for instance TSP [7], graph partitioning [3,28], flow-shop scheduling problem [33], and QAP [27,35].

The results of our analysis on the 16 QAP instances are presented in Table 4. In column 'ρ' of Table 4, we report FDC coefficients for these instances. For illustrative purposes, FD plots for three instances (*tai*100*a*, *sko*100*a* and *tai*100*b*) are given in Fig. 2. As it can be seen from the FDC coefficients in Table 4, there is a clear difference in correlation among instances of different types. For unstructured instances (type II), the FDC coefficient ρ is negative except in one case (*tai*50*a*) where ρ is close to zero. Indeed, from the FD plots in Fig. 2 it is clear that there is no correlation between fitness and distance for *tai*100*a*, which implies that these instances are hard for any heuristic algorithm.

For grid-based instances (type IV), significant FDC exists except in one case (*sko*81, $\rho < 0$), while for two real-life like instances *tai*80*b* and *tai*100*b* the FDC is also low $\rho < 0.15$ but higher than for the unstructured instances. The FD plots in Fig. 2 for instances *sko*100*a* and *tai*100*b* confirm this observation.

Table 4 additionally reports the average distance between local optima *avg* d_{lo}, the average distance between local optima and the nearest global optimum *avg* d_{go}, and the average distance between the highest quality local optima *avg* d_{hq}. It can be observed that, regardless of the instance type, the values of *avg* d_{lo}, *avg* d_{go}, and *avg* d_{hq} are very large, close to the maximal possible value n. This implies that local optima are scattered all over the search space. Even high quality local optima are distant from each other and share no common structure.

To gain an even better insight in the distribution of local optima in the search space, we investigate minimal distances between two medium or high quality local optima (i.e., 'strong' attractors). Such analysis, previously applied to the maximum clique problem [4], has shown to provide some useful indications on the influence of different diversification strengths to algorithm performance. More precisely, given a set of medium or high quality local optima S, for all $lo_i \in S$ we determine the distance d_{min} between lo_i and some other solution $lo_j \in S$ which is the closest to lo_i, i.e., $d_{min} = min_{lo_j \in S, lo_j \neq lo_i} d(lo_i, lo_j)$. For

Table 4
Analytical results for 16 QAP instances. Column '#*dlo*' indicates the number of distinct local optima over 2000 independent runs of BLS; Columns '*avg* d_{lo}', '*avg* d_{go}' and '*avg* d_{hq}' report respectively the average distance between local optima, the average distance between local and global optima, and the average distance between the highest quality local optima; Column 'ρ' shows the value of the fitness-distance correlation coefficient.

Instance	#*dlo*	avg d_{lo}	avg d_{go}	avg d_{hq}	ρ
tai40a	349	38.5	39.2	38.4	−0.028
tai50a	1760	48.6	49.1	48.7	0.029
tai60a	1995	58.6	59.2	58.6	−0.024
tai80a	2000	78.8	79.0	78.8	−0.058
tai100a	2000	98.8	99.0	98.9	−0.012
tai80b	730	61.7	78.7	68.7	0.040
tai100b	1007	80.7	97.7	89.1	0.11
sko72	228	66.8	68.3	65.7	0.287
sko81	388	75.4	74.5	76.5	−0.048
sko90	487	85.08	85.7	82.8	0.348
sko100a	968	95.4	94.2	95.4	0.366
sko100b	628	95.9	92.5	95.5	0.280
sko100c	559	91.0	91.3	93.9	0.233
sko100d	1212	96.3	93.9	96.3	0.433
sko100e	545	94.3	92.4	96.2	0.569
sko100f	1230	96.6	93.5	97.2	0.317

