PAPER

Ant Colony Optimization with Memory and Its Application to Traveling Salesman Problem

Rong-Long WANG[†a)], *Member*, Li-Qing ZHAO[†], *and* Xiao-Fan ZHOU[†], *Nonmembers*

SUMMARY Ant Colony Optimization (ACO) is one of the most recent techniques for solving combinatorial optimization problems, and has been unexpectedly successful. Therefore, many improvements have been proposed to improve the performance of the ACO algorithm. In this paper an ant colony optimization with memory is proposed, which is applied to the classical traveling salesman problem (TSP). In the proposed algorithm, each ant searches the solution not only according to the pheromone and heuristic information but also based on the memory which is from the solution of the last iteration. A large number of simulation runs are performed, and simulation results illustrate that the proposed algorithm performs better than the compared algorithms.

key words: *ant colony optimization, memory, combinatorial optimization problems, traveling salesman problem*

1. Introduction

Ant colony optimization (ACO) is a recently developed, population-based approach which has been successfully applied to several NP-hard combinatorial optimization problems [1]. Combinatorial optimization problems are of high importance both for the industrial world as well as for the scientific world. It arises in many different fields such as economy, commerce, engineering, industry and medicine [2]–[4]. As the name suggests, ACO has been inspired by the behavior of real ant colonies, in particular, by their foraging behavior. One of its main ideas is the indirect communication among the individuals of a colony of agents based on an analogy with trails of a chemical substance, called pheromone, which real ants use for communication. The pheromone trails are a kind of distributed numeric information [5] which is modified by the ants to reflect their experience accumulated while solving a particular problem. Recently, the ACO meta-heuristic has been proposed to provide a unifying framework for most applications of ant algorithms [6], [7] to combinatorial optimization problems. Algorithms which actually are instantiations of the ACO meta-heuristic will be called ACO algorithms in the following.

The first ACO algorithm, called Ant System (AS) was proposed by Dorigo in 1992 [8]. Since then, the ACO algorithm attracted the attention of more researchers and a number of other ACO algorithms have been introduced. Even though the original AS algorithm achieved encouraging results for the TSP problem, it was found to be inferior to state-of-the-art algorithms for the TSP as well as for other problems. Therefore, several extensions and improvements of the original AS algorithm were introduced over the years. ACS [9], [10] has been introduced to improve the performance of AS. It differs in three main aspects from the ant system. First, in ACS ants choose the next city using the pseudo-random-proportional action choice rule: when located at city i, ant k moves with probability q_0 to city l for which $\tau_{il}(t)[\eta_{il}]^{\beta}$ is maximal. With probability $(1 - q_0)$ an ant performs a biased exploration of edges according to the probability. Second, in ACS only the global best ant is allowed to add pheromone. The most interesting contribution of ACS is the introduction of a local pheromone update in addition to the pheromone update performed at the end of the construction process (called *offline* pheromone update). The effect of the local updating rule is to make an already chosen edge less desirable for a following ant. The Max-Min AS [11] is a direct improvement over AS. The main modifications introduced by Max-Min AS with respect to AS are the following. First, to exploit the best solution found, after each iteration only the best ant, which can be either the $iteration - best$ or the $best - so - far$, is allowed to add pheromone. Second, to avoid search stagnation, the allowed range of the pheromone trail strengths is limited to the interval $[\tau_{min}, \tau_{max}]$. Last, the pheromone trails are initialized to the upper trail limit, which causes a higher exploration at the start of the algorithm. Another improvement about AS is the rank-based version of Ant System (AS_{rank}) [12]. In AS_{rank}, always the global-best tour is used to update the pheromone trails. Additionally, a number of best ants of the current iteration are allowed to add pheromone. To this aim the ants are sorted by tour length, and the quantity of pheromone an ant may deposit is weighted according to the rank r of the ant. Only the $(\omega - 1)$ best ants of each iteration are allowed to deposit pheromone. The global best solution, which gives the strongest feedback, is given weight ω. The r_{th} best ant of the current iteration contributes to pheromone updating with a weight given by $max\{0, \omega - r\}$.

Therefore, one important focus of the research on ACO algorithms is the introduction of algorithmic improvements to achieve a much better performance. Typically, these improved algorithms have been tested again on the TSP [9], [12], [13]. Recent researches on the search space characteristics of some combinatorial optimization problems have shown that during the searching process it is very difficult to control the balance between intensification and diversification [14]. In this paper the ant colony optimization al-

Manuscript received June 24, 2011.
Manuscript revised October 27, 2011.

