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 prob-

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-

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 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**

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

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

encoding the desirability of each assignment $\chi_{i,j}$ in time-step t. These assignment matrices store the desirability of

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

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

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.

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

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.

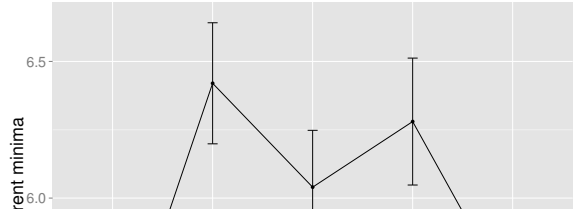
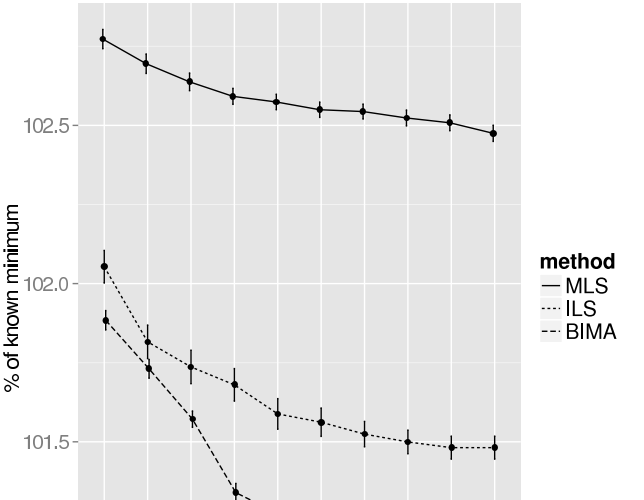


Table IV
THE BEST AND AVERAGE BEST SOLUTION FOR TAI60A, EXPRESSED IN PERCENTAGE OF THE KNOWN MINIMUM. THE LAST ROW REPRESENTS THE STANDARD ERROR.

	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.



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

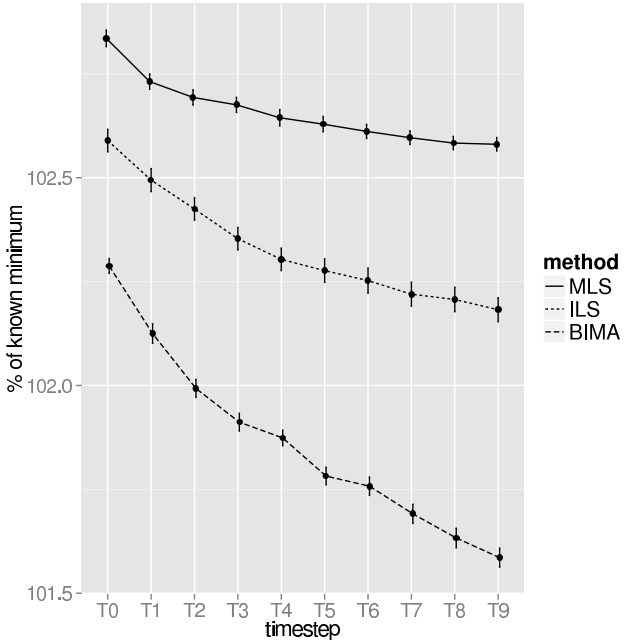


Figure 4. This figure shows how the average fitness of the best solution found improves over time for tai80a. We plotted the results after each 30,000,000 evaluations (steps T0 until T9). The experiments were repeated 50 times.

The results for *sko100a* can be found in Table VI. The best found solution of the three methods was again found by BIMA

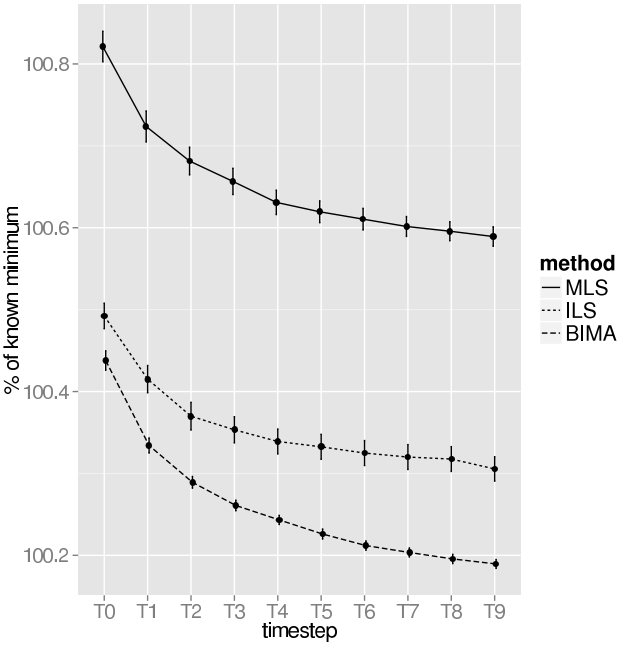


Figure 5. This figure shows how the average fitness of the best solution found improves over time for sko100a. We plotted the results after each 30,000,000 evaluations (steps T0 until T9). The experiments were repeated 50 times.

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

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.