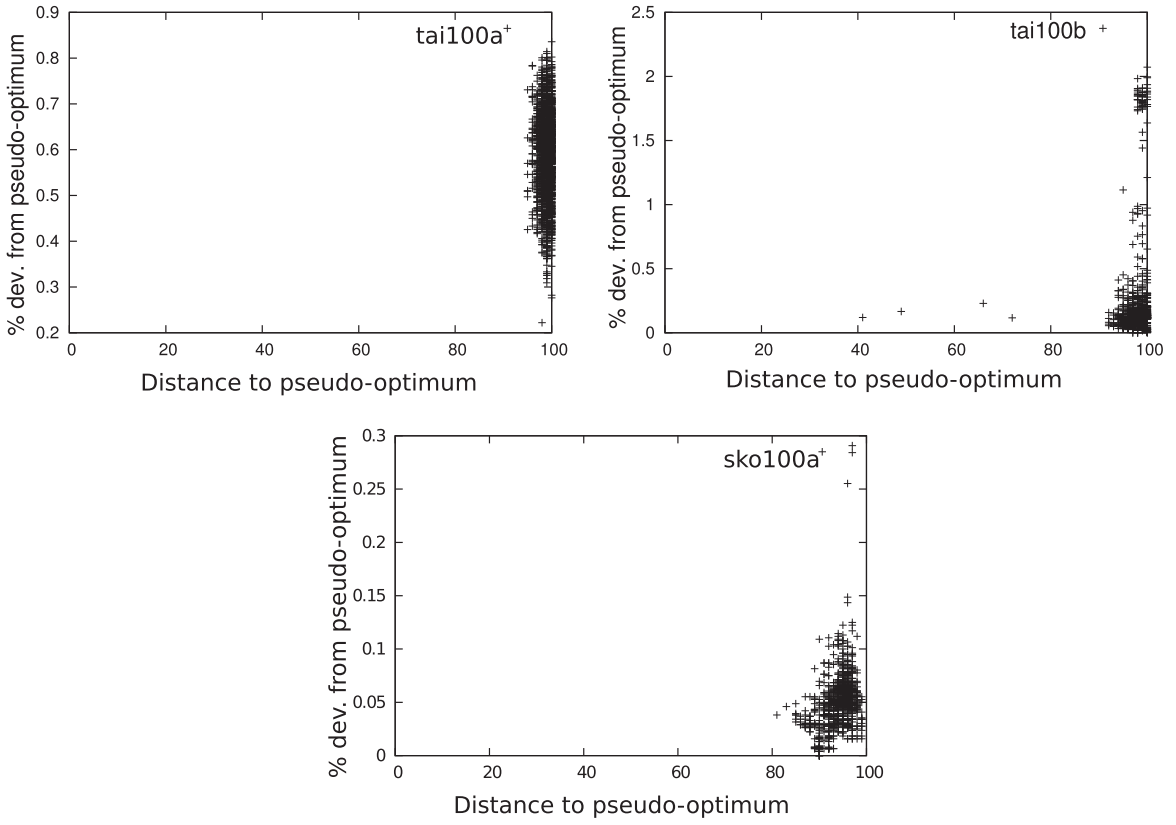

Fig. 2. Distance of local optima to the best-known solution based on solutions sampled by the BLS algorithm for instances *tai*100*a* (type II), *tai*100*b* (type III) and *sko*100*a* (type IV).

each possible distance $d_i \in [0, n]$, we then count the number of time that d_i is the distance between $lo \in S$ and another solution in S which is the closest to lo. The results of this study on three instances (*tai*100*a*, *tai*100*b* and *sko*100*a*) are given in Fig. 3. The x-axis shows the minimal distance between two 'strong' attractors, while the y-axis shows the number of pairs of 'strong' attractors separated by the given distance.

Fig. 3 indicates that there exists a significant difference in the distribution of medium and high quality local optima for QAP instances. For *tai100b* and *sko100a*, the minimal distances between two 'strong' attractors are significantly smaller than in the case of the unstructured instance *tai100a*. Intuitively, a weaker diversification introduced into the search for such instances may cause the search to cycle between two or more 'strong' attractors that are not necessarily globally optimal solutions. For an effective solving of these instances, strong diversifications are needed as illustrated in the next sections.

