ELSEVIER

Future Generation Computer Systems 17 (2001) 441–449

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.nl/locate/future

Parallel Ant Colonies for the quadratic assignment problem

E.-G. Talbi [a,*], O. Roux [b], C. Fonlupt [b], D. Robillard [b]

[a] *LIFL URA-369 CNRS/Université de Lille 1, Bat.M3 59655, Villeneuve d'Ascq Cedex, France*
[b] *LIL, Université du Littoral, BP 719, Calais, France*

Abstract

Ant Colonies 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). The cooperation between simulated ants is provided by a pheromone matrix that plays the role of a global memory. The exploration of the search space is guided by the evolution of pheromones levels, while exploitation has been boosted by a tabu local search heuristic. Special care has also been taken in the design of a diversification phase, based on a frequency matrix. We give results that have been obtained on benchmarks from the QAP library. We show that they compare favorably with other algorithms dedicated for the QAP.

Keywords: Metaheuristics; Ant colonies; Tabu search; Parallel algorithm; Quadratic assignment problem; Combinatorial optimization

1. Introduction

Many heuristic methods currently used in combinatorial optimization are inspired by adaptive natural behaviors or natural systems, such as genetic algorithms, simulated annealing, neural networks, etc. Ant colonies algorithms belong to this class of biologically inspired heuristics. 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 (TS). The testbed optimization problem we used is the quadratic assignment problem (QAP), one of the hardest among the NP-hard combinatorial optimization problems.

The paper is organized as follows. First, we will introduce ant colonies for combinatorial optimization. Our ant colony based algorithm for the QAP will be detailed in Section 3. In Section 4, we present the parallel implementation of the algorithm. Finally, we give results of experiments for several standard instances from the QAP-library.

2. Ant colonies for combinatorial optimization

The ants based algorithms have been introduced with Dorigo's Ph.D. [1]. They are based on the principle that using very simple communication mechanisms, an ant group is able to find the shortest path

* Corresponding author.
E-mail addresses: talbi@lifl.fr (E.-G. Talbi), fonlupt@lifl.fr (C. Fonlupt).

0167-739X/01/$ – see front matter
PII: S0167-739X(99)00124-7

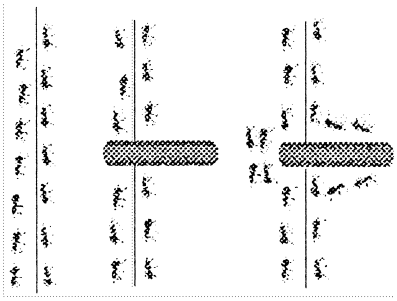

Fig. 1. Ants facing an obstacle.

between any two points. During their trips 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 number of ants using this trail. As illustrated 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 and so required less travel time, it will end up with higher level of pheromone. More the ants will take the left path, higher the pheromone trail is. This fact will be increased by the evaporation stage.

This principle of communicating ants has been used as a framework for solving combinatorial optimization problems. Fig. 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.

Step 1 : Initialization

- Initialize the pheromone trail

Step 2 : Iteration

- For each Ant Repeat
  - Solution construction using the pheromone trail
  - Update the pheromone trail
- Until stopping criteria

Fig. 2. A generic ant algorithm.

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.

3. Ant colonies for the quadratic assignment problem

The AC algorithm has been used to solve the QAP. The QAP represents an important class of NP-hard combinatorial optimization problems 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 $O = \{O_1, O_2, \ldots, O_n\}$, a set of n locations $L = \{L_1, L_2, \ldots, L_n\}$, 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:

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

3.2. Application to the quadratic assignment problem

Our method is based on a 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 given in Fig. 3.

