Bandit-Inspired Memetic Algorithms for Solving Quadratic Assignment Problems

Puglierin, Francesco; Drugan, Madalina M.; Wiering, Marco

Published in:
Proceedings of IEEE International Conference on Evolutionary Computation

Publication date:
2013

Bandit-Inspired Memetic Algorithms for Solving Quadratic Assignment Problems

Francesco Puglierin
Information and Computing Sciences
Utrecht University, The Netherlands
Email: francesco@puglier.in

Mădălina Drugan
Artificial Intelligence Lab
Vrije Universiteit Brussel, Belgium
Email: mdrugan@vub.ac.be

Marco Wiering (*IEEE Member*)
Department of Artificial Intelligence
University of Groningen, The Netherlands
Email: m.a.wiering@rug.nl

***Abstract*—In this paper we propose a novel algorithm called the Bandit-Inspired Memetic Algorithm (BIMA) and we have applied it to solve different large instances of the Quadratic Assignment Problem (QAP). Like other memetic algorithms, BIMA makes use of local search and a population of solutions. The novelty lies in the use of multi-armed bandit algorithms and assignment matrices for generating novel solutions, which will then be brought to a local minimum by local search. We have compared BIMA to multi-start local search (MLS) and iterated local search (ILS) on five QAP instances, and the results show that BIMA significantly outperforms these competitors.**

***Index Terms*—Meta-heuristics, Memetic Algorithms, Combinatorial Optimization, Quadratic Assignment Problem, Multi-armed Bandit Algorithms**

I. Introduction

Many real-world problems in logistics, transport, and manufacturing can be modeled as combinatorial optimization problems. These problems have a huge set of possible solutions, and the goal of an optimization algorithm is to find the best solution. Since most of these problems are known to be NP-hard, the most promising optimization algorithms use heuristics to find good solutions without huge computational costs. There are many different optimization algorithms, such as Genetic Algorithms [1], Tabu Search [2], Simulated Annealing being derived from the Metropolis Algorithm [3], and Ant Colony Systems [4]. A promising class of methods are meta-heuristics [5] and memetic algorithms [6], [7] that integrate a simple local search algorithm in their framework. The advantage of using local search is that it automatically finds a local optimum when given some newly generated solution. Therefore, the global search only has to consider the space of local optima, which is usually much smaller than the space containing all feasible solutions.

Quadratic Assignment Problems. The quadratic assignment problem (QAP) is a combinatorial optimization problem introduced by Koopmans and Beckmann in 1957 as a formal model for allocating indivisible economical activities [8]. Informally, there is a given number of facilities to assign to the same number of locations in an optimal way; a mutual distance is given between locations, as is the flow, a number which quantifies the mutual interaction between facilities. The optimality is reached by placing the facilities in the locations so that the complete solution minimizes the summation of the products of distance and flow between all the facilities.

Several optimization objectives of the problem have been proposed in literature; the following one, proposed by Çela in [9], is probably the most commonly used:

$$cost(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)}, \tag{1}$$

where n is the size of the problem instance, f is the *flow matrix*, f_{ij} is the directed flow between facility i and facility j, d is the *distance matrix*, and d_{ij} is the directed distance between location i and location j. Finally, π represents a possible permutation over (1,2,...,n) and $\pi(i)$ corresponds to the index of the location to which facility i is assigned. The aim is to minimize the cost function defined by formula (1). The QAP has been proven NP-hard in [10].

Main contributions. This paper describes the novel bandit-inspired memetic algorithm (BIMA), which combines memetic algorithms with multi-armed bandit algorithms [11] and local assignment matrices to generate novel solutions. BIMA uses a population of solutions and local search to obtain a set of local minima from newly generated solutions. The novelty in the method is the use of a multi-armed bandit algorithm on a set of local assignment fitness matrices to globally search for promising novel solutions. The idea of BIMA is to select local assignments using the multi-armed bandit algorithm and to enforce these promising assignments on a solution of an individual in the population. After generating a novel solution, local search is used to obtain a local optimum.

Multi-armed bandit (MAB) algorithms are an important framework in reinforcement learning [12], [13] used mainly to study the theoretical properties of these advanced machine learning algorithms. In evolutionary computation, adaptive operator selection algorithms have made use of MAB algorithms to select a good genetic operator (like mutation operators with different exchange rates [14], [15]). MABs are preferred to other on-line learning algorithms because they are simple, easy to implement and to tune. To our knowledge the application of MAB algorithms to combinatorial optimization for keeping information about the structure of the search space and to generate novel solutions by selecting assignments to enforce on a solution is new. In this context, the strength of bandit algorithms is that they integrate information about the fitness and popularity of assignments over time in order to optimally

cope with the exploration/exploitation trade-off.

We have performed a comparison of BIMA to multi-start local search and iterated local search on five QAP instances. The results show that BIMA obtains significantly better results on all problem instances.

