Memetic search for the quadratic assignment problem

Una Benlic[a], Jin-Kao Hao[b,*]

[a] *CHORDS, University of Stirling, Stirling FK9 4LA, United Kingdom*
[b] *LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers, Cedex 01, France*

Accepted to Expert Systems with Applications, 12 August 2014

Abstract

The quadratic assignment problem (QAP) is one of the most studied NP-hard problems with various practical applications. In this work, we propose a powerful population-based memetic algorithm (called BMA) for QAP. BMA integrates an effective local optimization algorithm called Breakout Local Search (BLS) within the evolutionary computing framework which itself is based on a uniform crossover, a fitness-based pool updating strategy and an adaptive mutation procedure. Extensive computational studies on the set of 135 well-known benchmark instances from the QAPLIB revealed that the proposed algorithm is able to attain the best-known results for 133 instances and thus competes very favorably with the current most effective QAP approaches. A study of the search landscape and crossover operators is also proposed to shed light on the behavior of the algorithm.

Keywords: Memetic algorithm; local search; landscape analysis; quadratic assignment; combinatorial optimization.

1. Introduction

The quadratic assignment problem (QAP) is a classic NP-hard combinatorial optimization problem with a number of applications (Cheng, Ho, & Kwan, 2012; Garey & Johnson, 1979; Li, Xu, Jin, & Wang, 2012; Miao, Cai, & Xu, 2014; Nikolić & Teodorović, 2014; Pardalos, Rendl, & Wolkowicz, 1994). QAP is to determine a minimal cost assignment of n facilities to n locations, given a flow a_{ij} from facility i to facility j for all $i, j \in \{1, ..., n\}$ and a distance b_{qp} between locations q and p for all $q, p \in \{1, ..., n\}$. Let Π denote the set of the permutation functions $\pi : \{1, ..., n\} \rightarrow \{1, ..., n\}$, then QAP can mathematically be formulated as follows:

$$\min_{\pi \in \Pi} f(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi_i \pi_j} \tag{1}$$

*Corresponding author.
Email addresses: `ube@cs.stir.ac.uk` (Una Benlic), `hao@info.univ-angers.fr` (Jin-Kao Hao)

where a and b are the flow and distance matrices respectively, and $\pi \in \Pi$ is a solution where π_i represents the location chosen for 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., $f(\pi^*) \leq f(\pi)$, $\forall \pi \in \Pi$.

Besides the facility location problem, QAP is notable for its ability to formulate a number of other practical problems such as backboard wiring in electronics, design of typewriter keyboards, campus planning, analysis of chemical reactions for organic compounds, balancing turbine runners, and many others. QAP can equally formulate some classic combinatorial optimization problems such as the traveling salesman, maximum clique and graph partitioning problems. Reviews on some significant applications of QAP can be found in Burkard (1991); Duman & Or (2007) and Pardalos, Rendl, & Wolkowicz (1994), while many solution methods are reviewed in Anstreicher (2003).

QAP is among the most studied and the hardest combinatorial optimization problems. In fact, from a theoretical point of view, QAP is NP-hard (Garey & Johnson, 1979). Consequently, no exact algorithm is expected to solve the problem in a polynomial time and even small instances may require considerable computation time. This hardness is confirmed in practice since the existing exact algorithms can solve to optimality only small instances from the QAP benchmark library with up to 36 locations. Even approximation of the problem with a guaranteed performance is known to be very hard (Hassin, Levin, & Sviridenko, 2009). For these reasons, heuristic and metaheuristic methods constitute a natural and useful approach for tackling this problem (Blum, Puchinger, Raidl, & Roli, 2011). Such algorithms aim to provide satisfactory sub-optimal solutions in acceptable computing time, but with no theoretically provable guarantee that the attained solutions are the optimal ones. Performance of these heuristic algorithms is typically assessed using a set of benchmark instances. Among the numerous heuristic algorithms reported for QAP in the literature, local search methods are very popular approaches, including simulated annealing (Wilhelm & Ward, 1987), tabu search (Battiti & Tecchiolli, 1994; James, Rego, & Glover, 2009a,b; Misevicius & Kilda, 2006; Skorin-Kapov, 1990; Taillard, 1991), and iterated local search (Benlic & Hao, 2013c; Stützle, 2006). Population-based approaches constitute another class of popular tools for finding high quality near-optimal solutions for QAP (Ahuja, Orlin, & Tiwari, 2000; Drezner, 2003, 2008; Fleurent & Ferland, 1993; Merz & Freisleben, 2000; Misevicius, 2004; Stützle, 2006).

In this work, we are interested in solving QAP with heuristic algorithms. We introduce a powerful memetic algorithm (BMA) for QAP which combines an effective local search algorithm (BLS - Breakout Local Search), a crossover operator, a pool updating strategy, and an adaptive mutation mechanism. BLS is a general local search method which has shown very good results for several NP-hard problems including maximum clique (Benlic & Hao, 2013a), maximum cut (Benlic & Hao, 2013b), QAP (Benlic & Hao, 2013c) and vertex separator (Benlic & Hao, 2013d). Its basic idea is to use a descent procedure to discover local optima and employ dedicated perturbations to continually move from one attractor to another in the search space. In this paper, we integrate BLS into the memetic framework, thus extending our work of Benlic & Hao (2013c). As we show in this paper, the proposed memetic algorithm BMA exhibits an excellent performance on the whole set of 135 well-known QAP benchmark instances. Indeed, BMA attains the best-known solution for all the instances except for only two cases. Furthermore, it outperforms its local search procedure BLS which confirms the usefulness of the memetic framework. In order to gain some insight into the functioning of the proposed memetic algorithm, we perform a landscape analysis and justify the choice for the used crossover operator (see Section 5).

The paper is organized as follows. In the next section, we briefly review the most effective QAP approaches and highlight the contributions of this work. In Section 3, we present our proposed memetic algorithm (BMA) and detail its main components. Moreover, we highlight the differences and similarities between BMA and the reviewed state-of-art approaches. Computational results and comparisons with the top-performing QAP algorithms are presented in Section 4. Section 5 provides a landscape study that we use to justify the choices for the crossover operator of our memetic algorithm, and additionally shows a comparison of several crossover operators. Conclusions are provided in the last section.

2. State-of-art approaches for QAP and main contributions

In this section, we provide a literature review of the most popular heuristic approaches for QAP, including four population-based algorithms (Drezner, 2008; Merz & Freisleben, 2000; Misevicius, 2004; Stützle, 2006) and two local search algorithms (James, Rego, & Glover, 2009a; Misevicius & Kilda, 2006), followed by a summary of the main contributions of this study. In Section 3.5, we discuss in more detail the relationships between these approaches and the proposed BMA. Among the reviewed approaches, four algorithms from Drezner (2008); James, Rego, & Glover (2009a); Misevicius (2004) and Misevicius & Kilda (2006) report the best results on some particular class of QAP benchmark instances. For this reason, these four algorithms will be used as reference algorithms for our comparative study. Nevertheless, 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 structure of these instance (see Section 5).

A popular memetic approach for QAP (MA-QAP) is introduced in Merz & Freisleben (2000) which incorporates the 2-opt local search procedure and an adaptation of the standard uniform crossover UX that does not perform any implicit mutation. The selection for reproduction is performed on a purely random basis, while the selection for survival is achieved by choosing the best individuals from the pool of parents and children. To overcome premature convergence, the restart technique proposed in Eshelman (1991) is employed. During the run, it is checked whether the average distance of the population has dropped below a certain threshold or whether the average fitness of the population did not change after a certain number of consecutive generations. If one of these conditions holds, the whole population is mutated except the best individual, and each mutated individual is improved by the 2-opt local search to obtain a local optimum. Afterwards, the algorithm proceeds with the usual recombination process. We mention this work since it is one of the first memetic algorithms applied to QAP and achieved remarkable results at the time it was published.

The Improved Hybrid Genetic Algorithm (IHGA) (Misevicius, 2004) 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, based on the swap neighborhood, which contains 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 (SM), 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-performing algorithms for the unstructured instances and real-life like instances and is used as one of the references in our comparative study.

Drezner (2008) shows extensive computational experiments on QAP using various variants of a hybrid genetic algorithm. The author compared 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 were 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 the robust tabu search (RTS) (Taillard, 1991) except that the tabu tenure is generated in $[0.2n, 1.8n]$ rather than in $[0.9n, 1.1n]$. MRT60 is the best-performing memetic algorithm for the grid-based instances and is used as another reference algorithm in our comparative study.

Population-based Iterated Local Search (PILS) (Stützle, 2006) is another highly effective algorithm. The underlying iterated local search (ILS) algorithm starts from a random assignment, and applies a first-improvement local search procedure based on the 2-opt neighborhood. To speed up the search process, the algorithm uses the *don't 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 quality and distance between solutions, is then employed to form a new population of μ solutions from the set of $\mu + \lambda$ solutions.