Generate m (number of ants) permutations π^k of size n

Initialization of the pheromone matrix F

FOR $i = 1$ to I^{max} (I^{max}=$n/2$)

  FOR all permutations π^k ($1 \leq k \leq m$)

    Solution construction :

    $\hat{\pi}^k = n/3$ transformations of π^k based on the pheromone matrix

    Local search :

    $\tilde{\pi}^k$ = 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 π^*

      THEN **Diversification**

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

3.2.1. Initialization

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

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

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:

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

3.2.2. 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:

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

3.2.3. Local search

We have designed a local search procedure based on the 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 $n/2$ and $(3n)/2$ gives very good results. Each TS task is initialized with a random tabu list size in the interval $n/2$ to $(3n)/2$. The aspiration function allows a tabu move if it generates a solution better than the best found solution. In our implementation, we have restricted the TS to a limited number of iterations to avoid premature convergence.

3.2.4. 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 formula:

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

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 its 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:

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

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^{-} the worst solution found so far, π^{*} the best one, and π the current solution.

3.2.5. 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, and then we focus on unexplored areas.

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. It receives the pheromone matrix, constructs a complete solution, applies a TS for this solution, and sends the solution found and the local frequency matrix to the master. When the master receives all the solutions, it updates the pheromone, the frequency matrix, and the best solution found, and then the process is iterated. When the diversification phase is started, the master sends new generated solutions to the workers.

Our ant system has been implemented on a network of heterogeneous workstations 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 Silicon Graphics 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 TS to assess the cooperation paradigm of ant colonies. Those comparisons allow us to discuss the relative gains brought on one hand from local search, and the other hand from ants cooperation. 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. When comparing with other heuristics, the reader should notice that it is almost impossible to use the same experimental setting (e.g. machine, compiler efficiency, algorithm coding,...). Nonetheless, in order to provide a minimal guideline for comparison, we have used the same computing time, when time data are available.

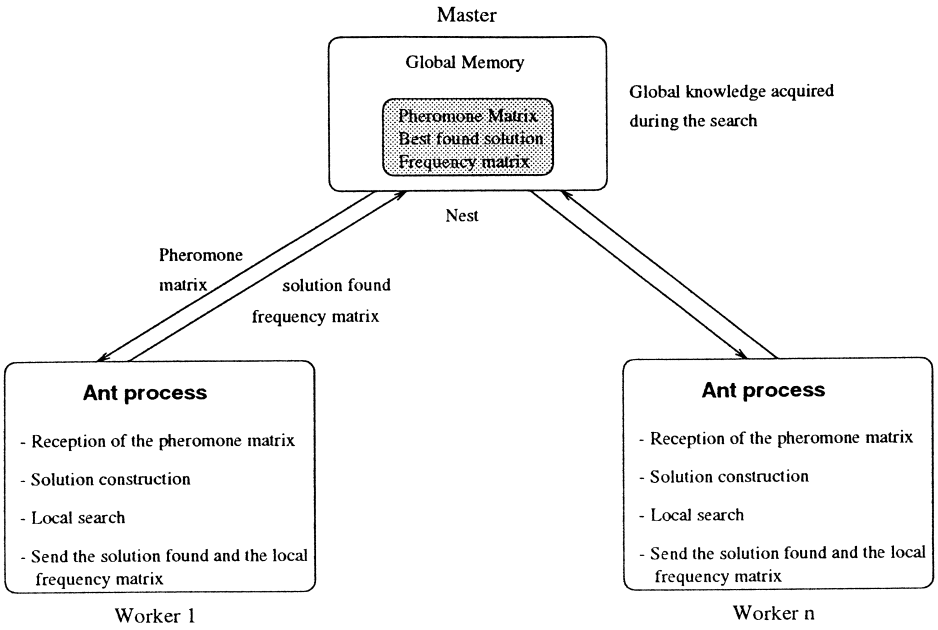

Fig. 4. Synchronous master/workers model for parallel ant colonies.

5.1. Comparison with HAS-QAP

The parameters for the ANTabu algorithm are set identical to HAS-QAP ones ($\alpha = 0.1$). The TS method is restricted to $5n$ iterations in order to evaluate the performance of the ant system. We have selected 12 problems from the QAPlib [8] either from the irregular class problems (bur26d, chr25a, els19, tai20b, tai35b) or from the regular class problems (nug30a, sko42, sko64, tai25a, 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]. The difference relatively to the best known solution of the QAP library is given as a percentage gap for both algorithms.