[†]The authors are with the Graduate School of Engineering, University of Fukui, Fukui-shi, 910-8507 Japan.
a) E-mail: wang@u-fukui.ac.jp

DOI: 10.1587/transfun.E95.A.639

gorithm with memory is proposed. It seems reasonable to assume that the concentration of the search around the solutions found in last iteration is the key aspect that leads to the improved performance. In the proposed algorithm, the ant searches for the solution not only according to the pheromone and heuristic information but also based on the memory, which is from the solution of the last iteration. To evaluate the performance of the proposed algorithm, we simulated some TSPLIB benchmark problems. The simulation results show that the proposed algorithm produces better results over the other existing ACO algorithms [15].

The remainder of this paper is structured as follows. In Sect. 2, the traveling salesman problem (TSP) is introduced. In Sect. 3 we outline how ACO algorithms can be applied to that problem. The proposed ant colony optimization algorithm with memory is addressed in Sect. 4. In Sect. 5 the proposed algorithm is applied to the TSP, and experimental results are presented in Sect. 6. The conclusions are given in Sect. 7.

2. The Traveling Salesman Problem

The traveling salesman problem is arguably the most famous problem in combinatorial optimization. The popularity of the TSP derives partly from the contrast between the simplicity of its statement and its computational complexity [16]. The TSP also plays an important role in ant colony optimization since the first ACO algorithm, called Ant System, as well as many of the subsequently proposed ACO algorithms, was initially applied to the TSP. The TSP was chosen for many reasons: (1) it is a problem to which ACO algorithms are easily applied, (2) it is an NP-hard optimization problem [17], (3) it is a standard test-bed for new algorithmic ideas and a good performance on the TSP is often taken as a proof of their usefulness, and (4) it is easily understandable, so that the algorithm behavior is not obscured by too many technicalities.

Intuitively, the TSP is the problem that a salesman who wants to find, starting from his home town, the shortest possible trip through a given set of customer cities and to return to its home town. The TSP can be represented by a complete graph $G = (N, A)$ with N being the set of nodes, also called cities, and A being the set of arcs fully connecting the nodes. Each arc $(i, j) \in A$ is assigned a value d_{ij} which represents the distance between cities i and j. The TSP then is the problem of finding a shortest closed tour visiting each of the N nodes of G exactly once (Such a tour is called *Hamiltonian.*). For symmetric TSP, the distances between the cities are independent of the direction of traversing the arcs, that is, $d_{ij} = d_{ji}$ for every pair of nodes. In the asymmetric TSP (ATSP) at least for one pair of nodes i, j we have $d_{ij} \neq d_{ji}$. In this paper the symmetric TSP is adopted. All the TSP instances used in the empirical studies presented in this paper are taken from the TSPLIB benchmark library. These instances have been used in many other studies and partly stem from practical applications of the TSP.

3. Ant Colony Optimization

The first ACO algorithm, called ant system (AS) was firstly applied to the traveling salesman problem (TSP). We call d_{ij} the length of the path between towns i and j, and let τ_{ij}, which called pheromone be the intensity of trail on edge (i, j) which connects i and j at time t. Each of m ants decides independently on the city to be visited next based on the intensity of pheromone trail τ_{ij} and a heuristic value η_{ij}, until the tour is completed. Each ant is placed on a random start city, and builds a solution going from city to city, until it has visited all of them. The probability by which an ant k in a city i chooses to go to a city j next is given by:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}(t)^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in J_i^k} \tau_{il}(t)^\alpha \cdot \eta_{il}^\beta} & \text{if } j \in J_i^k \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Where the heuristic value η_{ij}, the parameters α and β determine the relative influence of pheromone and heuristic, and J_i^k is the set of cities that remain to be visited by ant k positioned on city i. Once all ants have built a tour, ants perform the following pheromone update rule:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{n} \Delta\tau_{ij}^k(t) \tag{2}$$

Equation (2) consists of two parts. The left part makes the pheromone on all edges decay. The speed of this decay is defined by ρ, the evaporation parameter. The right part, where $\Delta\tau_{ij}^k(t)$ is defined by Eq. (3) below, in which Q is a positive constant, increases the pheromone on all the edges that are visited by ants. The amount of pheromone an ant k deposits on an arc (i, j) is defined by $L^k(t)$, the length of the tour created by that ant at iteration t.