Cooperative parallel tabu search algorithm (CPTS) (James, Rego, & Glover, 2009a) executes in parallel several tabu search (TS) operators on multiple processors. The TS operator is a modified version of Taillard's RTS (Taillard, 1991) 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 another reference algorithm in our comparative study.

Iterated tabu search (ITS) by Misevicius & Kilda (2006) follows the general scheme of the iterated local search metaheuristic. It 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 the 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. ITS obtains excellent results on the unstructured instances and real-life like instances and is used as another reference in our comparison.

Compared to the existing studies on QAP, this work has the following main contributions:

First, the proposed BMA algorithm is based on a new local optimization procedure (i.e., BLS) which adopts an adaptive perturbation mechanism to better escape local optima. The computational study discloses that this algorithm performs very well on the set of 135 very popular QAP benchmark instances by attaining the best-known result in 133 cases. This work thus confirms the usefulness of the memetic framework for QAP.

Second, we provide an empirical justification to explain why the uniform crossover is the best operator for QAP in comparison with the other crossover operators studied in the literature.

Third, ideas of the proposed algorithm could help in designing effective heuristics for other related permutation problems and applications such as those mentioned in the introductory section.

3. An effective memetic algorithm for QAP

The term *memetic algorithm* (MA) is employed to designate a general heuristic approach which typically combines local optimization with a population-based paradigm (Moscato, 1989; Moscato & Cotta, 2003). The purpose of such a combination is to take advantages of both crossover that discovers unexplored promising regions of the search space, and local optimization that finds good solutions by concentrating the search around these regions. Since memetic algorithm is a problem-independent framework, it needs to be properly adapted to the specific problem at hand to ensure the best performance (Hao, 2012; Krasnogor & Smith, 2005). In particular, local optimization operator and recombination operator are the two key components to consider. Finally, as for any population-based method, a healthy diversity of population must be maintained to avoid premature convergence. Previous studies show that memetic algorithms are able to achieve excellent performances for a number of optimization problems (Hart, Krasnogor, & Smith, 2004; Moscato & Cotta, 2003; Neri, Cotta, & Moscato, 2012).

Given an initial population which consists of locally optimal solutions, a memetic approach generates new solutions by applying crossover and/or mutation, followed by a phase of local search to improve each offspring solution. The right choice for a crossover operator depends on problem structure and landscape properties (in Section 5, we show a study on the relationship between the performance of a crossover operator and the structural properties of a given problem). Moreover, the success of a memetic approach is conditioned by the effectiveness of the local search procedure. While the main role of the crossover is to discover unexplored promising regions of the search space (i.e., exploration), local search basically aims to find good solutions by concentrating the search around these regions (i.e., exploitation).

The proposed memetic algorithm for QAP (BMA) employs the uniform crossover operator (Section 3.1) (In Section 5.2, we justify why we chose this particular crossover), a Breakout Local Search (BLS) procedure (Section 3.2), a fitness-based population replacement strategy (Section 3.3), and an adaptive mutation mechanism (Section 3.4). Each offspring solution, generated with the uniform crossover, is improved with the BLS procedure. Our memetic approach then applies a pool updating strategy to possibly replace the worst individual from the population with the improved offspring solution. To avoid premature convergence, BMA triggers an adaptive mutation mechanism to the entire population if the best solution found during the search has not been improved during a fixed number of generations. This displaces the search to distant regions each time a search stagnation is detected.

The general architecture of our memetic approach is described in Algorithm 1. The main components are detailed in the following sections.

3.1. Parents selection and recombination

To determine a subset $p \subset P$ of $|p|$ parent individuals, we employ the tournament selection strategy. Let λ be the size of the tournament pool. We select each individual $\pi^i \in p$ in the following way: randomly choose λ individuals from P; among the λ chosen individuals, place the best one into p if it is not already present in p. The time complexity of this operation is $\mathcal{O}(|P|)$. An advantage of the tournament selection is that the selection pressure can easily be adjusted by changing the size of the tournament pool λ. The larger the tournament pool is the less is the chance to select weaker individuals. An experimental evaluation of our QAP approach has revealed that putting a higher pressure on better individuals gives better results than using a random selection.

For the crossover process, we employ a standard uniform operator (UX) that recombines two parent individuals from subset p. Elements of the parents are scanned from left to right and each element in the offspring keeps with equal probability the value $j \in [1...n]$ found in either of the two parents, under the

Algorithm 1 General scheme of the proposed BMA algorithm

```
1: Initialize the number of BLS iterations for short and long runs $t_s$ and $t_l$ respectively,
   the minimum mutation degree $\mu_{min}$ and the increment $m$ of mutation degree
2: Randomly generate initial population $P$
3: $P \leftarrow BLS(P, t_s)$   /* Improve each individual with $t_s$ iterations of BLS, Sect. 3.2 */
4: $\pi^{best} \leftarrow BestIndividual(P)$ /* Initialize the best individual */
5: $\mu \leftarrow \mu_{min}$   /* Initialize the current mutation degree */
6: for $i := 0$ to number of generations $\phi$ do
7:    Select a subset of parent individuals $p$ from $P$ /* Sect. 3.1
8:    $\pi^0 \leftarrow Crossover(p)$ /* Generate an offspring, Sect. 3.1 */
9:    $\pi^0 \leftarrow BLS(\pi^0, t_l)$   /* Improve offspring $\pi^0$ with long BLS run, Sect. 3.2 */
10:   $P \leftarrow ReplacementStrategy(P, \pi^0)$ /* Sect. 3.3 */
11:   if ($\pi^{best}$ not improved after $\nu$ generations) then
12:      $P \leftarrow Mutate(P, \mu)$        /* Sect. 3.4 */
13:      $P \leftarrow BLS(P, t_s)$
14:      Update the best solution $\pi^{best}$ if necessary
15:      $\mu \leftarrow \mu + m$       /* Increase mutation degree */
16:   end if
17:   if ($f(\pi^{best}) > f(\pi^0)$ or $\mu > n$) then
18:      $\mu \leftarrow \mu_{min}$     /* Reset mutation degree to default */
19:   end if
20:   if ($f(\pi^{best}) > f(\pi^0)$) then
21:      $\pi^{best} \leftarrow \pi^0$
22:   end if
23: end for
```

constraint that j has not been assigned before to any element in the offspring. Any unassigned element is given a random value j such that j does not appear twice in the offspring. An example of this recombination process is illustrated in Figure 1. The complexity of this crossover is $\mathcal{O}(n)$.

Since the uniform crossover permits great flexibility, different variations of its basic procedure have been proposed and applied to QAP (Merz & Freisleben, 2000; Misevicius, 2004). Despite its simplicity, the UX has shown to provide very good results on different combinatorial optimization problems including QAP. A comparison of the simple UX with several other crossover operators used for QAP is reported in Section 5.2.

3.2. Breakout local search (BLS)

The local optimization procedure has a significant impact to the overall performance of a memetic algorithm. In our case, we adopt an existing local search algorithm BLS-QAP (Benlic & Hao, 2013c) (call it BLS for simplicity).

As most local search algorithms for QAP, BLS employs the swap operator which consists in exchanging the values from two different positions in π, i.e., permuting the locations of two facilities. It uses the best improvement descent procedure to exploit the whole swap neighborhood $N(\pi)$, which is evaluated in

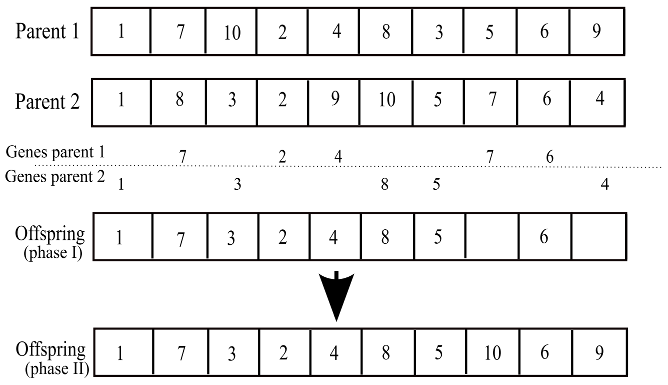

Figure 1: An example of the uniform crossover (UX)

$\mathcal{O}(n^2)$ time thanks to an effective neighborhood evaluation strategy proposed in Taillard (1991).

Once a local optimum is returned by the best improvement descent procedure, BLS triggers a multi-type diversification mechanism which adaptively determines the type T of perturbation moves and the number L of perturbations (called jump magnitude) by considering some information related to the search state.

