

Parallel Ant Colonies for Combinatorial Optimization Problems

El-ghazali Talbi ^{*}, Olivier Roux, Cyril Fonlupt, Denis Robillard ^{**}

Abstract. Ant Colonies (AC) optimization take inspiration from the behavior of real ant colonies to solve optimization problems. This paper presents a parallel model for ant colonies to solve the quadratic assignment problem (QAP). Parallelism demonstrates that cooperation between communicating agents improve the obtained results in solving the QAP. It demonstrates also that high-performance computing is feasible to solve large optimization problems.

1 Introduction

Many interesting combinatorial optimization problems are NP-hard, and then they cannot be solved exactly. Consequently, heuristics must be used to solve real-world problems within a reasonable amount of time. There has been a recent interest in the field of the Ant Colony Optimization (ACO). The basic idea is to imitate the cooperative behavior of ant colonies in order to solve combinatorial optimization problems within a reasonable amount of time. Ant Colonies (AC) is a general purpose heuristic (meta-heuristic) that has been proposed by Dorigo [1]. AC has achieved widespread success in solving different optimization problems (traveling salesman [2], quadratic assignment [3], vehicle routing [4], job-shop scheduling [5], telecommunication routing [6], etc.).

Our aim is to develop parallel models for ACs to solve large combinatorial optimization problems. The parallel AC algorithm has been combined with a local search method based on tabu search. The testbed optimization problem we used is the quadratic assignment problem (QAP), one of the hardest among the NP-hard combinatorial optimization problems.

2 Ant colonies for combinatorial optimization

The ants based algorithms have been introduced with Marco Dorigo's PhD [1]. They are based on the principle that using very simple communication mechanisms, an ant is able to find the shortest path between two points. During the trip a chemical trail (pheromone) is left on the ground. The role of this trail is to guide the other ants towards the target point. For one ant, the path is chosen according to the quantity of pheromone. Furthermore, this chemical substance has a decreasing action over time, and the quantity left by one ant depends on the amount of food found and the

^{*} LIFL URA-369 CNRS / Université de Lille 1, Bât.M3 59655 Villeneuve d'Ascq Cedex FRANCE. E-mail: talbi@lifl.fr

^{**} LIL, University of Littoral, Calais, France. E-mail: fonlupt@lifl.fr

number of ants using this trail. As indicated in Fig.1, when facing an obstacle, there is an equal probability for every ant to choose the left or right path. As the left trail is shorter than the right one, the left will end up with higher level of pheromone. The more the ants will take the left path, the higher the pheromone trail is. This fact will be increased by the evaporation stage.

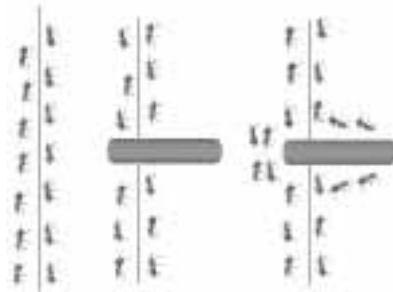


Fig. 1. Ants facing an obstacle

This principle of communicating ants has been used as a framework for solving combinatorial optimization problems. Figure 2 presents the generic ant algorithm. The first step consists mainly in the initialization of the pheromone trail. In the iteration step, each ant constructs a complete solution to the problem according to a probabilistic state transition rule. The state transition rule depends mainly on the state of the pheromone. Once all ants generate a solution, a global pheromone updating rule is applied in two phases: an evaporation phase where a fraction of the pheromone evaporates, and a reinforcement phase where each ant deposits an amount of pheromone which is proportional to the fitness of its solution. This process is iterated until a stopping criteria.

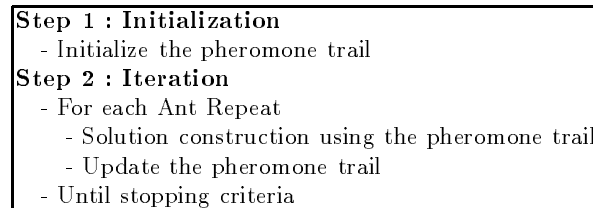


Fig. 2. A generic ant algorithm.

3 Ant colonies for the quadratic assignment problem

The AC algorithm has been used to solve the quadratic assignment problem (QAP). The QAP represents an important class of NP-hard combinatorial optimization prob-

lems with many applications in different domains (facility location, data analysis, task scheduling, image synthesis, etc.).