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L^k(t)} & \text{if edge } (i, j) \text{ is used by ant } k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

In this way, the increase of pheromone for an edge depends on the number of ants that use this edge, and on the quality of the solutions found by those ants.

Afterwards, several extensions and improvements of the original AS algorithm which mentioned above were introduced over the years. One of the typical extensions is the Rank-based AS [12]. In Rank-based AS, always the global-best tour is used to update the pheromone trails. Additionally, a number of best ants of the current iteration are allowed to add pheromone. To this aim the ants are sorted by tour length, and the quantity of pheromone an ant may deposit is weighted according to the rank r of the ant. Only the $(\omega - 1)$ best ants of the iteration are allowed to deposit pheromone. The r_{th} best ant of the current iteration contributes to pheromone updating with a weight given by $max\{0, \omega - r\}$. Thus the improved update rule is:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{r=1}^{\omega-1} (\omega - r) \Delta\tau_{ij}^r(t)$$

```
procedure ACO algorithms for TSP
  set parameters, initialize pheromone trails
  while (termination condition not met) do
    construct solutions as follows:
      1) randomly select the initial component
      2) decide the next component according
         to probability, which is based on the
         pheromone and heuristic information
    update trails
  end-while
end-procedure
```

Fig. 1 Algorithmic skeleton for ACO algorithm.

$$+\omega\Delta\tau_{ij}^{gb}(t) \tag{4}$$

Where $\Delta\tau_{ij}^{r}(t) = Q/L^{r}(t)$ and $\Delta\tau_{ij}^{gb}(t) = Q/L^{gb}(t)$. In general, the ACO algorithms for the TSP follow the scheme in Fig. 1.

4. Ant Colony Optimization with Memory

As mentioned above, the first ACO algorithm, called ant system, was applied to the traveling salesman problem (TSP). It gave encouraging results, yet its performance was not competitive with state-of-the-art algorithms for the TSP. Therefore, one important focus of research on ACO algorithms is the introduction of algorithmic improvements to achieve a much better performance [18], [19]. Typically, these improved algorithms are tested again on the TSP. While they differ mainly in specific aspects of search control, all these ACO algorithms are based on a stronger exploitation of the pheromone and heuristic trails.

The ACO algorithms make use of ant agents which iteratively construct candidate solutions to a combinatorial optimization. The solution of each ant is constructed according to the pheromone trails deposited before and problem-dependent heuristic information. According to the combinatorial optimization, a lot of pheromone and heuristic modes are proposed. However, the searching mechanism has not been improved. In this paper an algorithm called ant colony optimization with memory is proposed. In the proposed algorithm, a novel searching mechanism is proposed to enhance the searching ability.

To solve the combinatorial problem, an individual ant constructs candidate solutions by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated. The ants' solution construction is guided by pheromone trails and problem-dependent heuristic information. In this paper, the ant-cycle version of AS (ant system) is adopted, in which, the pheromone update is only done after all the ants had constructed the solutions and the amount of pheromone deposited by each ant was set to be a function of the solution quality. After the solution construction is completed, the ants give feedback on the solutions they have constructed by depositing pheromone on solution components which they have used in their solution. Typically, solution components, which are part of better solutions or are used by many ants receive a higher amount of pheromone, and hence, will more likely be used by the ants in future iterations of the algorithm. To avoid the search getting stuck, typically before the pheromone trails get reinforced, all pheromone trails are decreased by a factor ρ. We call each point, at which an ant has to decide which solution component to add to its current partial solution, a choice point. In the AS, at the choice point, an ant decides the next component according to certain probability which is based on the pheromone and heuristic information. In the proposed algorithm, each ant searches the solution not only according to the pheromone and heuristic information but also based on the memory which is from the solution of the last iteration. The algorithmic scheme of the proposed algorithm is outlined in Fig. 2.

```
procedure ACO with memory
  set parameters, initialize pheromone trails
  while (termination condition not met) do
    construct solutions as follows:
      1) Randomly select the initial component
      2) Phase I:
         decide the next component according to
         probability, which is based on pheromone
         and heuristic information
         Phase II:
         compare the solution with the memory
         solution and adjust the components.
    update trails
  end-while
end-procedure
```

Fig. 2 Algorithmic skeleton for the proposed algorithm.