This mechanism combines two complementary types of perturbation: a guided perturbation (using a tabu list) and a random perturbation. The tabu-based perturbation 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 (where γ is the tabu tenure that takes a random value from a given range), while the random perturbation performs moves selected uniformly at random. In order to insure a good balance between an intensified and a diversified search, BLS-QAP alternates probabilistically between the two types of perturbations. The probability of applying a particular perturbation is determined dynamically depending on the current number ω of consecutive non-improving attractors visited (see Benlic & Hao (2013c) for more details). We limit the probability of applying the tabu-based perturbation over the random perturbation to take values not less than Q.

To determine the jump magnitude L for the following perturbation phase, the proposed algorithm uses a basic strategy which increments L if the local search procedure returned to the immediate previous local optimum, and otherwise resets L to its initial value L_0.

Once the type T and the number L of perturbations are determined, we apply accordingly L moves of type T to the current solution. The resulting solution is used by the next round of the best improvement descent procedure as its new starting point.

Once an offspring solution is created in the crossover phase (line 8 of Alg. 1), it is improved with t_l iterations of the BLS procedure (line 9 of Alg. 1).

As previously mentioned, the complexity of each iteration of the BLS descent procedure is $\mathcal{O}(n^2)$. The number of the descent iterations performed in each iteration of BLS depends on the size s_{basin} of the basin of attraction of a local optimum. This, in turn, depends on the type of perturbation used for the previous perturbation phase. More precisely, after a phase of the directed (tabu-based) perturbation, the descent-based local search requires, on average, less steps to attain a local optimum than after the random perturbation. However, the computational complexity of one iteration of the tabu-based perturbation is $\mathcal{O}(n^2)$, while the complexity of the random perturbation is $\mathcal{O}(1)$. Therefore, the time complexity of one BLS iteration is bounded within $\mathcal{O}(s_{basin} \cdot n^2) + \mathcal{O}(L \cdot n^2)$.

3.3. Pool updating strategy

For each offspring solution π^0 created by the crossover operator and improved with the BLS procedure (Section 3.2), we decide whether π^0 should be inserted into the population pool (lines 8-10 of Alg. 1). To base this decision, our algorithm uses a classic replacement strategy which inserts π^0 into P if there is no individual in P identical to π^0 (i.e., $\forall \pi^i \in P, \pi^i \neq \pi^0$), and if the fitness of π^0 is better than the fitness of the worst individual π^{worst} from P (i.e., $f(\pi^0) < f(\pi^{worst})$). If this condition holds, P is updated by replacing π^{worst} with π^0. To determine whether an identical solution to π^0 is already present in P, we compute the hamming distance between π^0 and each $\pi^i \in P$. The hamming distance is defined as the number of positions at which the corresponding elements are different. If two solutions are identical, the corresponding hamming distance is 0. The overall time complexity of this pool update strategy is thus $\mathcal{O}(|P| \cdot n)$.

One potential risk of this replacement strategy is the premature loss of population diversity, since offspring is being inserted into the population regardless of its distance to other individuals in the population. To avoid this problem, more sophisticated pool updating strategies have been proposed in the literature that maintain population diversity by considering the similarity (distance) between the offspring and other individuals from the population (see for instance Benlic & Hao (2011); Lü & Hao (2010); Porumbel, Hao, & Kuntz (2010)). In our case, the diversity is maintained by an adaptive mutation mechanism described in Section 3.4. As a result, the proposed memetic approach is not really sensitive to the pool replacement strategy employed.

3.4. Adaptive mutation procedure

As soon as a search stagnation is detected, i.e., the best solution π^{best} found during the search has not been updated for a certain number of generations, our BMA algorithm mutates the entire population with an adaptive mutation procedure that adjusts the diversification strength $\mu \in [\mu_{min}, n]$ depending on the previous search progress (see lines 11-16, Algorithm 1). At the beginning, μ is set to μ_{min} and is gradually augmented by an increment m if a solution better than π^{best} has not been attained during ν generations. Once a solution better than π^{best} is obtained, or μ has reached the maximum possible value n (n being the problem size), μ is reset to μ_{min} (see lines 17-19, Algorithm 1).

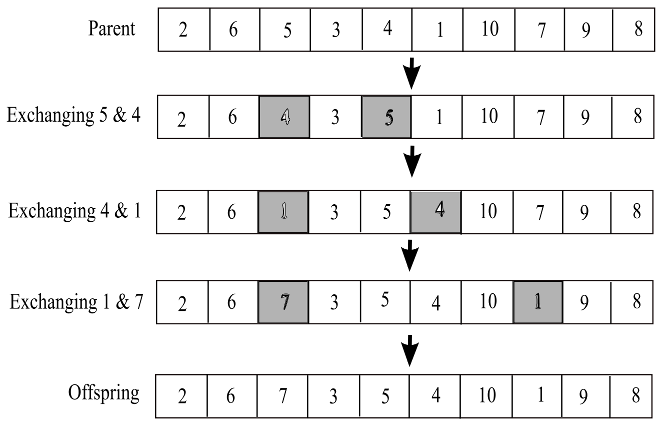

Figure 2: An example of mutation with 3 exchanges for $\mu = 3$

The mutation procedure of our memetic algorithm, which was previously used in Merz & Freisleben (2000) for QAP, exchanges a sequence of locations in the solution π to create an offspring that has a distance of μ to its parent π. The distance measure used for this mutation operator is the well-known hamming distance. To ensure that the distance between the offspring and its parent is μ, in each step, the second selected location is swapped again in the subsequent step, such that the resulting distance is equal to one plus the number of swaps. An example of the mutation procedure is illustrated in Figure 2. The complexity of the mutation procedure is $\mathcal{O}(|P| \cdot \mu)$.

3.5. Discussions

In this Section, we discuss similarities and differences between BMA and the previously mentioned state-of-art QAP approaches, in particular those based on the memetic framework.

In Section 2, we reviewed three popular memetic QAP algorithms, MA-QAP (Merz & Freisleben, 2000), IHGA (Misevicius, 2004), and MRT60 (Drezner, 2008), the two latter ones being among the best performing algorithms currently available in the literature for this problem. The genetic operators employed by BMA are quite similar to those of MA-QAP and IHGA. Indeed, these approaches create offspring solution by means of different adaptations of the classic uniform crossover operator which provides surprisingly good results for the QAP instances. One of the contributions of our work is an explanation for the efficiency of these random crossovers obtained by analyzing the structures of the QAP instances (see Section 5).

Moreover, as MA-QAP and IHGA, BMA applies a mutation mechanism to the entire population in order to avoid premature convergence. However, unlike the previously used mutation strategies, the mutation mechanism of BMA adaptively adjusts the diversification strength depending on the previous search progress.

Yet, the most important difference between BMA and the three reference memetic algorithms is the local search procedure. As stated earlier, BLS, employed by BMA for offspring improvement, constitutes the key component of BMA. It combines the steepest descent with a dedicated and adaptive diversification mechanism. On the other hand, the local search phase performed by MA-QAP is the basic steepest descent algorithm, while IHGA and MRT60 improve solutions by means of a simple tabu search procedure.

PILS (Stützle, 2006) is a population-based approach which uses an adaptation of the iterated local search framework to improve each newly generated solution. Although the perturbation mechanism of PILS determines the number of perturbation moves in an adaptive way, it is only based on random moves which constitutes the main difference with our BLS procedure.

Two local search algorithms, CPTS (James, Rego, & Glover, 2009a) and ITS (Misevicius & Kilda, 2006) (see Section 2), are among the top performing QAP methods. Even though BLS's directed perturbation is based on the use of a tabu list, CPTS is not very much related to BLS since it consists of a parallel execution of several tabu search (TS) operators on multiple processors. The proposed BLS is more related to ITS since both algorithms are variants of the iterated local search method. Nevertheless, there are two major differences between ITS and BLS. Firstly, BLS does not consider the tabu list during its local search (descent) phases, unlike ITS which constraints each iteration of its local search phase with a tabu list. As such, BLS and ITS explore different trajectories during their respective search, leading to different local optima. In fact, one of the keys to the effectiveness of BLS on QAP is that it completely excludes diversification during local search, unlike tabu search for which the intensification and diversification are always intertwined. Secondly, unlike ITS which applies solely random moves to perturb the current local optimum, BLS adaptively choses between two types of moves (random and directed) according to the search status, leading to variable levels of diversification.

Finally, we mention the approach proposed by Kelly, Laguna, & Glover (1994). In their work, the authors present a method which also starts with the steepest descent to find local optima, and then applies a diversification strategy to incrementally restrict the set of allowed swap moves in order to escape local optima. The diversification procedure of this approach exploits the search history as well, but unlike BLS, performs perturbations in a purely deterministic way. It consists of imposing a "maximum tabu tenure" to moves performed during each descent of local search, and applying the best move among the non prohibited ones to perturb a local optimum.