5.2. Justification for parameter settings

While the intensification mechanism of BLS is parameter-free, the diversification is controlled by the following parameters: the initial number of perturbation moves L_0, the number of perturbation moves for the strongest perturbation L_{Max}, the maximal number of non-improving attractors visited before strong perturbation T, the tabu tenure γ, and the smallest probability for applying directed perturbation P_0. The optimal setting of these parameters depends on landscape properties which, as seen from the landscape analysis, significantly differ for QAP instances. As a result, some state-of-art QAP approaches (e.g. [11,12,29–31]) report results only for a specific type of QAP instances. Despite the fact that all the BLS parameters have a role in establishing the right balance between intensification and diversification, experimental evaluations have shown that they are not equally sensitive to different structures of instances. While parameters L_{Max} and T show no particular sensitivity, the most sensitive seems to be the initial perturbation strength parameter L_0. For that reason, the reported results (for all the types of QAP instances) were obtained with the same setting of parameters, except for parameter L_0 whose value is varied according to the type of instance (see Table 1).

To illustrate the influence of the initial jump magnitude L_0 on solution quality, we run our BLS algorithm on three QAP instances (*tai*60*a* [type II], *tai*100*b* [type III], *sko*90 [type IV]) with a small and large value of L_0 which is set to $0.05n$ and $0.15n$ respectively, where n is the instance size. The results for each instance are plotted in Fig. 4, where the x-axis indicates the execution number and y-axis the percentage deviation from the best-known solution ever reported in the literature. The

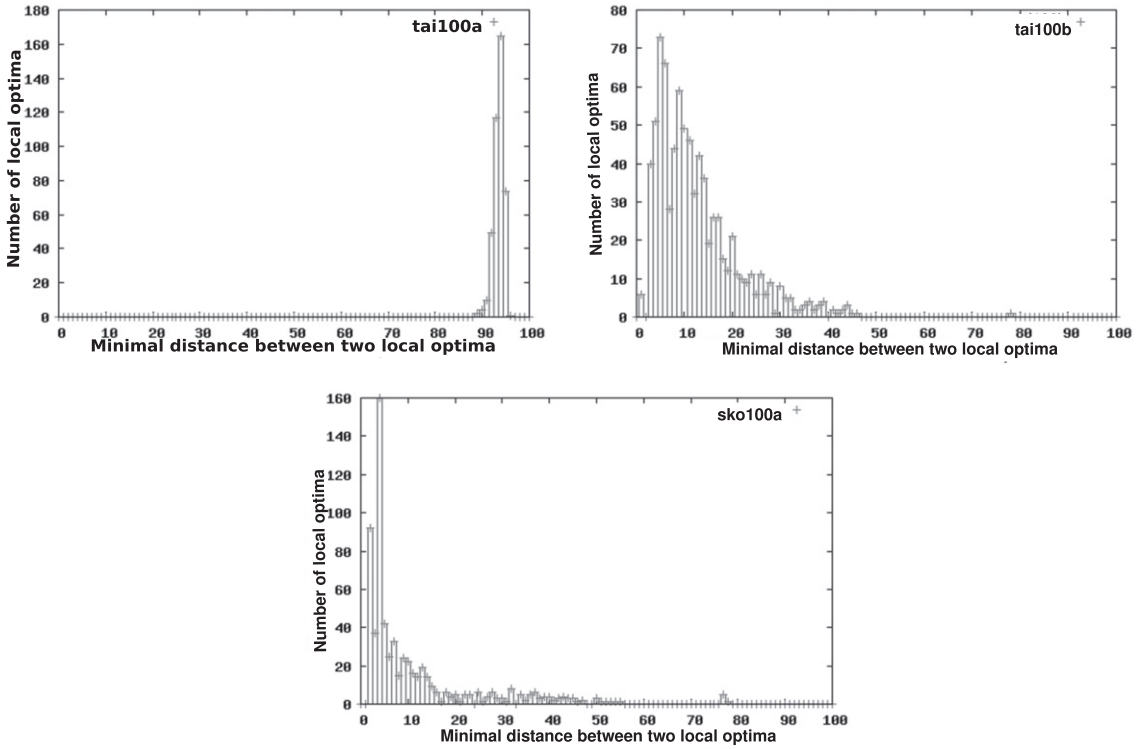

Fig. 3. Distribution of medium and high quality local optima (i.e., 'strong' attractors) for *tai*100*a* (type II), *tai*100*b* (type III) and *sko*100*a* (type IV) instances.