Outline of this paper. In Section II, we describe several optimization algorithms related to meta-heuristics and memetic algorithms. In Section III, we describe the Bandit-Inspired Memetic Algorithm. Section IV describes related work that inspired the development of BIMA. Then, in Section V the experimental setup and results will be given. Section VI concludes the paper and describes some possibilities for future work.

II. Background

In this section, we present the two main algorithmic concepts upon which the BIMA algorithm is built.

A. Local-search Based Algorithms

Intuitively, local search (LS) [16] starts from an initial solution and iteratively generates new solutions using a neighborhood strategy. Each step, a solution that improves over the existing best-so-far solution is chosen. The algorithm stops when there is no improvement possible. Best improvement LS explores *all* the individuals in the neighborhood of a solution and selects the best solution that is improving over the initial solution and all the other visited solutions. In a *first-improvement* local search the hunt stops as soon as a better solution is encountered. The local search technique has the limitation of stopping once a local minimum is reached, which translates into reaching a solution whose neighborhood does not contain any improvement.

A suitable neighborhood operator for QAPs is the 2-exchange swapping operator that swaps the locations of two different facilities. This operator is attractive because of its linear time to compute the change in the cost function with the condition that the flow and distance matrices are symmetrical [7]. The size of the neighborhood increases quadratically with the number of facilities.

Multi-start Local Search (MLS). Multi-start local search is probably the simplest meta-heuristic. The idea of MLS consists of restarting the search from a random solution once a local minimum has been reached by local search, and to repeat this until a termination condition is reached. The resulting MLS method can be classified as a meta-heuristic due to its ability of exploring different areas of the search space. There are certain limitations in MLS design because it is basically random sampling in the space of local optima. Therefore, it does not scale up well to a large number of local optima.

Iterated Local Search (ILS). In permutation problems like QAPs, the mutation operator interchanges facilities between different locations. When LS uses the 2-exchange operator to generate a neighborhood, the mutation operator should exchange at least the locations of three facilities to escape from the region of attraction of a local optimum. The m-exchange mutation uniform randomly selects m distinct pairs of facilities for which the locations are sequentially exchanged. This global step to generate a new initial solution is performed after a local optimum has been found using the best found solution during the entire run. Thus, the only difference between MLS and ILS is in the technique used to restart LS. The advantage of ILS is that it exploits possible structural relationships in the search space (i.e. correlations or blocks in the QAP matrices).

Memetic Algorithms. In memetic algorithms [7], [6]. a genetic algorithm is combined with an individual refinement of the single solutions using local search. The advantage is that the quality of solutions in the population is improved before they share genetic information with their peers. Memetic algorithms have been shown to outperform genetic algorithms for some difficult problems (see e.g., [7]). ILS and MLS store only a single best-so-far solution whereas memetic algorithms use a population of solutions optimized with local search.

B. Multi-armed Bandits

One of the most important problems involved in a search algorithm is to optimally cope with the exploration/exploitation trade-off. It is important that previously found solutions that are very good are used to construct new solutions (exploitation), but the new solution should be sufficiently different in order to explore large parts of the search space (exploration). Achieving a good compromise between the two extremes is not a problem found only in meta-heuristics; it is in fact common in reinforcement learning [12], planning [17] and dealing with imperfect knowledge in general. One of the archetypal examples of such dilemma is the **Multi-Armed Bandit** (MAB) model, known since the beginning of the century and first formalized in its current form by Robbins in 1951 [18].

The MAB models the uncertainty faced by a gambler who has to choose how to spend his coins among K one-armed bandits, each behaving according to an independent and initially unknown probability distribution, in order to maximize his final profit. An alternative way of describing the gambler's goal is to talk about regret minimization, where the regret is the difference between the profit obtained by an ideal sequence of plays (always pulling the best arm, if the reward distributions are stationary) and the performed plays.

There are multiple bandit solvers, and our focus is on optimistic index policies, and on the **Upper Convergence Bound** (cUCB for short, [11]) in particular. cUCB is one of the most applied techniques to solve multi-armed bandit problems. The idea behind cUCB involves computing for each arm an upper bound for the returned reward, basing the bound estimation on both the rewards history and the accuracy of the reward estimation (the number of pulls performed on the arm). Whenever the gambler has to choose which arm to pull, he finds out the value of cUCB for each arm according to formula (2) and selects the highest scoring one.

$$Score_j^t = \bar{x}_j^t + \sqrt{\frac{c \ln \sum_k p_k^t}{p_j^t}} \tag{2}$$

The first term in the formula, $\bar{x}_j^t$, encodes the expected average reward for arm j according to the knowledge available in time-step t. Always choosing the arm with the highest expected reward would result in a purely exploitative algorithm, so the formula includes a second term to deal with exploration. The variable p_j^t represents the number of times arm j has been pulled at time-step t, making the value of the second term in formula (2) inversely proportional to the arm popularity. c is some parameter that can be tweaked, although its value is often just set to 2, like in the original UCB algorithm [11].

