Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematics and Computation 183 (2006) 427–435

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

Optimization of the quadratic assignment problem using an ant colony algorithm

Nihan Çetin Demirel [a], M. Duran Toksarı [b,*]

[a] *Industrial Engineering Department, Machine Faculty, Yildiz Technique University, Istanbul, Turkey*
[b] *Industrial Engineering Department, Erciyes University, 38039 Kayseri, Turkey*

Abstract

Ant algorithm is a multi-agent systems inspired by the behaviors of real ant colonies function to solve optimization problems. In this paper an ant colony optimization algorithm is developed to solve the quadratic assignment problem. The local search process of the algorithm is simulated annealing. In the exploration of the search space, the evaluation of pheromones which are laid on the ground by ants is used. In this work, the algorithm is analyzed by using current problems in the literature and is compared with other metaheuristics.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Ant colony optimization; Simulated annealing; Metaheuristics; Quadratic assignment problem; Combinatorial optimization

1. Introduction

Ant colony optimization (ACO) is a metaheuristic, which is inspired by adaptation of a natural system. The medium used by ants to communicate information regarding shortest paths to food, consist of pheromone trails. ACO has been initiated by Dorigo [1] which has been successfully applied to several NP-hard combinatorial optimization problems such as traveling salesman [2], quadratic assignment problem [3], job-shop scheduling [4], vehicle routing [5], telecommunication networks [6], etc.

The quadratic assignment problem (QAP) is a combinatorial optimization problem has been introduced by Koopmans and Beckman in 1957 [7]. The QAP is a NP-hard (nondeterministic polynomial time complete) optimization problem [8]; even finding a solution within a factor of $(1 + e)$ of the optimal one remains NP-hard [9]. This paper presents a new method for the QAP, this method is a hybridization of the ant colony optimization with a local search method based on simulated annealing (SA). The algorithm is analyzed by using several standard instances. The paper is organized as follows: In Section 2, ACO is described generally.

After QAP is defined in Section 3, our ACO based algorithm for the QAP will be detailed in Section 4. Finally, results of experiments are given and are compared with other metaheuristics.

[*] Corresponding author.
E-mail address: dtoksari@erciyes.edu.tr (M.D. Toksarı).

0096-3003/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2006.05.073

2. Ant colony optimization

ACO is based on the natural foraging system found in an ant colony. Ants are used very simple communication mechanism to find the shortest path between any two points. This mechanism is guided by pheromone trail that is left on the ground. The role of this trail is to guide the other ants towards the target point. The idea is that if at a given point an ant has to choose among different paths, those which were heavily chosen by preceding ants (that is, those with a high trail level) are chosen with higher probability. Furthermore high trail levels are synonymous with short paths. As illustrated in Fig. 1, if the path is cut off by an obstacle, there is an equal probability for every ant to choose the left or right path. As the right trail is shorter than the left one and so required less travel time, it will end up with higher level of pheromone. More the ants will prefer the right path because of the higher pheromone trail.

ACO algorithms for combinatorial problems follow a specific ant algorithm outlined in Fig. 2.

After the initialization of the pheromone trails and some parameters, iterations are continued. In the each iteration, firstly, each ant construct feasible solution, then the solution is improved by applying local search, and finally pheromone trails are updated. This process is iterated until a stopping criterion.

3. Applying the ACO to the QAP

A large number of metaheuristic methods like simulated annealing (SA), tabu search (TS), genetic algorithm, have been used to solve the QAP.

The QAP represents an important class of NP-hard combinatorial optimization problems. In this section, first, QAP will be defined, then a new method (AntSimulated) will be presented that is a hybridization of the ACO with a local search method based on SA.