3.1 The quadratic assignment problem

The QAP can be defined as follows. Given a set of n objects $O = \{O_1, O_2, \dots, O_n\}$, a set of n locations $L = \{L_1, L_2, \dots, L_n\}$, a flow matrix C , where each element c_{ij} denotes a flow cost between the objects O_i and O_j , a distance matrix D , where each element d_{kl} denotes a distance between location L_k and L_l , find an object-location bijective mapping $M : O \rightarrow L$, which minimizes the objective function f :

$$f = \sum_{i=1}^n \sum_{j=1}^n c_{ij} \cdot d_{M(i)M(j)}$$

3.2 Application to the QAP

Our method is based on an hybridization of the ant system with a local search method, each ant being associated with an integer permutation. Modifications based on the pheromone trail are then applied to each permutation. The solutions (ants) found so far are then optimized using a local search method; update of the pheromone trail simulates the evaporation and takes into account the solutions produced in the search strategy. In some way, the pheromone matrix can be seen as shared memory holding the assignments of the best found solutions. The different steps of the ANTabu algorithm are detailed below (fig.3):

```

Generate  $m$  (number of ants) permutations  $\pi^k$  of size  $n$ 
Initialization of the pheromone matrix  $F$ 
FOR  $i = 1$  to  $I^{max}$  ( $I^{max}=n/2$ )
  FOR all permutations  $\pi^k$  ( $1 \leq k \leq m$ )
    Solution construction :
     $\hat{\pi}^k = n/3$  transformations of  $\pi^k$  based on the pheromone matrix
    Local search :
     $\tilde{\pi}^k = \text{tabu search}(\hat{\pi}^k)$ 
    IF  $\tilde{\pi}^k < \pi^*$ 
      THEN the best solution found  $\pi^* = \tilde{\pi}^k$ 
    Update of the pheromone matrix
    IF  $n/2$  iterations applied without amelioration of  $\pi^*$ 
      THEN Diversification

```

Fig. 3. ANTabu: Ant colonies algorithm for the QAP.

Initialization: We have used a representation which is based on a permutation of n integers:

$$s = (l_1, l_2, \dots, l_n)$$

where l_i denotes the location of the object O_i . The initial solution for each ant is initialized randomly. Three phases are necessary to initialize the matrix of pheromones. First, we apply a local search optimization of the m initial solutions. Then, we identify the best solution of the population π^* . Finally, the matrix of pheromones F is initialized as follows:

$$\tau_{ij}^0 = 1/100 \cdot f(\pi^*), i, j \in [1..n]$$

Solution construction: The current solution of each ant is transformed function of the pheromone matrix. We use a pair exchange move as a local transformation in which two objects of a permutation are swapped. $n/3$ swapping exchanges (n is the problem size) are applied as follows: the first element r is selected randomly, and the second one s is selected with a probability 0.9 such as $\tau_{r\pi_s}^k + \tau_{s\pi_r}^k$ is maximal ($\tau_{r\pi_s}^k$ is the pheromone for the position r containing the element s where π is a solution). In the other cases, the element is selected with a probability proportional to the associated pheromone:

$$\frac{\tau_{r\pi_s}^k + \tau_{s\pi_r}^k}{\sum_{r \neq s} (\tau_{r\pi_s}^k + \tau_{s\pi_r}^k)}$$

Local search: We have designed a local search procedure based on the tabu search (TS) method [7]. To apply TS to the QAP, we must define the short-term memory to avoid cycling. The long-term memory for the intensification/diversification phase has not been used, because it was handled by the ant system. The tabu list contains pairs (i, j) of objects that cannot be exchanged (recency-based restriction). The efficiency of the algorithm depends on the choice of the size of the tabu list. Our experiments indicate that choosing a size which varies between $\frac{n}{2}$ and $\frac{3n}{2}$ gives very good results. Each TS task is initialized with a random tabu list size in the interval $\frac{n}{2}$ to $\frac{3n}{2}$. The aspiration function allows a tabu move if it generates a solution better than the best found solution.