5. Ant Colony Optimization with Memory for TSP

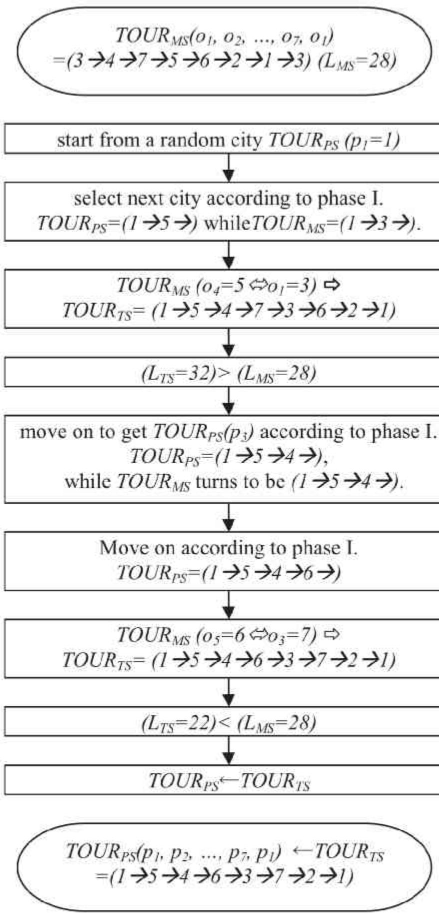
After the ant colony optimization with memory was proposed, we applied the algorithm to the TSP. When applying the proposed algorithm to the TSP, arcs between two cities are used as solution components, which was mentioned in Sect. 4. A pheromone trail $\tau_{ij}(t)$, where t is the iteration counter, is associated with each arc (i, j). These pheromone trails are modified during the run of the algorithm through the pheromone trail evaporation and the pheromone trail reinforcement by the ant colony. When it is applied to symmetric TSP instances, pheromone trails are also symmetric.

The tour construction is the most important part of the TSP. To describe the tour construction of the proposed method, we introduced $TOUR_{MS}(o_1, o_2, \cdots, o_n, o_1)$ as the memory solution, $TOUR_{PS}(p_1, p_2, \cdots, p_n, p_1)$ as the tour that is being constructed right now and $TOUR_{TS}$ as a temporary solution in the searching procedure, in which o_i, $p_i \in N$ in $G(N, A)$ are the city number of the TSP. The tour construction process of the proposed algorithm is outlined in Fig. 3. Initially m ants are placed on m randomly chosen cities, and

```
procedure Proposed decision rule
  randomly decide $p_1$ in $TOUR_{PS}$
  for $i$=1 to $n$ do
    find $o_x$==$p_i$ in $TOUR_{MS}$
    decide $p_{i+1}$ by AS decision rule
    find $o_y$==$p_{i+1}$ in $TOUR_{MS}$
    if $o_{x+1}$!=$o_y$ then
      swap $o_{x+1}$ and $o_y$ to get $TOUR_{TS}$
      if $L_{TS}$<$L_{MS}$ then
        $TOUR \leftarrow TOUR_{TS}$
        terminate
      else
        $TOUR_{MS} \leftarrow TOUR_{TS}$
      end-if
    end-if
  end-for
end-procedure
```

Fig. 3 Tour construction of the proposed algorithm.

the p_1 in $TOUR_{PS}$ of every ant is decided. In each construction step, two phases are performed. In the first phase, each ant moves based on a probabilistic decision to a city it has not yet visited. This probabilistic choice is biased by the pheromone trail $\tau_{ij}(t)$ and a locally available heuristic information η_{ij}. The function about the pheromone trail and the heuristic information was addressed in Sect. 3. As a result, the ants prefer cities which are close and connected by arcs with a high probability which was presented as Eq. (1). After every ant selects a city as its p_2 according to probability, the memory of every ant will be used in the following phase. In the second phase, the ant colony adjusts the solution component based on the memory which is from the solution of the last iteration. For each ant, the next city p_2 is decided according to the probability in the first phase, and then every ant compare its arc (p_1, p_2) with its memory solution respectively. As for the result of comparison, for the ant k, two kinds of situation will happen. The first situation is that the arc (p_1, p_2) of present solution is the same to that of the memory solution. For example $(p_1 = 5, p_2 = 3)$ in the present solution, and the arc $(o_i = 5, o_{i+1} = 3)$ appeared in the memory solution. In that situation, the ant make the next decision at choice city p_2, and in other words the next construction step begins from p_2 according two-phase selection formula to searching the next city p_3. The other situation is that the arc (p_1, p_2) of present solution could not be found in the memory solution. For example, $(p_1 = 5, p_2 = 3)$ in the present solution, but in memory solution of ant k, there is an arc $(o_i = 5, o_{i+1} \neq 3)$. For the second situation, we can find $o_x = o_i = 5(p_i)$ and $o_y \neq o_{i+1} = 3(p_{i+1})$ in the memory tour $TOUR_{MS}$, and exchange the values of o_{x+1} and o_y of the memory solution to get a temporary solution $TOUR_{TS}$. If the tour length of the temporary solution (L_{TS}) is shorter than that of the memory solution (L_{MS}), the tour construction process is terminated and the temporary solution becomes the solution of the present iteration. On the other hand, if L_{TS} is not shorter than L_{MS}, $TOUR_{MS}$ is replaced with $TOUR_{TS}$, and the next construction step begin from p_2 according to two-phase selection formula to search the next city p_3. Then, in the second phase of the construction step from city p_3 to find the next city p_4, we first find o_x again in $TOUR_{MS}$, and then compare o_x with o_{i+2} in the memory solution. According to the above construction step, an individual ant constructs a candidate solution by starting with an empty solution and then iteratively adding solution component until a complete candidate solution is generated.