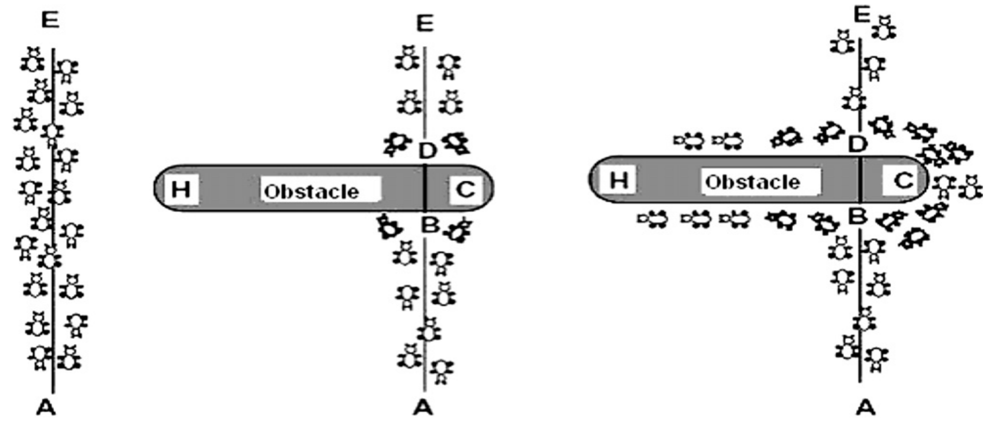

Fig. 1. How real ants find a shortest path.

Initialization
- Set parameters,
- Initialize pheromone trails

Iteration
Repeat (For each ant)
- Construct solution,
- Apply local search,
- Update trails,

Until stopping criteria

Fig. 2. General algorithmic schema for ACO algorithms.

3.1. Problem definition

The QAP can be described as the problem of assigning n number of facilities to n number of locations with given distances between the locations and given flows between the facilities. The goal then is to place the facilities on locations in such a way that the sum of the product between flows and distances is minimized [10].

More formally, given n facilities and n locations, two $n \times n$ matrices,

$A = [a_{ij}]$ where a_{ij} is the distance between locations i and j,
$B = [b_{kl}]$ where b_{kl} is the flow between facilities k and 1.

QAP can be stated as follows:

$$\min f(\pi) = \sum_{i=i}^{n} \sum_{j=1}^{n} a_{ij} b_{kl}, \tag{1}$$

$k = \pi_i$ and $l = \pi_j$ where $\Pi(n)$ is the set of permutations of n elements. Using $\Pi(n)$ QAP can be described as follows:

$$\min_{\pi \in \Pi} f(\pi) = \sum_{i=i}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi i \pi j}. \tag{2}$$

QAP instances of size larger than 20 are intractable, so a large number of metaheuristic methods for solving them have been used up to now.

3.2. Application for QAP

AntSimulated is a new method to solve QAP. It is a hybridization of the ACO with a local search method based on SA, which is presented into details Fig. 3.

```
Initialization
    Generate m initial solutions, each one associated to one ant
    Improve m initial solutions with the local search method based on SA
    Let π* be the best solution
    Initialization of the pheromone matrix F
    For i=1   to $I^{max}(I^{max}=n/2)$

      For all permutations $\pi^k (1 \le k \le m)$

      Solution manipulation
        $\hat{\pi}^k$ = T swaps of $\pi^k$ based on the pheromone matrix

        $\overset{\infty}{\pi}^k$ = Local search method based on Simulated Annealing ($\hat{\pi}^k$)

        If f($\overset{\infty}{\pi}^k$) < f(π*) Then
              The best solution π* = $\overset{\infty}{\pi}^k$

    Pheromone trail updating
        Update of the pheromone matrix

   Diversification
        If n/2 iterations have been performed without amelioration of π* then
        Diversification
```

Fig. 3. AntSimulated: the hybridization of the ACO with a local search method based on SA.

4. AntSimulated algorithm description

This section describes the four steps of the AntSimulated algorithm: Initialization, solution manipulation, pheromone trail updating and diversification.

4.1. Initialization

First, each ant is given a randomly chosen initial solution. These solutions are optimized by using local search that will be detailed later. All pheromone trails τ_{ij} are set to the same value τ_0. τ_0 is calculated as follows:

$$\tau_0 = \frac{1}{10 \cdot (f(\pi^*))}, \tag{3}$$

where n^* is the best solution found so far.