Update of the pheromone matrix: First, we update the pheromone matrix to simulate the evaporation process, which consists in reducing the matrix values F with the following formulae:

$$\tau_{ij}^{k+1} = (1 - \alpha)\tau_{ij}^k, i, j \in [1..n].$$

where $0 < \alpha < 1$ ($\alpha = 0.1$ in our experiments). If α is close to 0 the influence of the pheromone will be efficient for a long time. However, if α is close to 1 his action will be short-lived. In a second phase, the pheromone is reinforced function of the solution found. Instead of only taking into account the best found solution for updating the pheromone matrix as it is done in the HAS-QAP algorithm [3], we have devised a new strategy where each ant adds a contribution inversely proportional to the fitness of its solution. This contribution is weakened by dividing the difference between the solution and the worst one with the best one. So far, the update formula is:

$$\tau_{i\pi(i)}^{k+1} = (1 - \alpha)\tau_{i\pi(i)}^k + \frac{\alpha}{f(\pi)} \frac{f(\pi^-) - f(\pi)}{f(\pi^*)}, i \in [1..n]$$

where $\tau_{i\pi(i)}^{k+1}$ is the pheromone trail at the $(k+1)$ th iteration associated to the element $(i, \pi(i))$, π^- is the worst solution found so far, π^* the best one, and π the current solution.

Diversification: In the case where the best solution found has not been improved in $n/2$ iterations, the diversification task is started. This diversification scheme will force the ant system to start from totally new solutions with new structures. The diversification in the HAS-QAP system consisted in re-initializing the pheromone matrix and randomly generating new solutions [3]. In our case, we have created a long-term memory called *frequency matrix*. This matrix will be used to hold the frequency of all previous assignments, and will be used when a diversification phase will be triggered. During the diversification phase, the least chosen affectations will be used to generate new solutions.

4 Parallel ant colonies

To solve efficiently large optimization problems, a parallel model of ant colonies has been developed. The programming style used is a synchronous master/workers paradigm. The master implements a central memory through which passes all communication, and that captures the global knowledge acquired during the search. The worker implements the search process. The parallel algorithm works as follows (fig.4): The pheromone matrix and the best found solution will be managed by the master. At each iteration, the master broadcasts the pheromone matrix to all the workers. Each worker handles an ant process. He receives the pheromone matrix, constructs a complete solution, applies a tabu search for this solution and send the solution found to the master. When the master receives all the solutions, he update the pheromone matrix and the best solution found, and then the process is iterated.

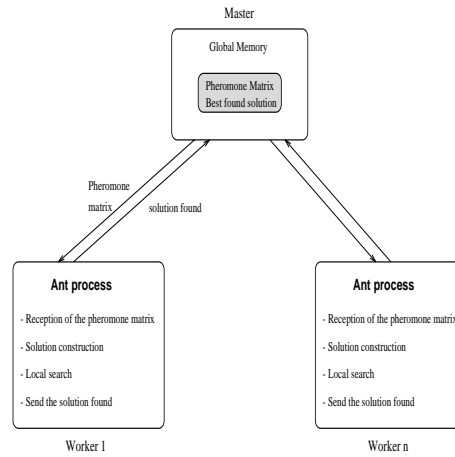


Fig. 4. Synchronous master/workers model for parallel ant colonies

Our ant system has been implemented for a parallel/distributed architecture using the programming environment C/PVM (Parallel Virtual Machine). Each ant is managed by one machine. In all the experiments, we work with only 10 ants (10 machines, network of SGIs Indy workstations).

5 Experimental Results

We first compare our metaheuristic ANTabu with the HAS-QAP method, which is also based on ant colonies. Then, we compare it with parallel independent tabu search to assess the cooperation paradigm of ant colonies. Finally, the performances of our approach is compared with other methods based on hill-climbing and genetic algorithms. For those comparisons, we studied instances from three classes of problems of the QAP-library [8]: random uniform cost and distance matrices, random cost matrix and a grid for the distance matrix, and real problems.

5.1 Comparison with HAS-QAP

The parameters for the ANTabu algorithm are set identical to HAS-QAP ones ($\alpha = 0.1$). The Tabu search method is restricted to $5n$ iterations in order to evaluate the performance of the ant system. We have selected about 20 problems from the QAPlib [8] either from the irregular class problems (**bur26d**, **chr25a**, **els19**, **tai20b**, **tai35b**) or from the regular class problems (**nug30**, **sko42**, **sko64**, **tai25a** and **wil50**). We have allowed 10 iterations for HAS-QAP and ANTabu. In Table 1, we compare the results of ANTabu and HAS-QAP. Results for HAS-QAP are directly taken from [3].

Table 1. Solution quality of HAS-QAP and ANTabu. Best results are in boldface

Problem	Best known	HAS-QAP	ANTabu
bur26b	3817852	0.106	0.018
bur26d	3821225	0.002	0.0002
chr25a	3796	15.69	0.047
els19	17212548	0.923	0
kra30a	88900	1.664	0.208
tai20b	122455319	0.243	0
tai35b	283315445	0.343	0.1333
nug30a	6124	0.565	0.029
sko42	15812	0.654	0.076
sko64	48498	0.504	0.156
tai25a	1167256	2.527	0.843
wil50	48816	0.211	0.066

It is clear that ANTabu outperforms HAS-QAP, but we may wonder if the gain is not only due to the tabu search method. We address this problem in the next section.