$TOUR_{MS}(o_1, o_2, ..., o_7, o_1)$
$=(3\rightarrow4\rightarrow7\rightarrow5\rightarrow6\rightarrow2\rightarrow1\rightarrow3)$ $(L_{MS}=28)$
start from a random city $TOUR_{PS}$ $(p_1=1)$
select next city according to phase I.
$TOUR_{PS}=(1\rightarrow5\rightarrow)$ while$TOUR_{MS}=(1\rightarrow3\rightarrow)$.
$TOUR_{MS}$ $(o_4=5 \Leftrightarrow o_1=3) \Rightarrow$
$TOUR_{TS}=(1\rightarrow5\rightarrow4\rightarrow7\rightarrow3\rightarrow6\rightarrow2\rightarrow1)$
$(L_{TS}=32)>(L_{MS}=28)$
move on to get $TOUR_{PS}(p_3)$ according to phase I.
$TOUR_{PS}=(1\rightarrow5\rightarrow4\rightarrow)$,
while $TOUR_{MS}$ turns to be $(1\rightarrow5\rightarrow4\rightarrow)$.
Move on according to phase I.
$TOUR_{PS}=(1\rightarrow5\rightarrow4\rightarrow6\rightarrow)$
$TOUR_{MS}$ $(o_5=6 \Leftrightarrow o_3=7) \Rightarrow$
$TOUR_{TS}=(1\rightarrow5\rightarrow4\rightarrow6\rightarrow3\rightarrow7\rightarrow2\rightarrow1)$
$(L_{TS}=22)<(L_{MS}=28)$
$TOUR_{PS} \leftarrow TOUR_{TS}$
$TOUR_{PS}(p_1, p_2, ..., p_7, p_1) \leftarrow TOUR_{TS}$
$=(1\rightarrow5\rightarrow4\rightarrow6\rightarrow3\rightarrow7\rightarrow2\rightarrow1)$

Fig. 4 An instance of the proposed algorithm for TSP.

To show the searching process of the proposed algorithm in detail, an instance is given out. Here we maked a small-sized TSP with seven cities and adopted it to demonstrate the searching process, which is shown in Fig. 4. From Fig. 4, we can see that for every searching step, the ant selects one city according to the probabilities firstly, then compares with the memory solution, adjusts the solution according to different situation, and finally finds a better solution.

It is worth noting that although the proposed method has the similar exchanging operations with 2-opt algorithm, the exchanging mechanisms are essentially different. 2-opt considers only exchanging arcs to try to acquire the better solution, while our method considers exchanging nodes, and the decision of the nodes is based on the memory solution.

In addition, the proposed exchanging mechanism is performed in the procedure of building each solution element based on the memory solution. As a result, the proposed method concentrates the search around solutions found in the last iteration to enhance intensification.

As described above,all the ants can construct their own tour solutions as the candidate solutions by using the proposed tour construction procedure, and then update their pheromone trails. In this paper, the method of pheromone updating is the same as AS mentioned above. The pheromone trails are updating according to Eq. (2) and Eq. (3).

So far the algorithm of ant colony optimization with memory has been founded. The following search procedure describes the proposed ant colony algorithm for the TSP.

1. Set parameters.
2. Initialize the pheromone trails and compute the heuristic information.
3. All the ants construct candidate solutions according to Fig. 3.
4. Evaluate the candidate solutions and judge whether to terminate the procedure.
5. Update the pheromone trails and then go to the next iteration of tour construction.