4.2. Solution manipulation

Solution manipulation has two stages: pheromone trail based modification and local search method.

4.2.1. Pheromone trail based modification

Pheromone trail based modification is applied to each ant solution (π^k) to obtain $\hat{\pi}^k$. Number of T swaps is applied in this stage. Swap is a pair exchange of two objects of a permutation. Firstly, the element r is selected randomly. Then element s is determined according to two different cases. First case: with a probability 0.9, s is selected as follows:

$$\max(\tau^k_{r\pi_s} + \tau^k_{s\pi_r}), \tag{4}$$

$\tau^k_{r\pi_s}$ is pheromone for the position r containing the element s where π is solution of ant k.

Second case, with a probability 0.1, s is selected as follows:

$$\frac{\tau^k_{r\pi_s} + \tau^k_{s\pi_r}}{\sum_{j \neq r}\left(\tau^k_{r\pi_j} + \tau^k_{j\pi_r}\right)}. \tag{5}$$

In this case a probability proportional to the values contained in the pheromone trail is used.

Finally, determined objects of permutation (r and s) are swapped.

```
STEP1
c=0; {c is a counter}
Determine the initial permutation ($\pi^k$ ; the permutation of ant k), a initial temperature ($T_0$), a last temperature ($T_s$), a cooling proportion (r);

STEP2
Create new permutation ($\pi_j$) from current permutation ($\pi_i$) by using selected neighborhood mechanism;
$\Delta$=f($\pi_j$)-f($\pi_i$)
If $\Delta \leq 0$ Then
$\pi_i$:=$\pi_j$
Else If random (0.1) < exp(- /T) Then
$\pi_i$:=$\pi_j$

STEP3
$T_{c+1}$=$T_c$/(1+r$T_c$);{$T_c$ is a temperature of iteration}
c:=c+1;
If c>K Then
Stop;
Else
Go to STEP 2.
```

Fig. 4. The local search method based on SA.

4.2.2. Local search: simulated annealing

AntSimulated algorithm has a local search procedure based on the SA method which is an optimization method suitable for combinatorial optimization problems. The base of the method of simulated annealing is an analogy with thermodynamics, specifically with the way that liquids freeze and crystallize, or metals cool and anneal. At high temperatures, the molecules of a liquid move freely with respect to one another. If the liquid is cooled slowly, thermal mobility is lost. The atoms are often able to line themselves up and form a pure crystal that is completely ordered over a distance up to billions of times the size of an individual atom in all directions. This crystal is the state of minimum energy for this system. The amazing fact is that, for slowly cooled systems, nature is able to find this minimum energy for this system. So the essence of the process is slow cooling, allowing ample time for redistribution of the atoms as they lose mobility. This is the technical definition of annealing, and it is essential for ensuring that a low energy state will be achieved [11]. This process gives inspiration to solve optimization problems. The relation between simulated annealing and combinatorial optimization can be explained as follows:

Simulated annealing		Combinatorial optimization
States of system	$\leftrightarrow$	Suitable solutions
Energy	$\leftrightarrow$	Objective function
Change of state	$\leftrightarrow$	Neighborhood solution
Temperature	$\leftrightarrow$	Control parameters
Freezing state	$\leftrightarrow$	Heuristic solution

In this paper, the used local search procedure based on the SA is shown in Fig. 4. The efficiency of the algorithm depends on the number of iterations (K), the initial temperature (T_0), the last temperature (T_s), the temperature of iteration $c(T_c)$ and the neighborhood mechanism.

The cooling proportion (r) and temperature of subsequent iteration (T_{c+1}) based on the cooling proportion (r) are formulated as follows [12]:

$$r = [(T_0 - T_c)/(I - 1)T_0T_c], \tag{6}$$

$$T_{c+1} = T_c/1 + (r \times T_c). \tag{7}$$

Design of experiment (DOE) is utilized to determine number of iterations (K), the initial temperature (T_0) and the last temperature (T_s). This work indicates that algorithm gives very good results when parameters K, T_0 and T_s were set to $K = 200$, $T_0 = 100$ and $T_s = 10$. The neighborhood of a permutation has been obtained according to one of two different randomly chosen mechanisms.