III. Bandit-Inspired Memetic Algorithm (BIMA)

In this section, we describe how we combined multi-armed bandits with local search in a novel memetic algorithm. BIMA consists of 3 important factors: (1) a population of solutions, (2) a set of local assignment matrices, and (3) a multi-armed bandit algorithm to select assignments to enforce on a solution. Algorithm (1) shows the pseudocode of BIMA.

In the initialization phase, a pool of random solutions is generated. Here ps denotes the population size. Then, **2opt_local_search** creates an initial pool with only local minima.

After that the algorithm constantly selects assignments to enforce on a solution of an individual in the pool. This is done by first selecting a subset A^t of all allowed assignments. This subset either contains the assignments from (1) another parent, a randomly selected donor parent, or (2) the assignments currently in the whole population, or (3) all possible assignments (which contain $n \times n$ possibilities). The decision which subset to select is made randomly.

After a subset is selected, the method selects assignments based on some parameters and local assignment matrices. This method, **select_assignments** forms the core of the algorithm, and will be explained later.

The method **enforce_assignments** simply puts the selected assignments in the selected individual. To create valid solutions, local swaps will be performed between the assignments to enforce and the assignments that were in the individual before.

Finally, the newly generated solution is brought to a local minimum using **2opt_local_search**. After that the fitness of the new solution is compared to the fitness of the previous solution of the individual and replaces the old solution if its fitness (cost) is lower.

The core of the algorithm is the **select_assignments** step. This step makes use of local assignment matrices and a multi-armed bandit algorithm, which we will now explain in more detail.

Algorithm 1 BIMA-QAP

$Pool(rnd) \leftarrow \{\pi_1^{rnd}, \pi_2^{rnd}, \ldots, \pi_{ps}^{rnd}\}$

for all π_i^{rnd} in $Pool(rnd)$ **do**
  $\pi_i^0 \leftarrow$ **2opt_local_search(**π_i^{rnd}**)**
end for
$Pool(0) \leftarrow \{\pi_1^0, \pi_2^0, \ldots, \pi_{ps}^0\}$
$t \leftarrow 0$

repeat
  $A^t \leftarrow$ **subset_assignments(**$donor$**,** $population$**,** all**)**
  $SA^t \leftarrow$ **select_assignments(**A^t, l, ms, c**)**
  $\pi_i^{tmp} \leftarrow$ **enforce_assignments(**SA^t, π_i^t**)**
  $\pi_i^{new} \leftarrow$ **2opt_local_search(**π_i^{tmp}**)**

  if fitness(π_i^{new}**)** $\leq$ **fitness(**π_i^t**) then**
    $Pool(t+1) \leftarrow Pool(t) \setminus \{\pi_i^t\} \cup \{\pi_i^{new}\}$
  else
    $Pool(t+1) \leftarrow Pool(t)$
  end if

  $t \leftarrow t + 1$

until stop_condition

A. Local Assignment Fitness Matrices

An approach common to Tabu Search [2], ACS algorithms [19] and univariate EDA algorithms [20], [21] is to associate to single assignments flags or values that are then used to guide the search. Ant Colonies and several EDA algorithms in particular approach the QAP by keeping an $n \times n$ matrix encoding the desirability of each assignment $\chi_{i,j}$ in time-step t. These assignment matrices store the desirability of assigning each facility i to location j, and therefore have size $n \times n$, where n is the size of the problem instance. The best assignments can then be selected from the assignment matrix and used to change an existing solution. This kind of explicit memory is appealing, because it can add an additional dimension to the search by efficiently reusing data already gathered during the algorithm execution.

The idea in BIMA is to keep track of a local fitness value for each assignment, and to store them in the local fitness matrix $\tilde{F}^t(l)$, where l denotes the individual in the pool. The value $\tilde{f}_{ij}^t(l)$ in the matrix represents the aggregate fitness associated to assignment $\chi_{i,j}$. The local fitness matrix belongs to a single individual in the solution pool.

At every update, $\tilde{f}_{ij}^t(l)$ is computed by linearly combining the average fitness of the solutions visited by individual l containing that assignment ($\bar{f}_{ij}^t(l)$, with weight w_1) and the fitness of the best solution with $\chi_{i,j}$ in it ($\check{f}_{ij}^t(l)$, with weight $1 - w_1$), as in Equation (3). These values are updated every time a solution of individual l containing that assignment is evaluated.

$$\tilde{f}_{ij}^t(l) = w_1 \bar{f}_{ij}^t(l) + (1 - w_1) \check{f}_{ij}^t(l) \tag{3}$$

The great majority of the evaluations takes place during local search. Updating the matrix values during local search is considered sort of compulsory to get a decent estimation of the

fitness. We note that all fitness values are normalized between 0 and 1, which is required by the used bandit algorithm.