It is clear that ANTabu outperforms HAS-QAP, but we may wonder if the gain is not only due to the TS method. We address this problem in Section 5.2.

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**

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 TS, 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). The load of each workstation is monitored and tabu tasks are automatically migrated on idle machines from busy ones, using the MARS parallel programming environment.

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 (min)	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 (min)	139	137	139	137	201	139	478	258	741

For this comparison, we have studied large instances from three classes of problems. One instance with random uniform cost and distance matrices (tai100a), a set of instances with random cost matrix and a grid distance matrix (sko100a–e, wil100), and real-life problems (tai256c, esc128). Results are shown in Table 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.

Results show that the ANTabu system finds better results using less computing resources and less search agents (100 tabu tasks for PATS and 10 ants for ANTabu). Notice that the best known solutions have been found in four out of five sko100 instances.

5.3. Comparison with other algorithms

The performances of the ANTabu algorithm have been compared with other metaheuristics:

- the reactive tabu search (RTS) from Battiti and Tecchiolli [10],
- the TS from Taillard [11],

Table 3
Compared results on regular instances with the same computing time (best results are in boldface) [a]

Problem	Best known value	TT	RTS	SA	GH	HAS-QAP	ANTabu	Time (s)
nug20	2570	**0**	0.911	0.070	**0**	**0**	**0**	30
nug30	6124	0.032	0.872	0.121	0.007	0.098	**0**	83
sko42	15812	0.039	1.116	0.114	0.003	0.076	**0**	248
sko49	23386	0.062	0.978	0.133	0.040	0.141	**0.038**	415
sko56	34458	0.080	1.082	0.110	0.060	0.101	**0.002**	639
sko64	48498	0.064	0.861	0.095	0.092	0.129	**0.001**	974
sko72	66256	0.148	0.948	0.178	0.143	0.277	**0.074**	1415
sko81	90998	0.098	0.880	0.206	0.136	0.144	**0.048**	2041
sko90	115534	0.169	0.748	0.227	0.196	0.231	**0.105**	2825
tai20a	703482	0.211	0.246	0.716	0.268	0.675	**0**	26
tai25a	1167256	0.510	**0.345**	1.002	0.629	1.189	0.736	50
tai30a	1818146	0.340	0.286	0.907	0.439	1.311	**0.018**	87
tai35a	2422002	0.757	0.355	1.345	0.698	1.762	**0.215**	145
tai40a	3139370	1.006	0.623	1.307	0.884	1.989	**0.442**	224
tai50a	4941410	1.145	0.834	1.539	1.049	2.800	**0.781**	467
tai60a	7208572	1.270	**0.831**	1.395	1.159	3.070	0.919	820
tai80a	13557864	0.854	**0.467**	0.995	0.796	2.689	0.663	2045
wil50	48816	0.041	0.504	0.061	0.032	0.061	**0.008**	441

[a] Values are the average of gap between solution value and best known value in percent over 10 runs.

Table 4
Compared results on irregular instances with the same computing time (best results are in boldface) [a]

Problem	Best known value	TT	RTS	SA	GH	HAS-QAP	ANTabu	Time (s)
bur26a	5426670	0.0004	–	0.1411	0.0120	**0**	**0**	50
bur26b	3817852	0.0032	–	0.1828	0.0219	**0**	0.0169	50
bur26c	5426795	0.0004	–	0.0742	**0**	**0**	**0**	50
bur26d	3821225	0.0015	–	0.0056	0.0002	**0**	**0**	50
bur26e	5386879	**0**	–	0.1238	**0**	**0**	**0**	50
bur26f	3782044	0.0007	–	0.1579	**0**	**0**	**0**	50
bur26g	10117172	0.0003	–	0.1688	**0**	**0**	**0**	50
bur26h	7098658	0.0027	–	0.1268	0.0003	**0**	**0**	50
chr25a	3796	6.9652	9.8894	12.4973	2.6923	3.0822	**0.8957**	40
els19	17212548	**0**	0.0899	18.5385	**0**	**0**	**0**	20
kra30a	88900	0.4702	2.0079	1.4657	**0.1338**	0.6299	0.2677	76
kra30b	91420	0.0591	0.7121	0.1947	0.0536	0.0711	**0**	86
tai20b	122455319	**0**	–	6.7298	**0**	0.0905	**0**	27
tai25b	344355646	0.0072	–	1.1215	**0**	**0**	**0**	50
tai30b	637117113	0.0547	–	4.4075	0.0003	**0**	**0**	90
tai35b	283315445	0.1777	–	3.1746	0.1067	**0.0256**	0.0408	147
tai40b	637250948	0.2082	–	4.5646	0.2109	**0**	0.4640	240
tai50b	458821517	0.2943	–	0.8107	0.2142	**0.1916**	0.2531	480
tai60b	608215054	0.3904	–	2.1373	0.2905	**0.0483**	0.2752	855
tai80b	818415043	1.4354	–	1.4386	0.8286	**0.6670**	0.7185	2073