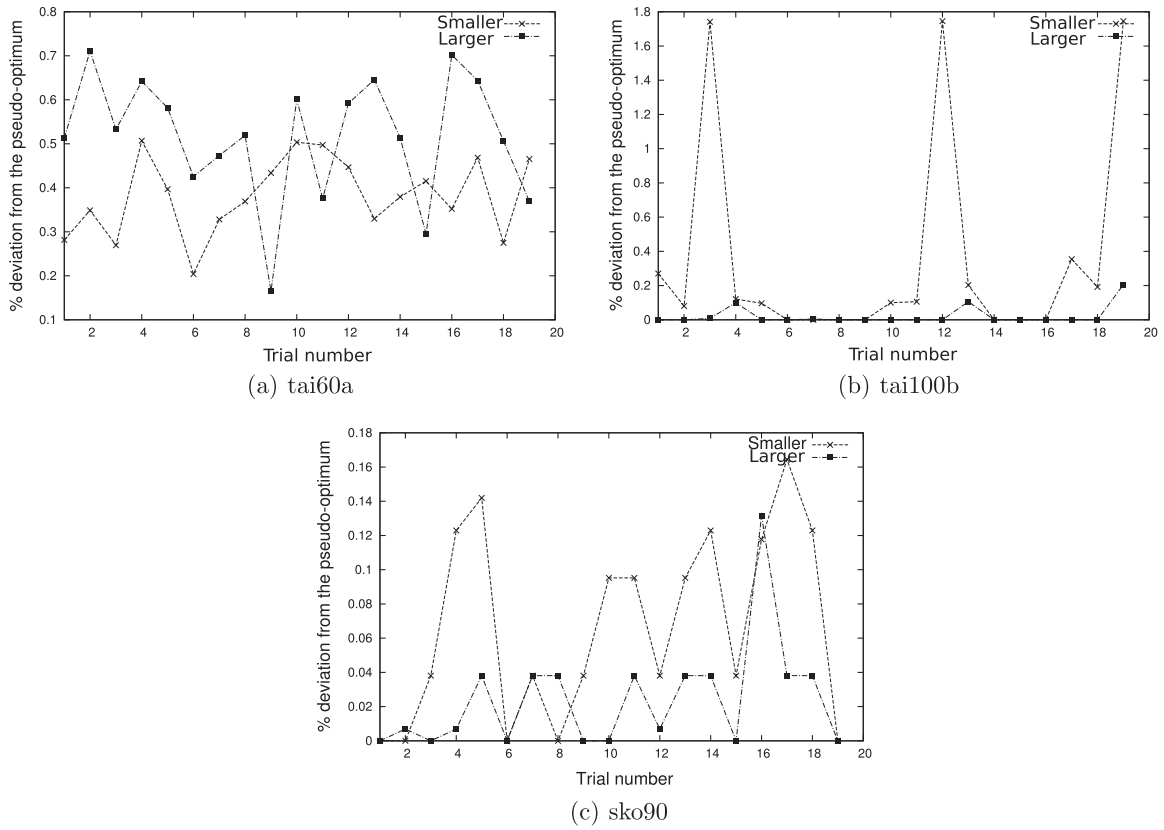

Fig. 4. Influence of different jump magnitudes on solution quality for instances tai60a (type II), tai100b (type III) and sko90 (type IV).

plots indicate that jumps of small magnitude, which induce less diversification, provide significantly better results for the unstructured instance *tai*60*a* than when larger jumps are performed. On the other hand, considerably better results are obtained when performing jumps of larger magnitude on instances of types III and IV. Although we do not present results of this analysis for other QAP instances, they generally exhibit the same behaviour.

An explanation for such results can be found in the landscape analysis. Fig. 3 indicates that the minimal distances between two high quality local optima are generally very large for the unstructured instances (type II), and are close to the maximal possible value. Intuitively, such distribution of local optima prevents the search from cycling even if a weak diversification is introduced into the search. For this reason it may be worthwhile to perform a more intensive search.

On the other hand, Fig. 3 shows that the situation for instances *tai100b* (type III) and *sko90* (type IV) is quite the opposite. Indeed, the minimal distances between any two high quality solutions are generally rather small for these two types of instances. Introducing a weaker form of diversification may thus cause the search to cycle between two or more local optima which are not necessarily the optimal solutions.