B. Local Assignment Pull-count Matrices

Next to the local fitness matrices, the algorithm uses local pull-count matrices. The pull counts $p_{ij}^t(l)$ are stored in matrix $P^t(l)$ and updated whenever the associated assignment is involved in a solution belonging to individual l being evaluated. The pull-counts are used to compute the accuracy of the estimation of the associated local assignment fitness values. The main question is when to update the counts associated to each assignment. This should go together with the update of the fitness estimations, which is mostly done during local search. Considering that the formula being put together is not going to be involved with local search, this means that in the vast majority of cases the arms are not pulled because of their higher index score. In fact, most of the pulls are performed implicitly while traversing 2-opt neighborhoods.

An interesting consequence of this is that when the bandit formula, which will be explained next, is used to select a set of assignments to enforce on a specific solution, the assignments not involved in local search will be more likely to be selected due to their higher exploration term. These assignments are also more likely to be outside the basin of attraction of the 2-optimized solution, which is a very desirable property for an operator that has to complement local search.

The local pull-count and fitness matrices are zero-initialized. Since local search is performed as the first step on the random initial solutions, we found that in practice after the first local search phase all assignments were visited. However, to deal with the theoretical possibility that an assignment was not explored, we checked the values of the pull-count matrices and initialized them to 1 in case they were never visited. We did this since the pull-counts should always be positive integers for the bandit algorithm.

C. Multi-armed Bandit Algorithms

We will now describe how we adapted the cUCB bandit algorithm to select assignments to enforce on some solution of a selected individual. Devised as a complement to local search, **select_assignments(**A^t, l, ms, c**)** is used once in each macro-iteration. An individual l is randomly chosen from the ps available (ps is the population size), then ms assignments are selected to be enforced on the associated solution. These assignments are the ones from the selected subset of assignments A^t for which the following formula takes the highest scores:

$$Score_{ij}^t = \left(1 - \frac{\tilde{f}_{ij}^t(l) - \min\limits_{(k,l)\in A^t}(\tilde{f}_{kl}^t(l))}{\max\limits_{(k,l)\in A^t}(\tilde{f}_{kl}^t(l)) - \min\limits_{(k,l)\in A^t}(\tilde{f}_{kl}^t(l))}\right) + \sqrt{\frac{c \ln \sum\limits_{(k,l)\in A^t} p_{kl}^t(l)}{p_{ij}^t(l)}}$$

The first part in this equation is the exploitation term. It uses 1 minus the normalized (between 0 and 1) fitness value to prefer the assignments from A^t having the lowest cost. The second term is the exploration term from cUCB now making use of the local pull-count matrices.

We want to note that many of the properties characterizing the MAB are lost in the combinatorial optimization scenario. More than one assignment needs to be enforced at once, otherwise the chances of getting out of the basin of attraction of the previous minimum would be pretty slim. Furthermore, the quality measure of an assignment is dependent on the rest of the assignments in the solution as well. This implies that assignments are not independent as arms are assumed to be, and that also non "pulled" assignments are influencing the feedback. One more consequence is that the feedback associated to an assignment can change according to the neighborhood(s) being analyzed. Nonetheless, the loss of theoretical bounds and properties does not take away from the fact that it is interesting to see how an algorithm like cUCB performs in balancing exploration and exploitation in this new algorithm.

IV. Related work

In this section we will describe several methods related to BIMA. We note that it is not our intention to fully cover the field of combinatorial optimization algorithms, since the field is simply too large to cover in one paper.

Ant Colony Systems. Ant Colony Systems (ACS), first introduced by Marco Dorigo in [4], attempt to solve combinatorial optimization problems by imitating the behavior of worker ants hunting for food. In ACS virtual ants are traveling around the search space, composing solutions by combining heuristic evaluations with values left behind in virtual pheromone trails, to which an evaporation factor is applied at each iteration. The pheromone trails determine the goodness of a specific assignment. By having different ants constructing different solutions using them, the best solution(s) can be used to increase the pheromone trails of assignments belonging to these solutions. Then, in the future more ants will compose solutions with the assignments in the best found solutions.

A later implementation of the Ant System specifically tailored for the Quadratic Assignment Problem is HAS-QAP, introduced by Gambardella, Taillard and Dorigo in 1999 [19]. Here the ants use the information in the trails to alter previously generated solutions, instead of generating new ones from scratch. Another relevant tweak is the inclusion of a local search routine that improves the solutions generated by the ants themselves. Both these approaches, altering previous solutions and local search, found their way into BIMA as well.