[a] Values are the average of gap between solution value and best known value in percent over 10 runs.

- the genetic hybrid method (GH) from Fleurent and Ferland [12],
- the simulated annealing (SA) from Connolly [13],
- the HAS-QAP, previously cited, from Gambardella et al. [3], but with a computing time extended to 100 iterations for each of the 10 ants. Again, results are taken from [3].

It has been shown in [14] that these methods do not have the same effectiveness according to whether they are applied to so-called regular or irregular instances, where regular instances have a flow dominance lower than 1.2 as opposed to irregular ones which are also sometimes called “structured” instances. Results for regular instances are shown in Table 3. Our algorithm ANTabu finds better solutions than other algorithms.

Next we have compared those algorithms on a set of 20 irregular instances ranging from 10 to 80 locations. Results are in Table 4. In this case best results are obtained by HAS-QAP, which found best average fitness for a set of 16 instances. ANTabu found the best average for only 13 instances.

In an attempt to understand the reasons behind this difference in behavior between regular and irregular instances, we have also monitored not only average results but also best and worst ones. These datas are shown in Table 5.

Table 5
Best and worst results on 10 runs on irregular instances for ANTabu [a]

Problem	Average	Time (s)	Worst	Best
bur26a	0.0000	50	0.0000	0.0000
bur26b	0.0169	50	0.1693	0.0000
bur26c	0.0000	50	0.0000	0.0000
bur26d	0.0000	50	0.0000	0.0000
bur26e	0.0000	50	0.0000	0.0000
bur26f	0.0000	50	0.0000	0.0000
bur26g	0.0000	50	0.0000	0.0000
bur26h	0.0000	50	0.0000	0.0000
chr25a	0.8957	40	4.5838	0.0000
els19	0.0000	20	0.0000	0.0000
kra30a	0.2677	76	1.3386	0.0000
kra30b	0.0000	86	0.0000	0.0000
tai20b	0.0000	27	0.0000	0.0000
tai25b	0.0000	50	0.0000	0.0000
tai30b	0.0000	91	0.0000	0.0000
tai35b	0.0408	148	0.2212	0.0000
tai40b	0.4640	241	2.6239	0.0000
tai50b	0.2531	486	0.9075	0.0000
tai60b	0.2752	860	1.5608	0.0000
tai80b	0.7185	2091	1.8549	0.0019

[a] Values are gaps between solution and best known value in percent. Notice that an optimal solution is found in all but one problems.

Notice that an optimal solution is found at least once in 10 runs for almost all problems. Thus we think that the slightly low average from Table 4 could be due to low quality local optima that are found after the diversification phase. It may well be the price to pay for a more complete exploration of the search space.

6. Conclusion and future work

In this paper we have proposed a powerful and robust algorithm for the QAP, which is based on ant colonies. Compared with previous ant 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 TS.