5.2 Impact of the cooperation between agents

In order to evaluate the gain brought by the ants cooperation system, we have compared the results with those of the PATS algorithm [9]. The PATS is a parallel adaptive tabu search, and it consists in a set of independent tabu algorithms running in a distributed fashion on a network of heterogeneous workstations (including Intel PCs, Sun and Alpha workstations).

Results are shown in Tab. 2. Best found for PATS and ANTabu is the best solution out of 10 runs. The difference relative to the QAPlib optimum is given as a percentage gap. Running times are in minutes, and correspond to the mean execution time over 10 runs on a network of 126 machines for PATS, and on a network of 10 machines for ANTabu. Thus, the ANTabu algorithm was at a great disadvantage in this regard.

Table 2. Comparison between PATS and ANTabu algorithms.

Best results are in boldface.

	tai100a	sko100a	sko100b	sko100c	sko100d	sko100e	wil100	esc128	tai256c
Best known	21125314	152002	153890	147862	149576	149150	273038	64	44759294
PATS									
Best found	21193246	152036	153914	147862	149610	149170	273074	64	44810866
Gap	0.322	0.022	0.016	0	0.022	0.013	0.013	0	0.115
Time (mn)	117	142	155	132	152	124	389	230	593
ANTabu									
Best found	21184062	152002	153890	147862	149578	149150	273054	64	44797100
Gap	0.278	0	0	0	0.001	0	0.006	0	0.085
Time (mn)	139	137	139	137	201	139	478	258	741

Results show that the ANTabu system finds better results using less computing resources and less search agents. Notice that the best known solutions have been found in three out of four sko100 instances.

5.3 Comparison with other algorithms

The performances of the ANTabu algorithm have been compared with other meta-heuristics on 28 problems:

- ant colonies: FANT [10], HAS-QAP [3].
- genetic algorithms: GDH [10], GTSH [11].
- hill-climbing heuristics: VNS-QAP and RVNS-QAP [12].

The algorithms are run 10 times to obtain an average performance estimate. The table 3 shows the average ecart-type value of the solutions obtained for instances of sizes between 19 and 80. The results of the other algorithms are obtained from [10].

Our algorithm ANTabu always succeed in finding better solutions than other algorithms. This result shows the efficiency and the robustness of the algorithm. The best known solutions are always found for 9 instances. According to the results, we observe that the algorithm is well fitted to a large number of instances.

Table 3. Comparison of algorithms with long execution times

(execution equivalent to 100 calls of the local search procedure).

Problem	Best known value	ANTabu	FANT	HAS-QAP	GDH	GTSH	VNS-QAP	RVNS-QAP
bur26a	5426670	0.0169	0.0449	0.0275	0.0581	0.0903	0.055	0.198
bur26b	3817852	0.0343	0.0717	0.1065	0.0571	0.1401	0.053	0.346
bur26c	5426795	0.0000	0.0000	0.0090	0.0021	0.0112	0.001	0.347
bur26d	3821225	0.0001	0.0022	0.0017	0.0039	0.0059	0.003	0.223
bur26e	5386879	0.0000	0.0068	0.0039	0.0059	0.0066	0.005	0.121
bur26f	3782044	0.0000	0.0007	0.0000	0.0034	0.0097	0.001	0.357
bur26g	10117172	0.0000	0.0025	0.0002	0.0027	0.0094	0.003	0.153
bur26h	7098658	0.0000	0.0003	0.0007	0.0021	0.0063	0.001	0.305
els19	17212548	0.0000	0.5148	0.9225	1.6507	2.2383	0.421	14.056
kra30a	88900	0.0000	2.4623	1.6637	2.9269	2.0070	1.660	5.139
kra30b	91420	0.0328	0.6607	0.5043	1.1748	0.6082	0.677	4.137
nug20	2570	0.0000	0.6226	0.1556	0.4825	0.1774	0.638	2.903
nug30	6124	0.0065	0.6172	0.5650	1.1920	0.3971	0.686	2.972
sko42	15812	0.0089	0.8158	0.6539	1.2168	0.6964	0.971	3.045
sko49	23386	0.0462	0.7740	0.6611	1.4530	0.4936	0.547	2.518
sko56	34458	0.0070	1.0941	0.7290	1.4272	0.5869	0.923	2.215
sko64	48498	0.0132	0.9213	0.5035	1.2739	0.6678	0.804	2.310
sko72	66256	0.1138	0.8826	0.7015	1.4224	0.7130	0.762	2.166
sko81	90998	0.0769	0.9950	0.4925	1.3218	0.7125	0.561	1.666
sko90	115534	0.1207	0.9241	0.5912	1.2518	0.7606	0.842	1.773
tai20b	122455319	0.0000	0.1810	0.2426	0.1807	1.0859	0.226	1.252
tai25b	344355646	0.0000	0.1470	0.1326	0.3454	1.9563	0.196	7.792
tai30b	637117113	0.0343	0.2550	0.2603	0.3762	3.2438	0.510	4.197
tai35b	283315445	0.0869	0.3286	0.3429	0.7067	1.5810	0.248	5.795
tai40b	637250948	0.6997	1.3401	0.2795	0.9439	3.9029	1.559	7.421
tai50b	458821517	0.2839	0.5412	0.2906	1.1281	1.6545	0.642	4.063
tai60b	608215054	0.1621	0.6699	0.3133	1.2756	2.6585	0.880	6.309
tai80b	818415043	0.4841	1.4379	1.1078	2.3015	2.7702	1.569	4.909
Mean		0.080	0.583	0.402	0.864	0.936	0.552	3.168