Estimation of Distribution Algorithms. Another category of population-based algorithms are Estimation of Distribution Algorithms (EDAs) [22], [20]. These methods, whose theoretical foundation is in probability theory, build a probabilistic model around good solutions and use that as a guide in exploring the search space. The model is constantly updated to include information about newly generated solutions. An

algorithm related to BIMA was proposed in [21], where an assignment matrix contains values that directly encode the probability of altering a solution with each of the assignments. These probabilities are adapted based on the best found solutions.

V. Experimental Setup and Results

In this section, we will first explain which QAP instances we have used to compare BIMA to MLS and ILS. After that we will present our experimental results.

A. Experimental Setup

We used QAP instances from QAPLIB, which is a repository for QAP instances and results. To compare BIMA to MLS and ILS, five instances of various sizes have been selected from QAPLIB: *nug30*, *ste36a*, *tai60a*, *tai80a*, and *sko100a*. Table I provides for each instance the fitness of the best known minimum, the size n of the instances, the amount of fitness evaluations specified for the stopping condition, and the number of runs we performed with BIMA, MLS, and ILS. The number of evaluations is the same for all methods, and this number was selected based on a compromise between the goodness of found solutions and the computational time needed to run the experiments. We have used first-improvement local search in all three methods.

Table I
THE INSTANCES WITH THE FITNESS OF THE BEST SOLUTION KNOWN, THE SIZE OF THE INSTANCES, THE AMOUNT OF EVALUATIONS FOR EACH RUN AND THE NUMBER OF RUNS FOR EACH EXPERIMENT.

Instance	*Size* n	*Known Min.*	*Evaluations*	*Runs*
nug30	30	6,124	40,000,000	50
ste36a	36	9,526	100,000,000	50
tai60a	60	7,205,962	300,000,000	50
tai80a	80	13,499,184	300,000,000	50
sko100a	100	152,002	300,000,000	50

We note that on the two smaller instances shown in Table I, BIMA always found the best known optimum [23]. However, these problems turn out to have multiple global minima. Therefore, on these two problems we will compare the methods based on how many different global optima they find in a run. This experiment will therefore show the explorative power of the three methods. For the three larger instances BIMA, ILS, and MLS could not always stretch down to the known minimum in the specified amount of iterations. Therefore, the behavior of the methods on these instances is compared focusing on how close - on average - they get to the best known minimum, within the allowed execution boundaries.

BIMA uses four parameters, which we selected based on preliminary experiments and which we kept constant for the five QAP instances. The used parameters for BIMA are: the population size ps is set to 70. The number of assignments that are each time selected and enforced, ms, is set to $1/6 \cdot n$, where n is the size of the problem (e.g. 80 for tai80a). The value for c in cUCB is simply set to 2 (although we will show results with different values for c for the two smaller QAP instances). The value for w_1 used for combining the lowest fitness of all solutions in which an assignment belonged to and the average fitness, is set to 0.5. ILS has one single parameter, m which was each macro-iteration set randomly between a value of 3 and $1/3 \cdot n$, as in [24], [25]. MLS does not require any parameters.

B. Experimental Results on the Smaller QAP Instances

We are also interested in the influence of the c parameter on BIMA's effectiveness to find different global minima. When c is large, the exploration power of the method increases, but this may be at the cost of exploiting previously found good solutions less well. The results on *nug30* for BIMA with different values of c and ILS and MLS can be found in Table II. The table shows how many different global minima are found by the three methods. The results show that unlike MLS and ILS, BIMA always finds a global minimum, and even finds multiple of them. MLS finds a global minimum in less than 50% of the runs and is therefore also outperformed by ILS.

Table II
THIS TABLE COMPARES THE AVERAGE AMOUNT OF MINIMA FOUND ON NUG30 BY EACH RUN OF BIMA (WITH VARYING C), MLS AND ILS.

Method	*c*	*Average # Minima*	*Standard Error*
BIMA	0	2.32	0.15
	1	3.20	0.12
	2	3.42	0.08
	10	3.80	0.06
	100	3.76	0.06
MLS	-	0.40	0.09
ILS	-	0.91	0.14

Figure 1 shows the average number of global minima found in *nug30* during an entire run when different c-values are used in BIMA. The figure shows that the number of found global minima in general increases with c, although the best value we tested is $c = 10$.