Results show a noticeable increase in performance compared to HAS-QAP and also to parallel independent TS, thus demonstrating the complementary gains brought by the combined use of a powerful local search, ants-like cooperation and parallelism. The comparison with the parallel TS 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 problems), and an implementation under the execution support Multi-user Adaptive Resource Scheduler (MARS) to allow efficient fault-tolerant runs on larger heterogeneous network of workstations [15]. We are also interested in investigating more closely the impact of the diversification phase, and in leading a finer study on the conditions needed for appearance of a fruitful cooperation in parallel agents systems.

References

[1] M. Dorigo, Optimization, learning and natural algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.
[2] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Mans, and Cybernetics 1 (26) (1996) 29–41.
[3] L. Gambardella, E. Taillard, M. Dorigo, Ant colonies for the QAP, Technical Report 97-4, IDSIA, Lugano, Switzerland, 1997.
[4] B. Bullnheimer, R.F. Hartl, C. Strauss, Applying the ant system to the vehicle routing problem, in: Second Metaheuristics International Conference, MIC'97, Sophia-Antipolis, France, 1997.
[5] A. Colorni, M. Dorigo, V. Maniezzo, M. Trubian, Ant system for job-shop scheduling, JORBEL-Belgian Journal of Operations Research Statistics and Computer Science 34 (1) (1994) 39–53.
[6] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, Ant-based load balancing in telecommunications networks, Adaptive Behavior 5 (2) (1997) 169–207.
[7] F. Glover, Tabu search-I, ORSA Journal of Computing 1 (3) (1989) 190–206.
[8] R.E. Burkard, S. Karisch, F. Rendl, Qaplib: a quadratic assignment problem library, European Journal of Operational Research 55 (1991) 115–119.
[9] E.G. Talbi, Z. Hafidi, J-M. Geib, Parallel adaptive tabu search for large optimization problems, in: Second Metaheuristics International Conference, MIC'97, Sophia-Antipolis, France, 1997, pp. 137–142.
[10] R. Battiti, G. Tecchiolli, The reactive tabu search, ORSA Journal on Computing 6 (1994) 126–140.
[11] E. Taillard, Robust tabu search for the quadratic assignment problem, Parallel Computing 17 (1991) 443–455.
[12] C. Fleurent, J.A. Ferland, Genetic hybrids for the quadratic assignment problem, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 16 (1994) 173–188.
[13] D.T. Connolly, An improved annealing scheme for the QAP, European Journal of Operational Research 46 (1990) 93–100.
[14] E. Taillard, Comparison of iterative searches for the quadratic assignment problem, Location Science 3 (1995) 87–103.
[15] E.-G. Talbi, J-M. Geib, Z. Hafidi, D. Kebbal, A fault-tolerant parallel heuristic for assignment problems, in: José Rolim (Ed.), BioSP3 Workshop on Biologically Inspired Solutions to Parallel Processing Systems, IEEE IPPS/SPDP'98 International Parallel Processing Symposium on Parallel and Distributed Processing, Orlando, FL, USA, 1998, Lecture Notes in Computer Science, vol. 1388, Springer, Berlin, pp. 306–314.

E.-G. Talbi received the Master's and Ph.D. degrees in Computer Science, both from the Institut National Polytechnique de Grenoble. He is presently Associate Professor in Computer Science at Université de Lille 1, and researcher in Laboratoire d'Informatique Fondamentale de Lille. He took part in several CEC Esprit and national research projects. His current research interests are in the fields of parallel optimization, evolutionary computation, and multi-criteria combinatorial optimization.

O. Roux is a Ph.D. student at the Université du littoral, Calais, France. His research interests include evolutionary algorithm using ant colonies and distributing computing. He received the master's degree (DEA) from the université de Lille 1, France.

C. Fonlupt received the Ph.D. degree in Computer Science from the Université de Lille 1, France, in 1994. He is currently an Assistant Professor of Computer Science at the Université du Littoral, Calais, France. His research interests are in the areas of evolutionary computation, ants colony and genetic programming.

D. Robilliard obtained a Ph.D. degree in Computer Science in 1996, at the Université de Lille 1, France. He is now Assistant Professor at the Université du Littoral-Côte d'Opale, France. His research interests are in the field of evolutionary computation and applications, notably to combinatorial optimization and image analysis.