5.3. Importance of adaptive perturbation

BLS jointly uses three types of perturbations (directed, recency-based and random) and applies them in an adaptive way. We use the term "undirected perturbation" to group both recency-based and random perturbation. We show in this section that if directed and undirected perturbations are applied separately, the performance of BLS is affected.

We perform three experiments to examine the impact of the directed and the undirected perturbation, as well as the importance of the adaptive perturbation strategy adopted in BLS. For the first experiment, the three perturbation types are jointly used by BLS (as presented in Algorithm 2). In the second experiment, only the directed perturbation is used, while for the third experiment only the undirected perturbation is employed. The plotted results of this analysis are presented in Fig. 5. The *x*-axis indicates the execution number, while the *y*-axis shows the percentage deviation from the best-known solution ever reported in the literature.

We observe that the directed perturbation, which introduces less diversification into the search, provides much better results than the undirected perturbation for instances of type II. However, the results are quite opposite for instances of types III and IV where the undirected perturbation seems to be more effective. This coincides with our observation that weaker diversification is more effective on unstructured instances (type II), while stronger diversification works better on other QAP instances.

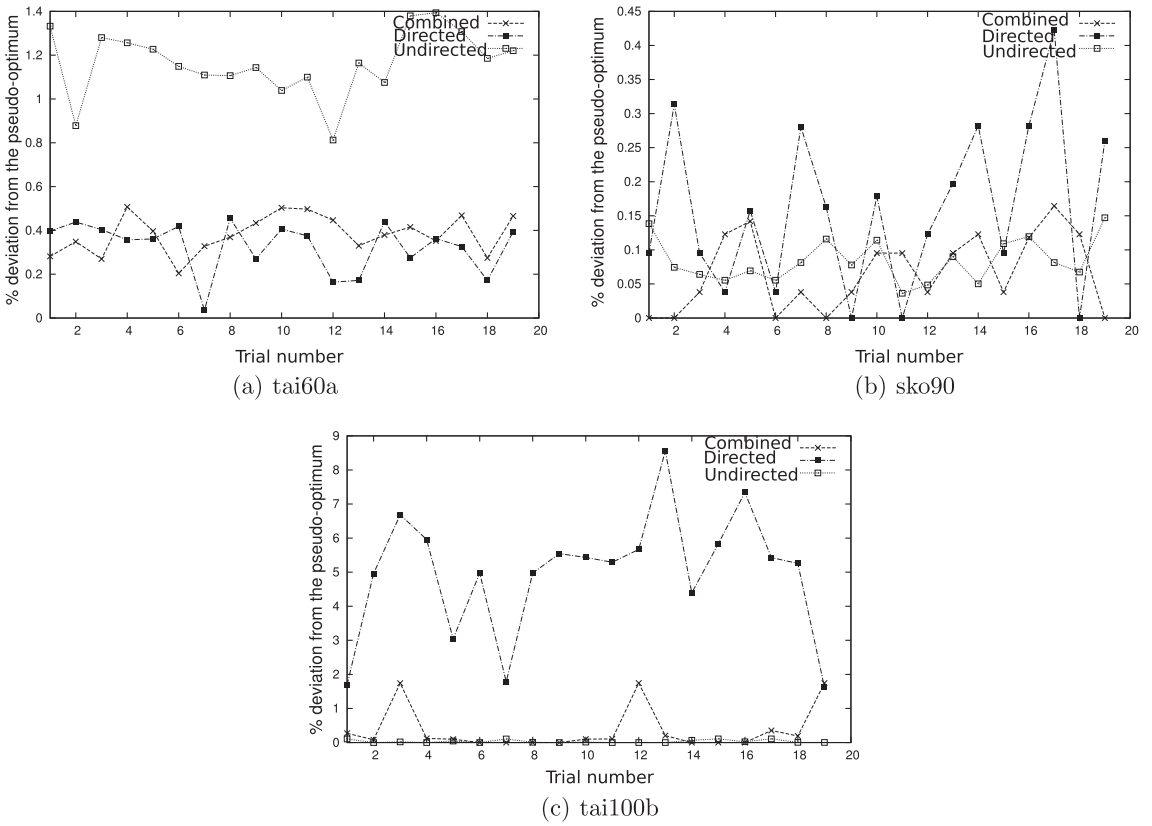

Fig. 5. Influence of different perturbation types on instances *tai*60*a* (type II), *sko*90 (type III) and *tai*100*b* (type IV).