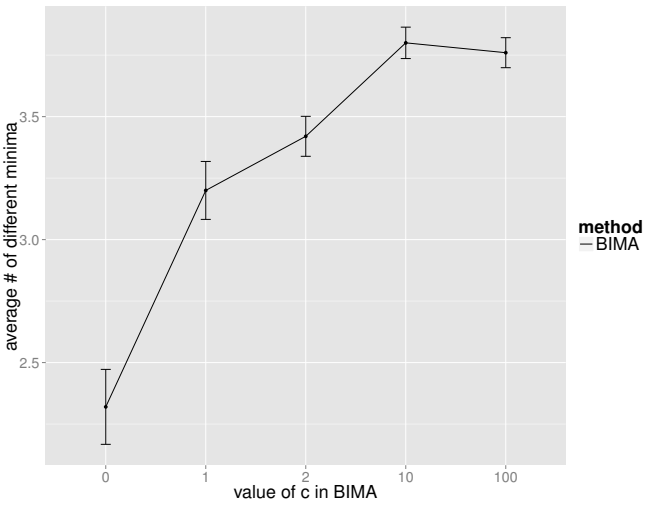

Figure 1. This figure shows how the number of found global minima by BIMA depends on the used value for c for nug30. The experiments were repeated 50 times.

The results for *ste36a* can be found in Table III. Again the results show that unlike MLS and ILS, BIMA always finds a global minimum, and finds multiple of them. MLS only rarely finds one global minima and is again the worst method.

Table III
THIS TABLE COMPARES THE AVERAGE AMOUNT OF MINIMA FOUND ON STE36A BY EACH RUN OF BIMA (WITH VARYING C), MLS AND ILS.

Method	*c*	*Average # Minima*	*Standard Error*
BIMA	0	5.34	0.23
	1	6.42	0.22
	2	6.04	0.21
	10	6.28	0.23
	100	5.62	0.19
MLS	-	0.04	0.03
ILS	-	0.66	0.07

Figure 2 shows the average number of global minima found on *ste36a* during an entire run when different c-values are used in BIMA. The figure shows that for this larger problem values of c between 1 and 10 work best. The use of a very large value, $c = 100$, can lead to too much exploration and too little exploitation.

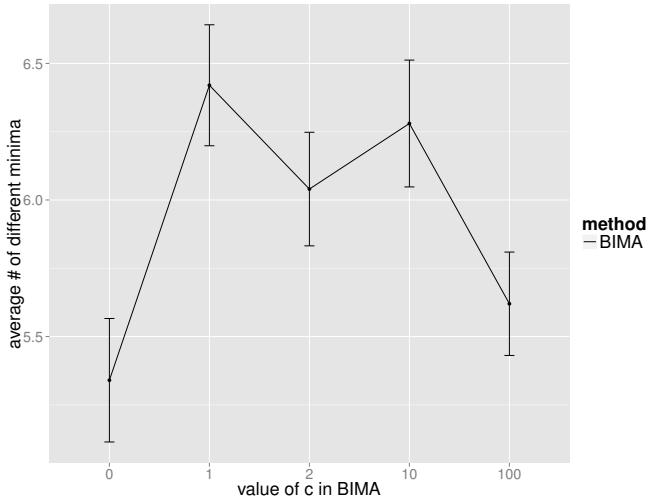

Figure 2. This figure shows how the number of found global minima by BIMA depends on the used value for c for ste36a. The experiments were repeated 50 times.

C. Experimental Results on the Larger QAP Instances

The results for *tai60a* can be found in Table IV. The minimum would be reached if the score would be 100.0, so it is clear that within the allowed search time, none of the three methods finds the optimum. However, the best found solution of the three methods was found by BIMA. Furthermore, if we compare the means of the best solutions found in the 50 runs using a student t-test, BIMA significantly ($p < 0.01$) outperforms ILS and MLS. On its turn, ILS significantly outperforms MLS.

In Figure 3, the optimization process is shown. It shows that BIMA immediately finds better solutions than ILS and MLS, and is able to improve the best found solution faster

Table IV
THE BEST AND AVERAGE BEST SOLUTION FOR TAI60A, EXPRESSED IN PERCENTAGE OF THE KNOWN MINIMUM. THE LAST ROW REPRESENTS THE STANDARD ERROR.

	BIMA	MLS	ILS
Best Solution	100.859	102.056	100.918
Average Best	101.125	102.475	101.482
Standard Error	0.019	0.036	0.026

than the other methods. The figure also shows that at the end BIMA is still improving, so it is probable that with much more evaluations, the optimum will be found.

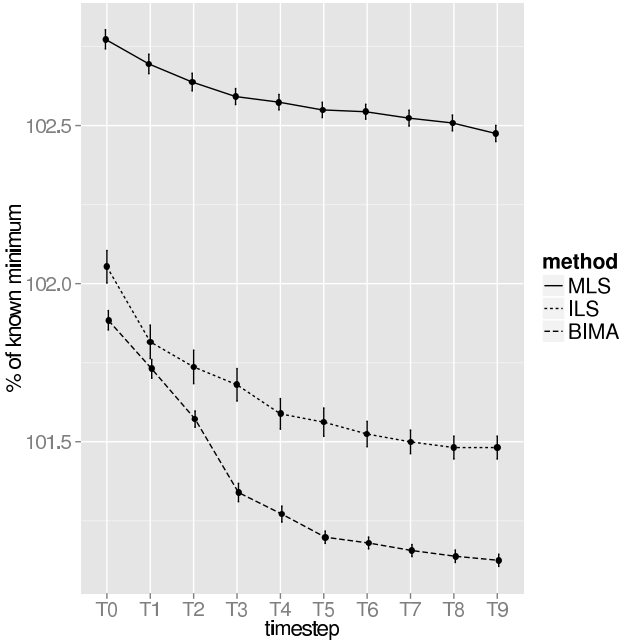

Figure 3. This figure shows how the average fitness of the best solution found improves over time for tai60a. We plotted the results after each 30,000,000 evaluations (steps T0 until T9). The experiments were repeated 50 times.