4. Experimental evaluation

4.1. Benchmark instances

It is worthy to recall that given the very nature of heuristic algorithms, it is a common practice to evaluate the performance of a new algorithm by testing it on well-established benchmark instances of the problem and comparing its results

Table 1: Settings of important parameters.

Para.	*Description*	*Value*
$\lvert P\rvert$	Population size	15
t_s	number of iterations for short BLS run	5000
t_l	number of iterations for long BLS run	10000
μ_{min}	minimal mutation degree	$0.5n$
m	increment of mutation degree	$0.1n$
λ	the size of the tournament pool	4
ν	number of generations without improvement before mutation	$\lvert P\rvert$
L_0	initial jump magnitude of BLS	$0.05n$ (T. I & II), $0.15n$ (T. III & IV)
γ	tabu tenure for directed perturb. with BLS	$random[0.9n, 1.1n]$
Q	smallest probability for applying BLS directed perturbation	0.75

with those of the state-of-art methods. In our case, we evaluate the performance of our BMA algorithm on the set of 135 instances from the QAPLIB[1], whose size n varies from 12 to 150 and is indicated in the instance name. These instances are very popular and largely used in the literature. They are typically classified into four types covering various real applications and random problems:

Type I. Real-life instances obtained from practical applications of QAP;

Type II. Unstructured, randomly generated instances for which the distance and flow matrices are randomly generated based on a uniform distribution;

Type III. Randomly generated instances with structure that is similar to that of real-life instances;

Type IV. Instances in which distances are based on the Manhattan distance on a grid.

Among the 135 instances from the QAPLIB, we focus on the set of 21 selected instances. The remaining 114 instances (including all the real-life instance of Type I) are omitted from our experimental comparisons since BLS and BMA (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). However, to be exhaustive, we include the results of our BMA algorithm for these 114 instances in the Appendix (Table 8).

4.2. Experimental protocol

The proposed memetic algorithm BMA[2] is programmed in C++, and compiled with GNU g++ on a Xeon E5440 with 2.83GHz and 2GB. Like other QAP heuristic algorithms, BMA requires a number of parameters to be tuned (see

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

[2]The code of our memetic algorithm, used to obtained the reported results, will be made available online at http://www.info.univ-angers.fr/pub/hao/BMA.html

Table 1), most of which are related to its BLS local optimizer and adopt the values used in Benlic & Hao (2013c). According to the parameter analyses performed in Benlic & Hao (2013b,c), the most relevant BLS parameters are the initial jump magnitude L_0 and the smallest probability for applying directed over random perturbation Q. Moreover, as discussed in Benlic & Hao (2013c), the optimal setting of these parameters depends on the landscape properties of the QAP instance at hand and may vary for different instances. Additionally, following the existing MA literature on discrete optimization (Benlic & Hao, 2011; Bontouxa, Artigues, & Feillet, 2010; Dorne & Hao, 1998; Hao, 2012; Lü & Hao, 2010; Merz & Freisleben, 2000), BMA maintains a population of fairly limited size. This, as well as the other parameters related to the genetic algorithm components, is determined with a preliminary experiment. Even though it is possible to find a configuration of parameters better than the one used in this paper, the computational experiments reported in this section show that the adopted setting performs globally well on the tested benchmark instances. More generally, since the source code of our BMA algorithm will be made available online, the potential user can possibly apply any preferred method to tune these parameters.

According to the practice in the QAP literature, the stopping condition is the elapsed time. In our case, we set the maximum time limit to 2 hours for all the instances, except for the largest instance $tho150$ for which we set the maximum time allowed to 10 hours. Notice that some reference QAP algorithms (Drezner, 2008; James, Rego, & Glover, 2009a) use an even higher time limit for $tho150$ to reach the reported result. As shown below, the best-known solutions are very often attained long before these time limits. Furthermore, we provide computational results of our BMA algorithm with the time limit reduced to 30 minutes. The reported results for BMA and BLS are obtained over 10 independent executions.

We focus primarily on the comparisons in terms of the solution quality with respect to the best-known results (BKR) reported in the literature, which were obtained with different QAP algorithms under various conditions. For indicative purposes, we also compare the results produced by our memetic approach and those obtained with the 4 state-of-art QAP approaches which are the current best performing methods:

1. Cooperative Parallel Tabu Search (CPTS) algorithm by James, Rego, & Glover (2009a). The reported results are obtained using ten (1.3GHz) Intel Intanium processors;

2. Iterated Tabu Search (ITS) by Misevicius & Kilda (2006). The results are obtained using a 900 MHz Pentium computer;

3. Improved Hybrid Genetic Algorithm (IHGA) by Misevicius (2004). The reported results are obtained on a x86 Family 6 processor;

4. A variant of a Hybrid Genetic Tabu Search Algorithm (MRT60) by Drezner (2008). The reported results are obtained on a Pentium IV 2.8 GHz computer.

An exhaustive comparative analysis with the reference approaches is not a straightforward task because of the differences in computing hardware, result reporting methodology, termination criterion, etc. This comparison is thus presented only for indicative purposes. Nevertheless, this experiment provides interesting indications on the performance of the proposed algorithm relative to the state-of-art algorithms.

In addition, we evaluate the contribution of the proposed memetic approach by comparing (under the same conditions) BMA with its BLS procedure (Benlic & Hao, 2013c) which itself shows to be highly effective on the tested QAP instances. This allows us to highlight the usefulness of the memetic framework.

According to the literature (Drezner, 2008; Misevicius, 2004; Stützle, 2006), we employ several criteria for evaluation and comparison of our memetic algorithm with other QAP approaches.

1. The number of instances for which an optimal or best-known solution is reached within a reasonable computing time. This constitutes an indicator on the effectiveness of an algorithm in terms of the solution quality.

2. The success rate of reaching an optimal or best-known solution which provides information about the robustness of an algorithm.

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 average result over a given number of runs and $\bar{z}$ the best-known objective value. This indicator provides additional information about the robustness of an algorithm.

Beside the aforementioned criteria, we also mention the amount of computational time required by each approach to reach the reported results. This provides an indication about the computational efficiency of an algorithm. Moreover, Table 7 in the Appendix provides a detailed summary of the computational results for 21 selected QAPLIB instances obtained by BMA within a time limit of 2 hours (10 hours for instance *tho*150) and 30 minutes respectively.

4.3. Computational results and comparisons

Table 2 reports comparative results with BLS (Section 3.2), CPTS (James, Rego, & Glover, 2009a), ITS (Misevicius & Kilda, 2006) and IHGA (Misevicius, 2004), for unstructured instances (Type II) and real-life like instances (Type III). Since MRT60 (Drezner, 2008) does not report results on these instances, it is excluded from this comparison (in fact, MRT60 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 between an average solution, obtained with the given approach over 10 independent trials, and the best-known solution. The success rate for reaching the best-known solution over 10 trials is given in

Table 2: Comparative results between the proposed memetic algorithm (BMA), BLS (Benlic & Hao, 2013c), and three best performing QAP approaches on unstructured instances (Type II) and real-life like instances (Type III): CPTS (James, Rego, & Glover, 2009a), ITS (Misevicius & Kilda, 2006) and IHGA (Misevicius, 2004). The success rate of reaching the best-known result over 10 executions is indicated between parentheses. Computing times are given in minutes for indicative purposes.

Problem	BKS	BMA		BLS		CPTS		ITS		IHGA	
		% $\bar{\delta}_{avg}$	t(m)	% $\bar{\delta}_{avg}$	t(m)	% $\bar{\delta}_{avg}$	t(m)	% $\bar{\delta}_{avg}$	t(m)	% $\bar{\delta}_{avg}$	t(m)
Random instances (Type II)											
tai40a	3139370	0.059(2)	8.1	**0.022**(7)	38.9	0.148(1)	3.5	0.210(1)	0.8	0.209(1)	1.4
tai50a	4938796	**0.131**(2)	42.0	0.157(2)	45.1	0.440(0)	10.3	0.373(0)	3.0	0.262(0)	5.0
tai60a	7205962	**0.144**(2)	67.5	0.251(1)	47.9	0.476(0)	26.4	0.330(1)	9.7	0.583(0)	12
tai80a	13499184	**0.426**(0)	65.8	0.517(0)	47.3	0.691(0)	94.8	0.494(0)	25.0	0.756(0)	53.3
tai100a	21052466	**0.405**(0)	44.1	0.430(0)	39.0	0.589(0)	261.2	0.427(0)	60.0	0.606(0)	200.0
Average		0.233	45.5	0.275	43.6	0.469	79.2	0.367	19.2	0.483	54.3
Real-life like instances (Type III)											
tai50b	458821517	**0.000**(10)	1.2	**0.000**(10)	2.8	**0.000**(10)	13.8	**0.000**(10)	0.9	**0.000**(10)	0.3
tai60b	608215054	**0.000**(10)	5.2	**0.000**(10)	5.6	**0.000**(10)	30.4	**0.000**(10)	2.2	**0.000**(10)	0.7
tai80b	818415043	**0.000**(10)	31.3	**0.000**(10)	11.4	**0.000**(10)	110.9	**0.000**(10)	5.8	**0.000**(10)	2.5
tai100b	1185996137	0.000(10)	13.6	**0.000**(10)	16.0	0.001(8)	241.0	**0.000**(9)	23.3	**0.000**(10)	7.3
tai150b	498896643	0.060(1)	78.1	**0.100**(0)	80.5	0.076(0)	7377.8	0.100(1)	60.0	0.111(2)	38.3
Average		0.012	25.9	0.020	23.3	0.015	1554.8	0.020	18.4	0.022	9.8

Table 3: Comparative results between the proposed memetic algorithm (BMA), BLS (Benlic & Hao, 2013c), CPTS (James, Rego, & Glover, 2009a) and MRT60 (Drezner, 2008) on grid-based (Type IV) instances. The success rate of reaching the best-known result over 10 executions is indicated between parentheses. Computing times are given in minutes for indicative purposes.