4.2.2.1. Swap mechanism. In this mechanism, two different randomly chosen elements from the current permutation are swapped. For example;

Current solution	Chosen elements	New solution
2–3–4–1–5	2 and 4	4–3–2–1–5

4.2.2.2. Optimal swap-greedy mechanism. A randomly chosen element from the current permutation is swapped with elements into different positions. Optimal swap for objective function is selected.

Current solution	Randomly chosen element	Chosen element	New solution
2–3–4–1–5	4	2	4–3–2–1–5
		3	2–4–3–1–5
		5	2–3–5–1–4
		1	2–3–1–4–5

4.3. Pheromone trail updating

Updating of the pheromone trails includes two phases which are the simulation of the evaporation process and taking into account the best three found solutions. First, pheromone trails are updated to simulate the evaporation process, which consists in reducing the value of the trails with the following formula:

$$\tau_{ij} = (1 - \alpha) \cdot \tau_{ij}, \quad 1 \leqslant i, j \leqslant n \text{ and } 0 < \alpha < 1. \tag{8}$$

If a value of α is close to 0, the influence of the pheromone will be efficient for a long time. However, if a value of α is close to 1, a degree of evaporation will be high. Then, a new strategy used to reinforce pheromone trails.

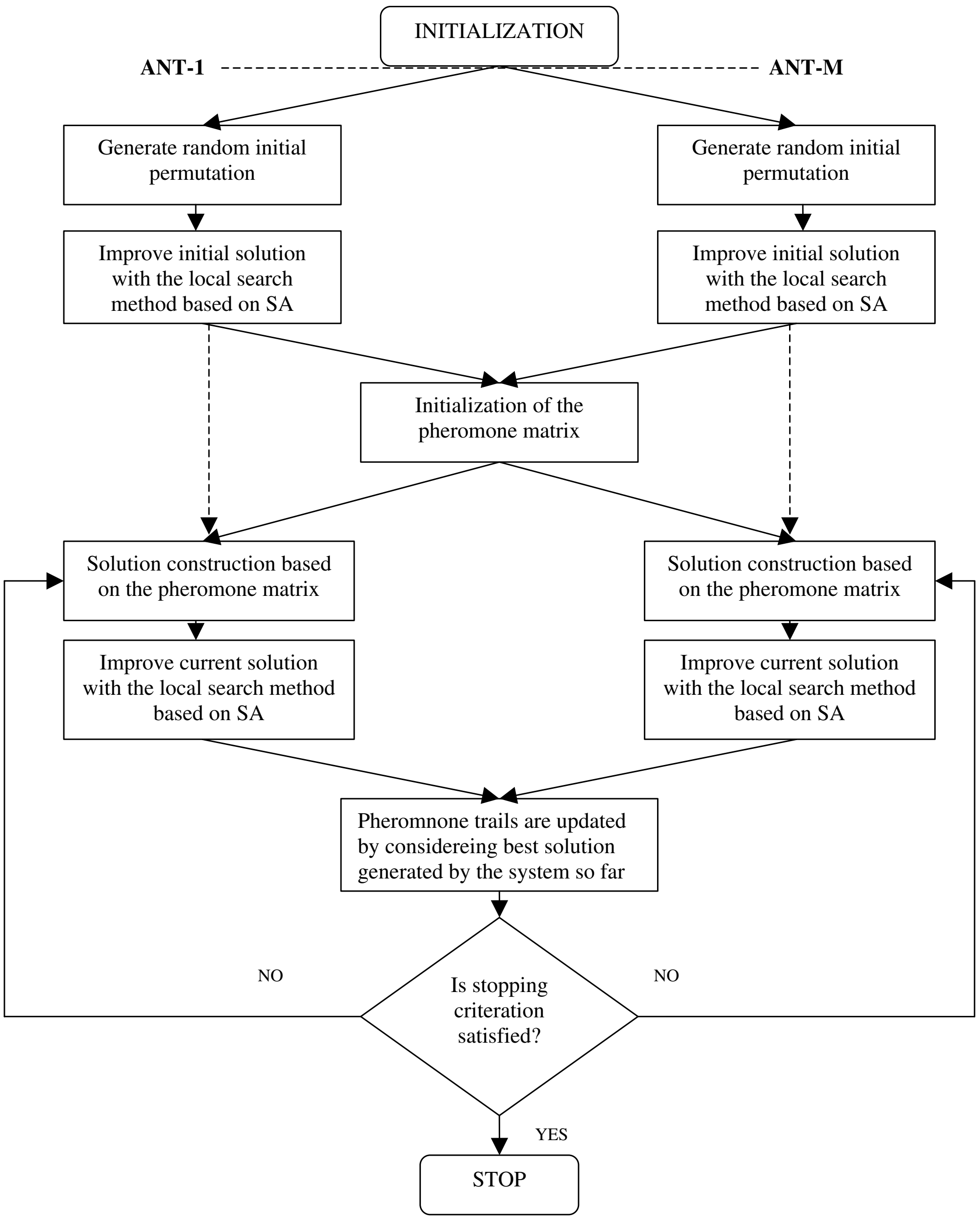
INITIALIZATION
ANT-1
ANT-M
Generate random initial permutation
Generate random initial permutation
Improve initial solution with the local search method based on SA
Improve initial solution with the local search method based on SA
Initialization of the pheromone matrix
Solution construction based on the pheromone matrix
Solution construction based on the pheromone matrix
Improve current solution with the local search method based on SA
Improve current solution with the local search method based on SA
Pheromnone trails are updated by considereing best solution generated by the system so far
NO
NO
Is stopping criteration satisfied?
YES
STOP