The results for *tai80a* can be found in Table V. The best found solution of the three methods was found by BIMA. If we compare the means using a student t-test, BIMA significantly ($p < 0.01$) outperforms ILS and MLS. Again, ILS significantly outperforms MLS.

Table V
THE BEST AND AVERAGE BEST SOLUTION FOR TAI80A, EXPRESSED IN PERCENTAGE OF THE KNOWN MINIMUM. THE LAST ROW REPRESENTS THE STANDARD ERROR.

	BIMA	MLS	ILS
Best Solution	101.191	102.114	101.704
Average Best	101.586	102.581	102.182
Standard Error	0.023	0.029	0.016

In Figure 4, the optimization process is shown on *tai80a*. The figure shows that BIMA again finds better solutions faster than ILS and MLS, and is able to improve the best found

solution faster than the other methods. At the end BIMA is still improving a lot.

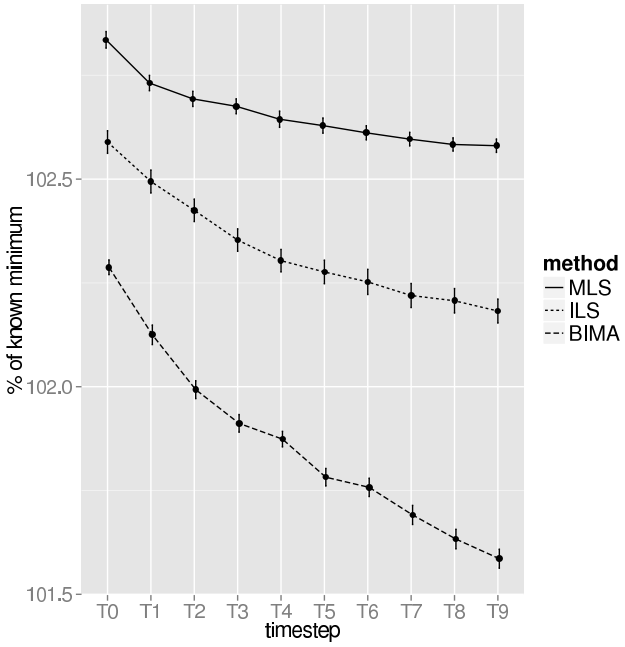

Figure 4. This figure shows how the average fitness of the best solution found improves over time for tai80a. We plotted the results after each 30,000,000 evaluations (steps T0 until T9). The experiments were repeated 50 times.

The results for *sko100a* can be found in Table VI. The best found solution of the three methods was again found by BIMA, which for this problem found a solution very close to the best optimum known. If we compare the means using a student t-test, BIMA significantly ($p < 0.01$) outperforms ILS and MLS. Again, ILS significantly outperforms MLS.

Table VI
THE BEST AND AVERAGE BEST SOLUTION FOR SKO100A, EXPRESSED IN PERCENTAGE OF THE KNOWN MINIMUM. THE LAST ROW REPRESENTS THE STANDARD ERROR.

	BIMA	MLS	ILS
Best Solution	100.092	100.412	100.150
Average Best	100.189	100.589	100.305
Standard Error	0.005	0.015	0.012

In Figure 5, the optimization process is shown on *sko100a*. The figure shows that BIMA again finds better solutions faster than ILS and MLS, and is able to improve the best found solution faster than the other methods. Again, at the end BIMA is still improving.

VI. CONCLUSIONS AND FUTURE WORK

In this paper the novel BIMA algorithm was described. BIMA is a memetic algorithm combining a population of solutions and a local search algorithm. BIMA is different from other memetic algorithms in the way it generates novel solutions in its global search. BIMA combines a multi-armed bandit algorithm and information stored in local fitness and pull-count matrices to select promising explorative assignments for a specific individual in the solution pool. These assignments are then enforced on a solution and a valid solution is made by using local swaps between new assignments and previous ones in the solution. Finally, BIMA uses local search to bring the newly generated solution to a local optimum.