6 Conclusion and future work

In this paper we have proposed a powerful and robust algorithm for the QAP, based on ants colonies. Compared with previous ants systems for the QAP (HAS-QAP algorithm), we have refined the ants cooperation mechanism, both in the pheromone matrix update phase and in the exploitation/diversification phase by using a frequency matrix. The search process of each ant has also been reinforced with a local search procedure based on tabu search.

Results show a noticeable increase in performance compared to HAS-QAP and also to parallel tabu search, thus demonstrating the complementary gains brought by the combined use of a powerful local search and ants-like cooperation. This last

comparison, with the parallel tabu search algorithm, pleads for a more widely spread use of cooperation in parallel heuristics.

Future works include an application of the metaheuristic to other optimization problems (set covering, graph coloring, multi-objective optimization), and an implementation under the execution support MARS (Multi-user Adaptive Resource Scheduler) to allow efficient fault-tolerant runs on larger heterogeneous networks of workstations [13].

References

1. M. Dorigo. *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.
2. M. Dorigo, V. Maniezzo, and A. Coloni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Mans, and Cybernetics*, 1(26):29–41, 1996.
3. L. Gambardella, E. Taillard, and M. Dorigo. Ant colonies for the qap. Technical Report 97-4, IDSIA, Lugano, Switzerland, 1997.
4. B. Bullnheimer, R. F. Hartl, and C. Strauss. Applying the ant system to the vehicle routing problem. In *2nd Metaheuristics Int. Conf. MIC'97*, Sophia-Antipolis, France, July 1997.
5. A. Coloni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant system for job-shop scheduling. *JORBEL - Belgian Journal of Operations Research, Statistics, and Computer Science*, 34(1):39–53, 1994.
6. R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive behavior*, 5(2):169–207, 1997.
7. F. Glover. Tabu search - part I. *ORSA Journal of Computing*, 1(3):190–206, 1989.
8. R. E. Burkard, S. Karisch, and F. Rendl. Qaplib: A quadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 1991.
9. E. G. Talbi, Z. Hafidi, and J-M. Geib. Parallel adaptive tabu search for large optimization problems. In *2nd Metaheuristics International Conference MIC'97*, pages 137–142, Sophia-Antipolis, France, July 1997.
10. E. D. Taillard and L. Gambardella. Adaptive memories for the quadratic assignment problem. Technical Report 87-97, IDSIA, Lugano, Switzerland, 1997.
11. C. Fleurent and J. A. Ferland. Genetic hybrids for the quadratic assignment problem. *DIMACS Series in discrete Mathematics and Theoretical Computer Science*, 16:173–188, 1994.
12. P. Hansen and N. Mladenovic. An introduction to variable neighborhood search. In *2nd Metaheuristic Int. Conf.*, Sophia-Antipolis, France, 1997.
13. E-G. Talbi, J-M. Geib, Z. Hafidi, and D. Kebbal. A fault-tolerant parallel heuristic for assignment problems. In *BioSP3 Workshop on Biologically Inspired Solutions to Parallel Processing Systems, in IEEE IPSP/SPDP'98 (Int. Parallel Processing Symposium / Symposium on Parallel and Distributed Processing)*.