Fig. 5. The main features of the AntSimulated algorithm.

This strategy includes function of the best three of found solutions for updating the pheromone trails. The newness was formulated as follows:

$$\tau_{i\pi_{i^*}} = \tau_{i\pi_{i^*}} + \frac{\alpha}{f\left(\frac{\pi^* + \pi^{*-1} + \pi^{*-2}}{3}\right)}, \tag{9}$$

where π^* is the best solution found so far.

Experimental works have indicated that this new strategy is decreased by use of diversification mechanism that will be described in the next section.

4.4. Diversification

Diversification mechanism is started if the best solution found has not been improved in $n/2$ iterations. Diversification mechanism provides that algorithm works on solutions with different structure. First, the best three of found solutions are preserved before diversification is started. Then, all the information contained the pheromone trails are erased and pheromone trail matrix is re-initialized. Finally, a new current solution for all the ants is generated randomly but for three ants that use best three of found solutions. The main features of the AntSimulated algorithm are given in Fig. 5.

5. Experimental results

The performance of the AntSimulated algorithm has been compared with other metaheuristics: the reactive tabu search (RTS) from Battiti and Tecchiolli [13], the TS from Taillard [14], the genetic hybrid method (GH) from Fleurent and Ferland [15], the simulated annealing (SA) from Connolly [16], the HAS-QAP from Gambardella [3]. In this comparison, so-called regular or irregular instances have been used selected problems from the QAP-library [17]. The problems have been classified using a simple statistic known as "flow-dominance" [18]. Regular instances have flow-dominance lower than 1.2 as opposed to irregular ones. So, irregular instances are also sometimes called "structured" instances. The statistical software package MINITAB and "design of experiments" techniques were employed to find the key parameters affecting performance of AntSimulated. Design of experiments (DOE) techniques were employed to identify the main factors influencing the