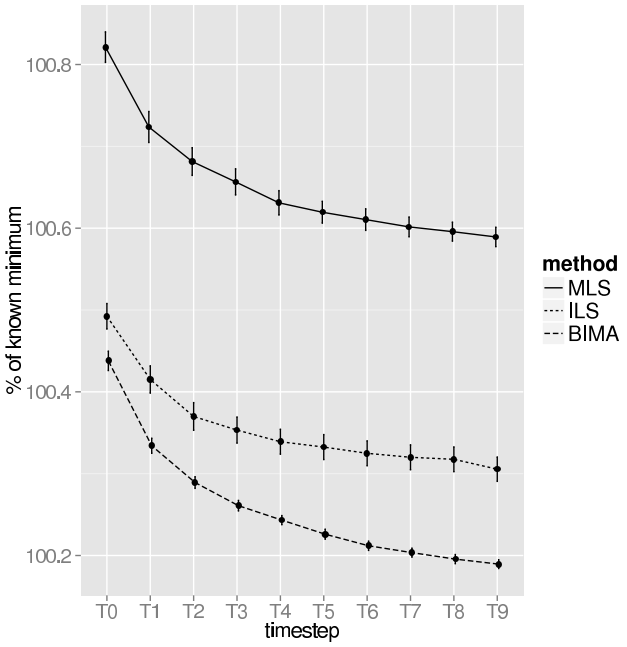

Figure 5. This figure shows how the average fitness of the best solution found improves over time for sko100a. We plotted the results after each 30,000,000 evaluations (steps T0 until T9). The experiments were repeated 50 times.

The results showed that BIMA is very fast in finding good solutions for difficult QAP instances and is able to keep on improving its best found solution during the entire run. BIMA significantly outperformed MLS and ILS on three hard QAP instances. On two smaller QAP instances, the results showed that BIMA has much better explorative capabilities than MLS and ILS, and it is able to find multiple global minima. Therefore, the power of BIMA is in its effective way of handling the exploration/exploitation trade-off. This was also our main goal when we developed the algorithm.

In future work we want to run longer experiments with BIMA to see whether and how fast it finds the best known minima. Furthermore, we want to integrate ideas from linkage learning in the BIMA framework, since currently assignments are selected without considering other assignments already in a solution. Using higher order information in the local assignment matrices, it may be possible to develop an even better combinatorial optimization algorithm. Finally, we want to extend the BIMA algorithm to other combinatorial optimization problems and compare BIMA to the best known algorithms in literature.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial*

Intelligence. Cambridge, MA, USA: MIT Press, 1992.
[2] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.
[3] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
[4] M. Dorigo, "Optimization, Learning and Natural Algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.
[5] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.
[6] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, 1989.
[7] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, Nov. 2000.
[8] M. Beckman and T. Koopmans, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, pp. 53–76, 1957.
[9] E. Cela, *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer Academic Publishers, 1998.
[10] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM*, vol. 23, no. 3, pp. 555–565, Jul. 1976.
[11] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, May 2002.
[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.
[13] M. Wiering and M. van Otterlo, *Reinforcement Learning: State of the Art*. Springer Verlag, 2012.
[14] D. Thierens, "Adaptive strategies for operator allocation," in *Parameter Setting in Evolutionary Algorithms*, 2007, pp. 77–90.
[15] M. M. Drugan and D. Thierens, "Generalized adaptive pursuit algorithm for genetic Pareto local search algorithms," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2011, pp. 1963–1970.
[16] T. Stützle, "Iterated local search for the quadratic assignment problem." *European Journal of Operational Research*, vol. 174, no. 3, pp. 1519–1539, 2006.
[17] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. A. Castellanos, and A. Doucet, "A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot," *Autonomous Robots*, pp. 93–103, 2009.
[18] H. Robbins, "Some Aspects of the Sequential Design of Experiments," in *Bulletin of the American Mathematical Society*, vol. 58, 1951, pp. 527–535.
[19] L. Gambardella, É. Taillard, and M.Dorigo, "Ant colonies for the quadratic assignment problem," *Journal of the Operational Research Society*, vol. 50, pp. 167–176, 1999.
[20] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Pittsburgh, PA, USA, Tech. Rep., 1994.
[21] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem," in *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithm*. Springer-Verlag, 2006, pp. 281–292.
[22] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "BOA: The bayesian optimization algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 1. Orlando, Florida, USA: Morgan Kaufmann, 1999, pp. 525–532.
[23] F. Puglierin, "A bandit–inspired memetic algorithm for quadratic assignment problems," 2012, unpublished master's thesis, Utrecht University.
[24] M. M. Drugan and D. Thierens, "Path-guided mutation for stochastic Pareto local search algorithms," in *Parallel problem solving from Nature (PPSN)*, vol. LNCS. Springer, 2010, pp. 485–495.
[25] ——, "Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies," *Journal of Heuristics*, vol. 18, no. 5, pp. 727–766, 2012.