On the other hand, a combined use of directed and undirected perturbation provides quality solutions independent of the instance type. This confirms the interest of an adaptive combination of guided and random perturbation within a diversification mechanism.

6. Conclusion

In this paper, we presented the breakout local search approach for QAP. BLS alternates between a local search phase (to find local optima) and a perturbation-based diversification phase (to jump from a local optimum to another local optimum). To be effective, the local search procedure uses the steepest descent strategy combined with a very fast neighborhood evaluation technique from the literature. To visit local optima of high quality, the jumps toward new local optima are adaptively controlled according to the state of search. This is achieved by varying the magnitude of a jump and selecting the most suitable perturbation for each diversification phase.

Experimental evaluations on the collection of benchmark instances from the QAPLIB showed that the proposed approach competes very favorably with the current most effective (often more complex) QAP algorithms. In particular, BLS is able to reach the current best solution for all the benchmark instances except for 2 cases in reasonable computing times. To the best of our knowledge, this is the best overall result obtained by a single QAP approach. BLS performs particularly well on unstructured instances (type II) which are considered to be the hardest for many current QAP approaches. For the instances with grid distances (type III) and the real-life like instances (type IV), BLS also remains very competitive with respect to the best performing approaches.

Acknowledgment

This work was partially supported by the Region of "Pays de la Loire" (France) within the Radapop and LigeRO Projects. We are grateful to the anonymous referees for valuable suggestions and comments which helped us improve the paper.

References

[1] R.K. Ahuja, J.B. Orlin, A. Tiwari, A greedy genetic algorithm for the quadratic assignment problem, Computers & Operations Research 27 (10) (2000) 917–934.
[2] R. Battiti, G. Tecchiolli, The reactive tabu search, ORSA Journal of Computing 6 (2) (1994) 126–140.
[3] U. Benlic, J.K. Hao, A multilevel memetic approach for improving graph k-partitions, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 624–642.
[4] U. Benlic, J.K. Hao, Breakout local search for maximum clique problems, Computers & Operations Research 40 (1) (2013) 192–206.
[5] U. Benlic, J.K. Hao, Breakout local search for the max-cut problem, Engineering Applications of Artificial Intelligence (in press). http://dx.doi.org/10.1016/j.engappai.2012.09.001.
[6] U. Benlic, J.K. Hao, A study of breakout local search for the minimum sum coloring problem, in: L.T. Bui et al. (Eds.) SEAL 2012 Lecture Notes in Computer Science, vol. 7673, 2013, pp. 128–137.
[7] K.D. Boese, Cost versus distance in the traveling salesman problem, Technical Report TR-950018 UCLA CS Department, 1995.
[8] R.E. Burkard, Locations with spatial interactions: The quadratic assignment problem, in: P. Mirchandani and L. Francis (Eds.), Discrete Location Theory, New York, 1991.
[9] R.E. Burkard, E. Çela, P.M. Pardalos, L. Pitsoulis, The quadratic assignment problem, in: P.P. Pardalos, M.G.C. Resende (Eds.), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Dordrecht, 1998, pp. 238–241.
[10] V.D. Cung, T. Mautor, P. Michelon, A. Travares, A scatter search based approach for thz quadratic assignment problem, in: Proceedings of the IEEE International Conference on Evolutionary Computation and Evolutionary Programming (ICEC'97), 1997, pp. 165–170.
[11] Z. Drezner, A new genetic algorithm for the quadratic assignment problem, INFORMS Journal on Computing 15 (3) (2003) 320–330.
[12] Z. Drezner, Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem, Computers and Operations Research 35 (3) (2008) 717–736.
[13] E. Duman, I. Or, The quadratic assignment problem in the context of the printed circuit board assembly process, Computers and Operations Research 34 (1) (2007) 163–179.
[14] G. Erdogan, B. Tansel, A branch-and-cut algorithm for quadratic assignment problems based on linearizations, Computers and Operations Research 34 (4) (2007) 1085–1106.
[15] C. Fleurent, J.A. Ferland, Genetic hybrids for the quadratic assignment problem, DIMACS Series in Mathematics and Theoretical Computer Science (1993) 173–187.
[16] C. Régo, T. James, F. Glover, An ejection chain algorithm for the quadratic assignment problem, Networks 56 (3) (2010) 188–206.
[17] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.
[18] P. Hahn, T. Grant, N. Hall, A branch-and-bound algorithm for the quadratic assignment problem based on the Hungarian method, European Journal of Operational Research 108 (3) (1998) 629–640.
[19] T. James, C. Régo, F. Glover, A cooperative parallel tabu search algorithm for the quadratic assignment problem, European Journal of Operational Research 195 (3) (2009) 810–826.
[20] T. James, C. Régo, F. Glover, Multistart tabu search and diversification strategies for the quadratic assignment problem, IEEE Transactions on Systems Man and Cybernetics – Part A: Systems and Humans 39 (3) (2009) 579–596.
[21] S. Kirkpatrick, C.D. Gelett, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 621–630.
[22] E.M. Loiola, N.M. Maia de Abreu, P.O. Boaventura-Netto, P. Hahn, T. Querido, A survey for the quadratic assignment problem, European Journal of Operational Research 176 (2) (2007) 657–690.
[23] T. Jones, S. Forrest, Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, 1995, pp. 184–192.
[24] H.R. Lourenco, O. Martin, T. Stützle, Iterated local search, in: Handbook of Meta-heuristics, Springer-Verlag, Berlin, Heidelberg, 2003.
[25] F. Malucelli, Quadratic assignment problems: solution methods and applications. PhD Thesis, Departimento di Informatica, Universita di Pisa, Italy, 1993.
[26] P. Merz, B. Freisleben, Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, IEEE Transactions on Evolutionary Computation 4 (4) (2000) 337–352.