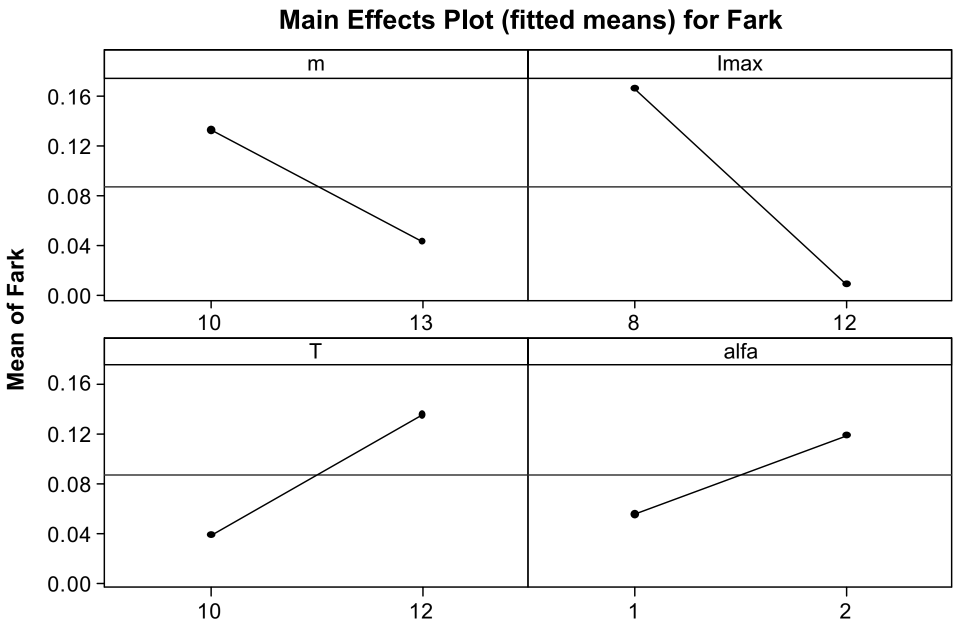

Fig. 6. Main effects plot for Sko64 problem.

solution quality. A two factorial experiment was carried out, resulting in 16 (2^4) runs per experiment for four parameters. Parameters are the number of iterations for local search procedure (I), the number of ants (m), the number of swaps (T), and α. The results from each experiment were then loaded into the statistical package MINITAB to produce mean effects plots, used to highlight the average value of the measure being studied and the effect of each parameter on that measure. Insignificant factors could then be removed from the optimization process. The algorithm is coded by C++ programming language. In the end of DOE, reasonable parameters were designated for each problem. For example, Fig. 6 shows the results of Sko64 ($m = 13$, $T = 10$, $1 = 200$ and $\alpha = 0.1$).

Table 1
Compared results on regular instances with the same computing time (best result are in italics)[a]

Problem	Best known value	TT	RTS	SA	GH	HAS-QAP	AntSimulated	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	*0.027*	415
sko56	34458	0.080	1.082	0.110	0.060	0.101	*0.001*	639
sko64	48498	0.064	0.861	0.095	0.092	0.129	*0.041*	974
sko72	66256	0.148	0.948	0.178	*0.143*	0.277	0.221	1415
sko8l	90998	0.098	0.880	0.206	0.136	0.144	*0.077*	2041
sko90	115534	0.169	0.748	0.227	0.196	0.231	*0.136*	2825
tai20a	703482	*0.211*	0.246	0.716	0.268	0.675	0.255	26
tai25a	1167256	0.510	0.345	1.002	0.629	1.189	*0.098*	50
tai30a	1818146	0.340	*0.286*	0.907	0.439	1.311	0.454	87
tai35a	2422002	0.575	0.355	1.345	0.698	1.762	*0.440*	145
tai40a	3139370	1.006	0.623	1.307	0.884	1.989	*0.476*	224
tai50a	4941410	1.145	0.834	1.539	1.049	2.800	*0.520*	467
tai60a	7208572	1.270	*0.831*	1.395	1.159	3.070	0.964	820
tai80a	13557864	0.854	0.467	0.995	0.796	2.689	*0.362*	2045
wil50	48816	*0.041*	0.504	0.061	0.032	0.061	0.088	441

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

Table 2
Compared results on irregular instances with the same computing time (best result are in italics)[a]