Problem	BKS	BMA		BLS		CPTS		MRT60	
		% $\bar{\delta}_{avg}$	t(m)	% $\bar{\delta}_{avg}$	t(m)	% $\bar{\delta}_{avg}$	t(m)	% $\bar{\delta}_{avg}$	t(m)
sko72	48498	**0.000**(10)	3.5	**0.000**(10)	4.1	**0.000**(10)	69.6	**0.000**(10)	19.9
sko81	66256	**0.000**(10)	4.3	**0.000**(10)	13.9	**0.000**(10)	121.4	**0.000**(10)	31.9
sko90	90998	**0.000**(10)	15.3	**0.000**(10)	16.6	**0.000**(10)	193.7	**0.000**(10)	48.5
sko100a	115534	**0.000**(10)	22.3	0.001(9)	20.8	**0.000**(10)	304.8	**0.000**(10)	73.6
sko100b	152002	**0.000**(10)	6.5	**0.000**(10)	10.8	**0.000**(10)	309.6	**0.000**(10)	73.6
sko100c	147862	**0.000**(10)	12.0	**0.000**(10)	15.5	**0.000**(10)	316.1	**0.000**(10)	73.6
sko100d	149576	0.006(9)	20.9	0.001(5)	38.9	**0.000**(10)	309.8	**0.000**(10)	73.6
sko100e	149150	**0.000**(10)	11.9	**0.000**(10)	42.5	**0.000**(10)	309.1	**0.000**(10)	73.6
sko100f	149036	**0.000**(10)	23.0	**0.000**(10)	17.3	0.003(4)	310.3	0.000(9)	43.5
wil100	273038	**0.000**(10)	14.5	**0.000**(10)	18.9	**0.000**(10)	316.6	**0.000**(10)	73.6
tho150	8133398	0.008(3)	416.4	0.023(1)	268.8	0.013(0)	1991.7	**0.003**(3)	1223.6
Average		0.001	50.1	0.002	42.6	0.001	413.9	0.000	164.5

parentheses next to the value of $\bar{\delta}_{avg}$. The CPU time (in minutes) is only given for indicative purposes. From Table 2, we can make the following observations.

For the unstructured instances (Type II), BMA finds the best-known solution for 3 out of the 5 instances, with an average deviation $\bar{\delta}_{avg}$ of 0.233 over the 5 instances. For three hard instances *tai*40*a*, *tai*50*a* and *tai*60*a*, it reaches the best-known solution in 20% of the trials. Compared to its local search procedure BLS, BMA statistically outperforms (with p-value< 0.05) BLS for two hard instances, *tai*60*a* and *tai*80*a*, and reduces the average percentage deviation from 0.275 to 0.233 over the 5 instances. The current most effective approach for instances of Type II is probably ITS (Misevicius & Kilda, 2006), which attains the best-known result for 2 out of the 5 instances with an average deviation $\bar{\delta}_{avg}$ of 0.367. One notices that ITS shows worse results than both BMA and BLS, but it requires shorter computing time. To verify the performance of BMA under short computing budgets, we show in Appendix (Table 7) the results of BMA within a significantly shorter cutoff limit (30 minutes). From Table 7, we observe that BMA does remain very competitive with ITS, with an average deviation $\bar{\delta}_{avg}$ of 0.356 over the 5 instances. Finally, the two other reference algorithms CPTS and IHGA achieve results which are slightly worse than BMA, BLS and ITS.

For the real-life like instances (Type III), BMA 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%. It reports an average deviation $\bar{\delta}_{avg}$ of 0.012 over the 5 instances. Unlike BMA, BLS is unable to reach the best-known solution for *tai*150*b* with the given computing conditions. When the running time of BMA is greatly reduced, it is still able to attain the best-known solution for all the five instances but with an average deviation $\bar{\delta}_{avg}$ of 0.051 (see Table 7). Two reference approaches, ITS (Misevicius & Kilda, 2006) and IHGA (Misevicius, 2004), are also able to attain the best-known result for all the 5 real-life like instances with an average deviation $\bar{\delta}_{avg}$ of 0.020 and 0.022 respectively.

We now turn our attention to the instances with grid distances (Type IV, Table 3). Among the 4 reference methods, only CPTS (James, Rego, & Glover, 2009a) and MRT60 (Drezner, 2008) report results for these instances. In Table 3, we show the results of our BMA and BLS algorithms together with those of CPTS and MRT60 for these Type IV instances. From Table 3, we observe that BMA is able to reach the best-known result for all the instances of this type. For 9 out of 11 instances, it has a success rate of 100%. For the two remaining instances (*sko*100*d* and *tho*150), the success rate is 90% and 30% respectively. BLS also attains the best-known result for the hardest instance *tho*150 with a success rate of 10%. MRT60 has a slightly better success rate than BMA on instance *sko*100*d* (10/10 v.s. 9/10) and reports a slightly worse success rate than BMA on *sko*100*f* (9/10 v.s. 10/10). On the other instances, BMA and MRT60 have the same success rates with a very slight advantage for MRT60 in terms of the average gap over all the instances. On the other hand, CPTS performs globally well except for the hardest instance *tho*150 for which CPTS fails to attain the best-known solution. We conclude that BMA competes favorably with MRT60 and CPTS on the Type IV instances.

5. Analysis and discussion

The performance of a memetic algorithm may be influenced by the characteristics of the search landscape like the average distance between local optima and the relative distance of local optima to the nearest global optimum. Analyses of correlation between solution fitness and distance to global optimum have shown to be particularly useful for a better understanding of algorithm behavior and for designing suitable operators for a more effective search. Landscape studies have been previously reported for QAP in Merz & Freisleben (2000) and Stützle (2006) and for other well-known problems like the TSP problem (Boese, 1995) and the flow-shop scheduling problem (Reeves, 1997). In this section, we report a similar analysis, based on solutions sampled by our BLS procedure which are probably quite different from the solutions used in the previous studies on QAP.

We perform the landscape analysis on 16 QAPLIB instances, based on a set of distinct solutions obtained after 2000 independent runs of our BLS approach. The number of iterations per run is set to 200000. As the distance between solutions, we calculate the number of facilities that are allocated to distinct locations in two solutions π and π', i.e., $d(\pi, \pi') = |\{i|\pi_i \neq \pi'_i\}|$. Since global optima for the analyzed instances are not known, we use instead the best-known local optima to compute fitness-distance correlation and refer to them as global optima.

5.1. Landscape analysis - FDC and distribution of local optima

The fitness distance correlation (FDC) coefficient ρ (Jones & Forrest, 1995) captures the correlation between the solution fitness and its distance to the nearest global optimum (or best-known solution if global optimum is not available). For a minimization problem, if the fitness of a solution decreases with the decrease of distance from the optimum, then it should be easy to reach the target optimum for an algorithm that concentrates around the best candidate solutions found so far, since there is a "path" to the optimum via solutions with decreasing (better) fitness. A value of $\rho = 1$ indicates perfect correlation between fitness and distance to the optimum. For correlation of $\rho = -1$, the fitness function is completely misleading. FDC can also be visualized with a FD plot, where the same data used for estimating ρ is displayed graphically.