[27] P. Merz, B. Freisleben, Fitness landscapes memetic algorithms and greedy operators for graph bi-partitioning, Evolutionary Computation 8 (1) (2000) 61–91.
[28] A. Misevicius, An improved hybrid genetic algorithm: new results for the quadratic assignment problem, Knowledge Based Systems 17 (2–4) (2004) 65–73.
[29] A. Misevicius, A tabu search algorithm for the quadratic assignment problem, Computational Optimization and Applications 30 (1) (2005) 95–111.
[30] A. Misevicius, A. Lenkevicius, D. Rubliauskas, Iterated tabu search: an improvement to standard tabu search, Information Technology and Control 35 (3) (2006) 187–197.
[31] P.M. Pardalos, F. Rendl, H. Wolkowicz, The quadratic assignment problem: a survey and recent developments, in: Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, vol. 16 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1994, pp. 1–42.
[32] C.R. Reeves, Landscapes operators and heuristic search, Annals of Operations Research 86 (1997) 473–490.
[33] J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem, ORSA Journal of Computing 2 (1990) 33–45.
[34] T. Stützle, Iterated local search for the quadratic assignment problem, European Journal of Operational Research 174 (3) (2006) 1519–1539.
[35] T. Stützle, M. Dorigo, ACO algorithms for the quadratic assignment problem, In New Ideas for Optimization (1999) 33–50.
[36] E. Taillard, Robust taboo search for the quadratic assignment problem, Parallel Computing 17 (1991) 443–455.
[37] E.-G. Talbi, O. Roux, C. Fonlupt, D. Robillard, Parallel ant colonies for the quadratic assignment problem, Future Generation Computer Systems 17 (2001) 441–449.
[38] L.Y. Tseng, S.C. Liang, A hybrid Metaheuristic for the quadratic assignment problem, Computational Optimization and Applications 34 (2006) 85–113.
[39] M.R. Wilhelm, T.L. Ward, Solving quadratic assignment problems by simulated annealing, IIE Transactions 19 (1) (1987) 107–119.