Problem	Best known value	TT	RTS	SA	GH	HAS-QAP	AntSimulated	Time (s)
bur26a	5426670	0.0004	–	0.1411	0.0120	*0*	*0*	50
bur26b	3817852	0.0032	–	0.1828	0.0219	*0*	*0*	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	1.2448	40
els 19	17212548	*0*	0.0899	18.5385	*0*	*0*	*0*	20
kra30a	88900	0.4702	2.0079	1.4657	0.1338	0.6299	0.7764	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	0.003	*0*	*0*	90
tai35b	283315445	0.1777	–	3.1746	0.1067	*0.0256*	0.0376	147
tai40b	637250948	0.2082	–	4.5646	0.2109	*0*	0.4872	240
tai50b	458821517	0.2943	–	0.8107	0.2142	*0.1916*	0.2475	480
tai60b	608215054	0.3904	–	2.1373	0.2905	*0.0483*	0.2258	855
tai80b	818415043	1.4354	–	1.4386	0.8286	*0.6670*	0.8214	2073

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

Ten iterations have been allowed for AntSimulated and HAS-QAP. Table 1 shows the results for regular instances. AntSimulated algorithm have been compared other algorithms AntSimulated algorithm finds better solutions than those algorithms. When AntSimulated algorithm found the best average for 13 instances, HAS-QAP algorithm obtained the best average for only one instance.

Irregular instances are shown in Table 2 (a set of 20 irregular instances ranging from 19 to 80 locations). AntSimulated algorithm found the best average for 14 instances. In this case best results are obtained by HAS-QAP, which is the best average fitness for a set of 16 instances.

6. Conclusion

This paper has proposed a powerful algorithm (AntSimulated) for the QAP, which is based on the ant colonies. The search process of each ant has been reinforced with a local search procedure based on SA. AntSimulated has a new strategy to reinforce pheromone trails. This strategy includes function of the best three of found solutions for updating the pheromone trails.

Comparisons with some of the best heuristics for the QAP have shown that AntSimulated is among the best as far as irregular and regular problems are concerned. Especially, results show a noticeable increase in performance compared with previous ant systems for the QAP (HAS-QAP algorithm). Thus demonstrating the complementary gains brought by the combined use of a powerful local search.

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).
[3] L. Gambardella, E. Taillard, M. Dorigo, Ant colonies for the QAP, Technical Report 97-4, IDSIA, Lugano, Switzerland, 1997.
[4] 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.
[5] 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.
[6] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, Ant-based load balancing in telecommunications networks, Adaptive Behavior 5 (2) (1997) 169–207.
[7] T.C. Koopmans, M.J. Beckmann, Assignment problems and the location of economics activities, Econometrica 25 (1957) 53–76.
[8] S. Shani, T. Gonzalez, P-complete approximation problems, Journal of the ACM 23 (3) (1976) 555–565.
[9] M. Queyranne, Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems, Operations Research Letters 4 (1986) 231–234.
[10] T. Stutzle, M. Dorigo, AGO algorithms for the quadratic assignment problem, New Ideas in Optimization, McGraw-Hill, 1999.
[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Numerical Recipes in C, second ed., Cambridge University Press, 1992, ISBN 0-521-43108-5.
[12] M. Lundy, A. Mees, Convergence of an annealing algorithm, Mathematical Programming 34 (1986) 111–124.
[13] R. Battiti, G. Tecchiolli, The reactive tabu search, ORSA Journal on Computing 6 (1994) 126–140.
[14] E. Taillard, Robust tabu search for the quadratic assignment problem, Parallel Computing 17 (1991) 443–455.
[15] 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.
[16] D.T. Connolly, An improved annealing scheme for the QAP, European Journal of Operational Research 46 (1990) 93–100.
[17] R.E. Burkard, S. Karisch, F. Rendl, Qaplib: a quadratic assignment problem library, European Journal of Operational Research 55 (1991) 115–119.
[18] R.S. Liggett, The quadratic assignment problem: an experimental evaluation of solution strategies, Management Science 27 (1981) 442–458.