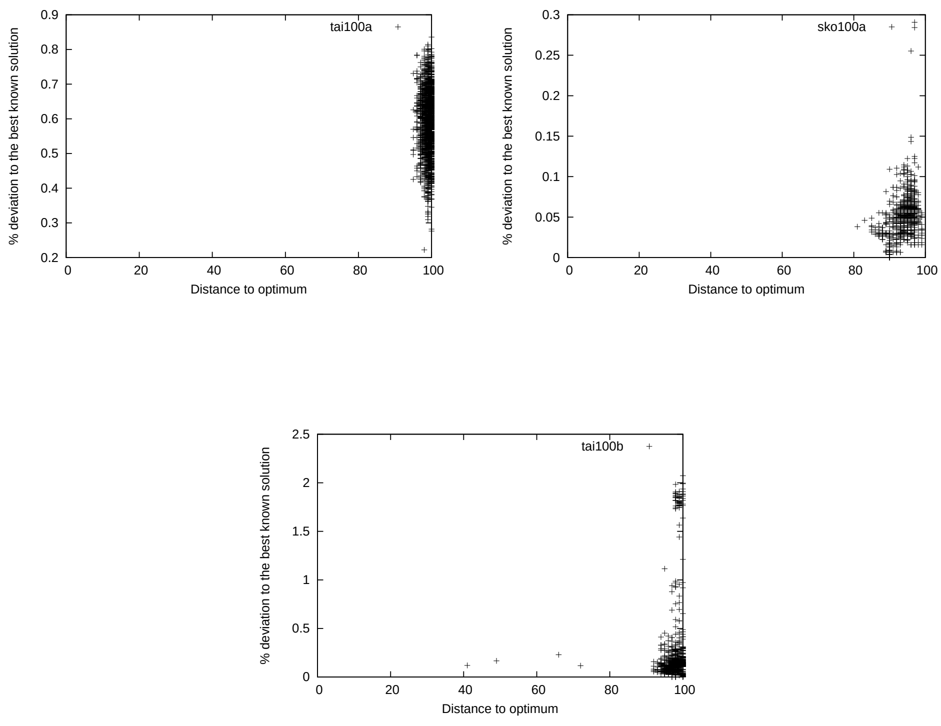

Figure 3: Distances of local optima to the best-known solution based on solutions sampled by the BLS algorithm for instances $tai100a$ (Type II), $sko100a$ (Type III) and $tai100b$ (Type IV).

In column 'ρ' of Table 4, we report FDC coefficients for the 16 selected QAP instances. For illustrative purpose, FD plots for three instances ($tai100a$, $sko100a$ and $tai100b$) are given in Figure 3. 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 randomly generated instances (Type II), the FDC coefficient ρ is negative except in one case ($tai50a$) where ρ is close to zero. Indeed, from the FD plots in Figure 3 it is clear that there is no correlation between fitness

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

and distance for the random instance $tai100a$.

For grid-based instances (Type IV), significant FDC exists except in one case ($sko81$, $\rho < 0$), while for two real-life like instances $tai80b$ and $tai100b$ the FDC is also low $\rho < 0.15$ but higher than for random instances. The FD plots in Figure 3 for instances $sko100a$ and $tai100b$ also 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 in the search space. Even high quality local optima are distant from each other and share no common structure. These observations will allow us to understand why the uniform crossover operator is appropriate for QAP.

5.2. Comparison of recombination operators

The objective of this section is to justify, based on the landscape analysis provided in Section 5.1, the choices for the crossover operator used by our BMA algorithm. We compare three versions of our BMA algorithm incorporating the three recombination operators of the literature for permutation problems: the standard uniform crossover (UX) described in Section 3.1, the block crossover (BX) and the distance preserving crossover (DPX).

The block crossover (BX), also called the multi-point crossover, is one of the classic recombination operators. Firstly, it chooses randomly a certain number of crossing points. Once the points have been chosen, starting from left to right, all the elements of the first parent are simply copied over to the offspring π^0 until the first crossing point is reached. Then all the elements of the second parent are copied over to π^0 until the second crossing point is reached. This procedure continues until reaching the last position in π^0. During the crossover process, we constrain an element to take on a value $j \in [1...n]$ that has not already been assigned to some element in π^0. The rest of unassigned elements are randomly assigned a value $j \in [1...n]$ such that solution feasibility is maintained.

The distance preserving crossover (DPX) has previously been applied to QAP in Merz & Freisleben (2000). The basic idea of DPX is to create an offspring that has the same distance to each of its parents, and that distance is equal to the distance between the parents themselves. Elements with the same values in both parents are copied to the offspring. The values of all the other elements change.

Generally, we can classify recombination operators into two classes, those that try to exploit an existing structure in a problem (e.g., the DPX crossover) and those that perform recombination in a purely random way (e.g., UX and BX crossovers). The former operators usually perform well on the class of problems with exploitable global structure and landscapes with highly correlated local optima (e.g., the traveling salesman problem (Bontouxa, Artigues, & Feillet, 2010) and the graph partitioning problem (Benlic & Hao, 2011)). Otherwise, these operators become destructive introducing a too strong perturbation.

Table 5 shows the average percentage deviation from the best-known result $\bar{\delta}_{avg}$ for the three versions of our BMA over 10 runs on 16 QAPLIB instances. The stopping condition used is the number of generations $\phi = 1,500$ with $10,000$ BLS iterations per generation. We indicate in parentheses the number of times each version of BMA reached the best-known result over the 10 executions. Moreover, we study the degree of perturbation p_{str} induced by each crossover, and report in Table 6 the average p_{str} caused by the three operators over the 1,500 generations. Here, we define the perturbation degree p_{str} as the minimum distance between the created offspring and one of its parents expressed as a percentage of n.

From Table 5, we observe that the best performance is obtained by the BMA version integrating the standard UX operator, with an average $\bar{\delta}_{avg}$ of 0.073 over the 16 instances. The second best performing BMA version integrates the block crossover (BX) with an average $\bar{\delta}_{avg}$ of 0.079. On the other hand, the results indicate that DPX is less effective with an average $\bar{\delta}_{avg}$ of 0.100 over the 16 instances. We further observe that the difference in performance between operators UX & BX and DPX is particularly notable on unstructured instances (Type II) since local optima are the most uncorrelated for these instances. Indeed, on the other instances with slightly higher FDC coefficient ρ, this difference is much less obvious.

As expected, we observe from Table 6 that on random, unstructured instances, the amount of perturbation induced with DPX is much stronger than with UX and BX, since DPX is designed to exploit the problem structure that is missing for instances of Type II. Indeed, the degree of perturbation p_{str} with DPX crossover on unstructured instances is almost 100%. On the other hand, the amount of perturbation caused by DPX is greatly smaller when applied to more structured instances of Type IV.

Table 5: Percentage deviations $\bar{\delta}_{avg}$ of the average solution (obtained after 10 runs) from the published best-known result for the three versions of our BMA integrating respectively the uniform (UX), the block (BX), and the distance preserving (DPX) crossover.

Instance	UX	BX	DPX	*Instance*	UX	BX	DPX
tai40a	**0.059**(2)	0.067(1)	0.074(0)	sko100a	**0.000**(10)	**0.000**(10)	0.001(9)
tai50a	**0.135**(2)	0.250(1)	0.286(0)	sko100b	**0.000**(10)	**0.000**(10)	**0.000**(10)
tai60a	**0.220**(2)	0.249(1)	0.270(0)	sko100c	**0.000**(10)	**0.000**(10)	**0.000**(10)
tai80a	**0.410**(0)	0.420(0)	0.536(0)	sko100d	**0.000**(10)	**0.000**(10)	**0.000**(10)
tai100a	0.340(0)	**0.270**(0)	0.431(0)	sko100e	**0.000**(10)	**0.000**(10)	**0.000**(10)
sko72	**0.000**(10)	**0.000**(10)	**0.000**(10)	sko100f	**0.000**(10)	**0.000**(10)	**0.000**(10)
sko81	**0.000**(10)	**0.000**(10)	**0.000**(10)	tai80b	**0.000**(10)	**0.000**(10)	**0.000**(10)
sko90	**0.000**(10)	0.010(8)	0.003(9)	tai100b	**0.000**(10)	**0.000**(10)	**0.000**(10)

Table 6: The average perturbation degree p_{str} over 1500 generations introduced by three versions of our BMA integrating respectively the uniform (UX), the block (BX), and the distance preserving (DPX) crossover. p_{str} is expressed as a percentage of n.

Instance	UX	BX	DPX	*Instance*	UX	BX	DPX
tai40a	45.8	36.7	95.9	sko100a	21.8	13.7	95.4
tai50a	46.9	39.0	97.1	sko100b	16.9	14.0	47.6
tai60a	45.2	38.6	97.5	sko100c	18.9	15.9	41.6
tai80a	48.4	36.9	98.4	sko100d	18.5	11.8	40.3
tai100a	46.9	36.5	98.6	sko100e	29.5	22.7	43.1
sko72	22.3	32.4	76.8	sko100f	28.3	20.6	63.9
sko81	25.4	21.1	67.1	tai80b	13.7	10.6	35.6
sko90	27.9	24.6	49.0	tai100b	17.3	11.9	27.5

6. Conclusion

In this article, we presented a simple and effective memetic algorithm (BMA) for the well-known quadratic assignment problem. BMA combines a dedicated local search named Breakout Local Search (BLS) with the standard uniform crossover (UX), a simple pool updating strategy and an adaptive mutation mechanism.

The BLS procedure, which is the key component of BMA, alternates between a local search phase (to reach local optima) and a dedicated perturbation phase (to discover new promising regions). The perturbation mechanism of BLS dynamically determines the number of perturbation moves and adaptively chooses between two types of perturbation moves of different intensities depending on the current search state.

Genetic operators (crossover and mutation) are integrated to further enforce the search capacity of BMA. The choice of these operators is based on observations made from a landscape analysis which investigates the structure of QAP instances. The analysis revealed a very low fitness-distance correlation between local optima for randomly generated instances, and a medium correlation for instances of other types. These observations provide a basis to explain why the standard uniform crossover generally leads to better results for unstructured instances (Type II) than a crossover that tries to exploit the problem structure.

We evaluated the proposed algorithm on the set of 135 benchmark instances from the QAPLIB. Computational results revealed that BMA performs very well on these instances. Indeed, BMA outperforms its local search component (BLS) and is able to reach the current best-known solution for 133 instances.

In particular, BMA performs particularly well on unstructured instances (Type II) which are considered to be the hardest for the existing QAP methods. For real-life like instances (Type III) and instances with grid distances (Type IV), BMA remains competitive with respect to the best performing approaches as well. The computing time needed for our proposed algorithm to reach its best solution varies on average from several seconds to about one hour, except for the largest problem for which 7 hours are needed. When the computing time is limited to 30 minutes, BMA is still able to attain the best known result for 130 out of the 135 benchmark instances.

Acknowledgment

We are grateful to the anonymous referees for valuable suggestions and comments which helped us improve the paper. This work was carried out when the first author was with the LERIA, Université d'Angers (France) and was partially supported by the Region of 'Pays de la Loire' (France) within the RADAPOP and LigeRO Projects.

References

Ahuja, R.K., Orlin J.B., & Tiwari, A. (2000). A greedy genetic algorithm for the quadratic assignment problem. *Computers & Operations Research*, 27(10), 917–934.

Anstreicher, K. M. (2003). Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, Series B 97, 27–42.

Battiti, R., & Tecchiolli, G. (1994). The reactive tabu search. *ORSA Journal of Computing*, 6(2), 126–140.

Benlic, U., & Hao, J.K. (2011). A multilevel memetic approach for improving graph k-partitions. *IEEE Transactions on Evolutionary Computation*, 15(5), 624–642, 2011.

Benlic, U., & Hao, J.K. (2013a). Breakout local search for maximum clique problems. *Computers & Operations Research*, 40(1), 192–206.

Benlic, U., & Hao, J.K. (2013b). Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence*, 26(3), 1162–1173.

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

Benlic, U., & Hao, J.K. (2013d). Breakout local search for the vertex separator problem. In F. Rossi (Ed.) *Proceedings of the 23th Intl. Joint Conference on Artificial Intelligence (IJCAI-13)*, IJCAI/AAAI Press, pages 461–467, Beijing, China, August 2013.

Blum, C., Puchinger, J., Raidl, G.R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing* 11(6), 4135–4151.

Boese, K.D. (1995). Cost versus distance in the traveling salesman problem. Technical Report TR-950018, UCLA CS Department.

Bontouxa, B., Artigues, C., & Feillet, D. (2010). A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem. *Computers & Operations Research*, 37, 1844–1852.

Burkard, R.E. (1991). Locations with spatial interactions: The quadratic assignment problem. In *P. Mirchandani and L. Francis (Eds.)*, Discrete Location Theory, New York.

Cheng, C.H. Ho, S.C., & Kwan, C.L. (2012). The use of meta-heuristics for airport gate assignment. *Expert Systems with Applications*, 39(16), 12430–12437.

Dorne, R., & Hao, J.K. (1998). A New Genetic Local Search Algorithm for Graph Coloring. In A. E. Eiben, T. Bck, M. Schoenauer, H.P. Schwefel (Eds.): PPSN 1998, *Lecture Notes in Computer Science*, 1498 745–754.

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

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

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

Eshelman, L.J. (1991). The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pages 265–283, Morgan Kaufmann.

Fleurent, C., & Ferland, J.A. (1993). Genetic hybrids for the quadratic assignment problem. *DIMACS Series in Mathematics and Theoretical Computer Science*, 173–187.

Garey. M., & Johnson, D. (1979). Computers and Intractability. Freeman. N.Y..

Hao, J.K. (2012). Memetic algorithms in discrete optimization. In F. Neri, C. Cotta and P. Moscato (Eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Chapter 6, pages 73–94.

Hart, W.E., Krasnogor, N., & Smith, J.E. (2004). Recent advances in memetic algorithms. Studies in Fuzziness and Soft Computing 166, Springer.

Hassin, M., Levin, A., & Sviridenko, M. (2009). Approximating the minimum quadratic assignment problems. *ACM Transactions on Algorithms*, 6(1), article number 18.

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

James, T. Rego, C., & Glover, F. (2009b). 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), 579–596.

Jones, T., & S. Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmann, 184–192.

Kelly, J.P., Laguna, M., & Glover, F. (1994). A study of diversification strategies for the quadratic assignment problem. *Computers & Operations Research*, 21(8), 885–893.

Krasnogor, N., & Smith, J. (2005). A tutorial for competent memetic algorithms: model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computing*, 9(5), 474–488.

Li, F., Xu, L.D., Jin, C., & Wang, H. (2012). Random assignment method based on genetic algorithms and its application in resource allocation. *Expert Systems with Applications*, 39(15), 12213–12219.

Lü, Z., & Hao, J.K. (2010). A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1), 241–250.

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

Miao, Z., Cai, S., & Xu. D. (2014). Applying an adaptive tabu search algorithm to optimize truck-dock assignment in the crossdock management system. *Expert Systems with Applications*, 41(1), 16–22.

Misevicius, A. (2004). An improved hybrid genetic algorithm: New results for the quadratic assignment problem. *Knowledge Based Systems*, 17(2-4), 65–73.

Misevicius, A., & Kilda, B. (2006). Iterated tabu search: An improvement to standard tabu search. *Information Technology and Control*, 35(3), 187–197.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech Concurrent Computation Program, Report 826, California Institute of Technology, Pasadena, California, USA.

Moscato, P., & Cotta, C. (2003). A Gentle Introduction to memetic algorithms. In F. Glover and G. A. Kochenberger (Eds.), Handbook of Metaheuristic. Kluwer, Norwell, Massachusetts, USA.

Neri, F., Cotta, C., & Moscato, P. (Eds.) (2012). Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Springer.

Nikolić, M., & Teodorović, D. (2014). A simultaneous transit network design and frequency setting: computing with bees. *Expert Systems with Applications*, 41(16), 7200–7209.

Pardalos, P.M., Rendl, F., & Wolkowicz, H. (1994). The quadratic assignment problem: A survey and recent developments. *In Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, volume 16 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1–42.

Porumbel, D.C., Hao, J.K., & Kuntz, P. (2010). An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. *Computers & Operations Research*, 37(10), 1822–1832.

Reeves, C.R. (1997). Landscapes, operators and heuristic search. *Annals of Operations Research*, 86, 473–490.

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

Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3), 1519–1539.

Taillard, E. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17, 443-455.

Wilhelm, M.R., & Ward, T.L. (1987). Solving quadratic assignment problems by simulated annealing. *IIE Transactions*, 19(1), 107–119.

Appendix

This appendix includes two tables (Tables 7 and 8). Table 7 shows the best, average and worst result obtained by BMA after 10 independent runs for 21 selected QAPLIB instances, within a time limit of 2 hours (10 hours for instance *tho*150) and 30 minutes respectively.

Table 8 shows the computational results of our BMA algorithm on the set of 114 easy QAP instances of the QAPLIB. For all these instances, each run of BMA can attain the best known result with a computing time ranging from 0 second to at most 2.5 minutes.

Table 7: Computational results of the proposed BMA algorithm with a time limit of 2 hours and 30 minutes respectively on the set of 21 hard instances from the QAPLIB. Columns '% $\bar{\delta}_{best}$', '% $\bar{\delta}_{avg}$' and '% $\bar{\delta}_{worst}$' show respectively the percentage deviation of our best, average, and worst solution from the best-known solution (BKR) over all the trials; Column '$t_{avg}(min)$' indicates the average computing time in minutes when the best solution in each trial was attained.

Problem		Time limit of 2 hours				Time limit of 30 minutes			
Name	BKS	% $\bar{\delta}_{best}$	% $\bar{\delta}_{avg}$	% $\bar{\delta}_{worst}$	$t_{avg}(m)$	% $\bar{\delta}_{best}$	% $\bar{\delta}_{avg}$	% $\bar{\delta}_{worst}$	$t_{avg}(m)$
tai40a	3139370	0.000(2)	0.059	0.074	8.1	0.000(1)	0.067	0.074	3.7
tai50a	4938796	0.000(2)	0.131	0.291	42.0	0.053(0)	0.264	0.409	3.8
tai60a	7205962	0.000(2)	0.144	0.259	67.5	0.165(0)	0.307	0.369	13.9
tai80a	13499184	0.246(0)	0.426	0.561	65.8	0.430(0)	0.563	0.670	16.4
tai100a	21052466	0.269(0)	0.405	0.550	44.1	0.340(0)	0.582	0.715	11.2
tai50b	458821517	0.000(10)	0.000	0.000	1.2	0.000(10)	0.000	0.000	1.2
tai60b	608215054	0.000(10)	0.000	0.000	5.2	0.000(10)	0.000	0.000	5.2
tai80b	818415043	0.000(10)	0.000	0.000	31.3	0.000(9)	0.028	0.283	3.1
tai100b	1185996137	0.000(10)	0.000	0.000	13.6	0.000(8)	0.028	0.100	7.9
tai150b	498896643	0.000(1)	0.060	0.365	78.1	0.000(1)	0.198	0.225	15.0
sko72	66256	0.000(10)	0.000	0.000	3.5	0.000(8)	0.012	0.063	1.3
sko81	90998	0.000(10)	0.000	0.000	4.3	0.000(10)	0.000	0.000	4.5
sko90	115534	0.000(10)	0.000	0.000	15.3	0.000(7)	0.025	0.123	4.43
sko100a	152002	0.000(10)	0.000	0.000	22.3	0.000(7)	0.005	0.016	9.8
sko100b	153890	0.000(10)	0.000	0.000	56.5	0.000(7)	0.001	0.004	6.75
sko100c	147862	0.000(10)	0.000	0.000	12.0	0.000(10)	0.000	0.000	8.9
sko100d	149576	0.000(9)	0.006	0.065	20.9	0.000(6)	0.024	0.176	11.45
sko100e	149150	0.000(10)	0.000	0.000	11.9	0.000(10)	0.000	0.000	9.93
sko100f	149036	0.000(10)	0.000	0.000	23.0	0.000(7)	0.001	0.005	10.6
wil100	273038	0.000(10)	0.000	0.000	14.5	0.000(10)	0.000	0.000	7.7
tho150	8133398	0.000(3)	0.007	0.064	416.4	0.019(0)	0.057	0.096	22.4

Table 8: Computational results of our BMA algorithm for the set of 114 easy instances of the QAPLIB. Column 'BKR' indicates the optimal/best-known result ever reported in the literature. Column '#*best*′ indicates the number of times a best-known solution was found over 100 executions; Column '% $\bar{\delta}_{avg}$' shows the percentage deviation of our average solution from the best-known solution (BKR) over all the trials; Column '$t_{avg}(sec)$' indicates the average computing time in seconds when the best solution in each trial was attained.

Instance	*BKR*	#best	% $\bar{\delta}_{avg}$	$t_{avg}(sec)$	*Instance*	*BKR*	#best	% $\bar{\delta}_{avg}$	$t_{avg}(sec)$
bur26a	5426670	100/100	0.000	0.8	tai20a	703482	100/100	0.000	0.0
bur26b	3817852	100/100	0.000	0.8	tai25a	1167256	100/100	0.000	0.0
bur26c	5426795	100/100	0.000	0.5	tai30a	1818146	100/100	0.000	0.0
bur26d	3821225	100/100	0.000	0.2	tai35a	242002	100/100	0.000	0.1
bur26e	5386879	100/100	0.000	0.2	rou15	354210	100/100	0.000	0.4
bur26f	3782044	100/100	0.000	0.2	rou20	725522	100/100	0.000	0.5
bur26g	10117172	100/100	0.000	0.3	lipa20a	3683	100/100	0.000	0.1
bur26h	7098658	100/100	0.000	0.2	lipa20b	27076	100/100	0.000	0.0
tai64c	1855928	100/100	0.000	2.4	lipa30a	13178	100/100	0.000	0.2
chr12a	9552	100/100	0.000	0.2	lipa30b	151426	100/100	0.000	0.1
chr12b	9742	100/100	0.000	0.2	lipa40a	31538	100/100	0.000	1.1
chr12c	11156	100/100	0.000	0.2	lipa40b	476581	100/100	0.000	0.3
chr15a	9896	100/100	0.000	0.3	lipa50a	62093	100/100	0.000	1.5
chr15b	7990	100/100	0.000	0.4	lipa50b	1210244	100/100	0.000	0.4
chr15c	9504	100/100	0.000	0.3	lipa60a	107218	100/100	0.000	8.2
chr18a	11098	100/100	0.000	0.4	lipa60b	2520135	100/100	0.000	0.4
chr18b	1534	100/100	0.000	0.5	lipa70a	169755	100/100	0.000	30.7
chr20a	2192	100/100	0.000	2.1	lipa70b	4603200	100/100	0.000	1.5
chr20b	2298	100/100	0.000	5.7	lipa80a	253195	100/100	0.000	51.2
chr20c	14142	100/100	0.000	0.8	lipa80b	7763962	100/100	0.000	4.1
chr22a	6156	100/100	0.000	1.6	lipa90a	360630	100/100	0.000	149.1
chr22b	6194	100/100	0.000	1.8	lipa90b	12490441	100/100	0.000	4.2
chr25a	3796	100/100	0.000	9.5	tai10a	135028	100/100	0.000	0.0
els19	17212548	100/100	0.000	0.4	tai12a	224416	100/100	0.000	0.1
esc16a	68	100/100	0.000	0.0	tai15a	388214	100/100	0.000	0.2
esc16b	292	100/100	0.000	0.0	tai17a	491812	100/100	0.000	0.4
esc16c	160	100/100	0.000	0.0	nug12	578	100/100	0.000	0.1
esc16d	16	100/100	0.000	0.0	nug14	1014	100/100	0.000	0.1
esc16e	28	100/100	0.000	0.0	nug15	1150	100/100	0.000	0.2
esc16f	0	100/100	0.000	0.1	nug16a	1610	100/100	0.000	0.2
esc16g	26	100/100	0.000	0.1	nug16b	1240	100/100	0.000	0.3
esc16h	996	100/100	0.000	0.0	nug17	1732	100/100	0.000	0.3
esc16i	14	100/100	0.000	0.0	nug18	1930	100/100	0.000	0.3
esc16j	8	100/100	0.000	0.0	nug20	2570	100/100	0.000	0.4
esc32a	130	100/100	0.000	0.0	nug21	2438	100/100	0.000	0.4
esc32b	168	100/100	0.000	0.1	nug22	3596	100/100	0.000	0.4
esc32c	642	100/100	0.000	0.1	nug24	3488	100/100	0.000	0.6
esc32d	200	100/100	0.000	0.1	nug25	3744	100/100	0.000	0.6
esc32e	2	100/100	0.000	0.1	nug27	5234	100/100	0.000	0.5
esc32g	6	100/100	0.000	0.0	nug28	5166	100/100	0.000	0.7
esc32h	438	100/100	0.000	0.1	nug30	6124	100/100	0.000	1.8
esc64a	116	100/100	0.000	0.2	scr12	31410	100/100	0.000	0.1
esc128	64	100/100	0.000	1.3	scr15	51140	100/100	0.000	0.3
had12	1652	100/100	0.000	0.2	scr20	110030	100/100	0.000	0.5
had14	2724	100/100	0.000	0.1	tho30	149936	100/100	0.000	0.3
had16	3720	100/100	0.000	0.3	tho40	240516	100/100	0.000	0.4
had18	5358	100/100	0.000	0.2	wil50	48816	100/100	0.000	23.6
had20	6922	100/100	0.000	0.3	tai10b	1183760	100/100	0.000	0.0
kra30a	88900	100/100	0.000	2.4	tai12b	39464925	100/100	0.000	0.0
kra30b	91420	100/100	0.000	1.2	tai15b	51765268	100/100	0.000	0.1
kra32	88700	100/100	0.000	1.1	tai30b	637117113	100/100	0.000	0.0
ste36a	9526	100/100	0.000	4.9	tai35b	283315445	100/100	0.000	0.0
ste36b	15852	100/100	0.000	1.3	tai40b	637250948	100/100	0.000	0.2
ste36c	8239110	100/100	0.000	2.6	sko42	15812	100/100	0.000	1.3
rou12	235528	100/100	0.000	0.2	sko49	23386	100/100	0.000	0.5
tai20b	122455319	100/100	0.000	0.0	sko56	34458	100/100	0.000	1.0
tai25b	344355646	100/100	0.000	0.0	sko64	48498	100/100	0.000	1.5