6. Simulation Results

In order to assess the effectiveness of the proposed ACO algorithm, extensive simulations were carried out over TSPLIB benchmark problems on a PC station(Intel, 2.66 GHz). The parameters setting used in the proposed algorithm is that suggested in Rank-based AS [12], which are ($\alpha = 1, \beta = 5, \rho = 0.5, Q = 100$). The ant colony size is set to 100 in this paper.

Our first simulation is performed on Eil50 (50-city) to see the effect of the memory mechanism. We recorded the variation of the best, worst and average solutions during the evolution procedure of the solution, and illustrated it in Fig. 5. From this figure, we can find that our algorithm converges very fast. Furthermore, to see how efficiently the memory mechanism affects the performance, we also applied AS (Ant system) [8] to the Eil50 (50-city) for comparison. Note that the setting of parameters is the same as the proposed algorithm. Figure 6 shows the variation of

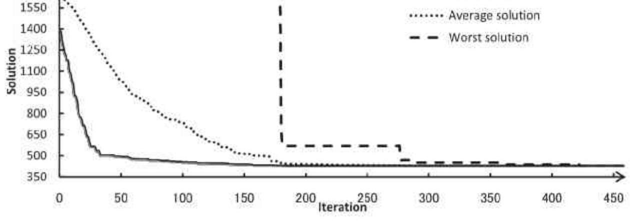


Fig. 5 Variation of the solution during the evolution procedure.

the average solution of AS and the proposed algorithm during the evolution procedure of solution. We can see that at the beginning, the effect of the memory is not obvious, but from the 150_{th} iteration, the memory plays an important role, and the advantage of the algorithm with memory comes out. Figure 7 shows the difference in detail. Figure 7 shows that the proposed algorithm converges quickly and can find a better solution. By this simulation, we can confirm the effection of the memory mechanism in the searching process.

To further evaluate the proposed algorithm, in the next simulations some other TSPLIB benchmark problems are selected and some best exsited ACO based algorithm such as Rank-based AS (AS_{rank}) [12], ACS [10] are used for comparison. Besides, some other softcomputing algorithms such as genetic algorithm (GA) [20], [21], evolutionary programming (EP) [22], simulated annealing (SA) [23] are also used for comparison. Note that for each instances, 100 simulation runs were performed. We give the results in Table 1, where the best integer tour length, the best real tour length (in parentheses) and the number of iteration required to find the best integer tour length (in square brackets) are recorded. The difference between integer and real tour length is that in the first case distance between cities are measured by integer numbers, while in the second case by floating point approximations of real numbers. Note that result of Rank-based

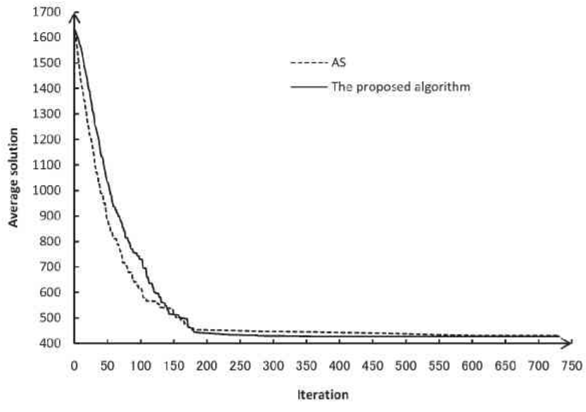


Fig. 6 The change of average solution using different algorithm.

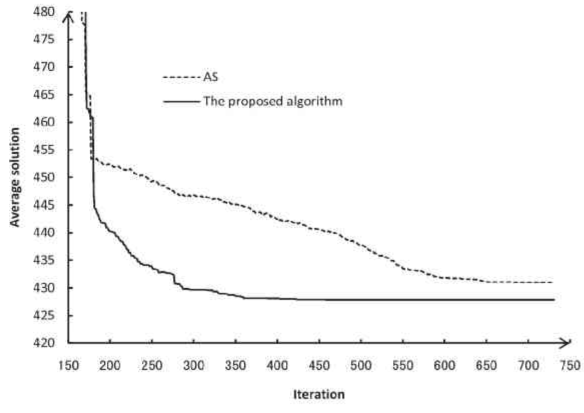


Fig. 7 The change of average solution in detail.

Table 1 Simulation results.

Problem	Optimum	GA	EP	SA	ACS	AS_{rank}	Proposed algorithm
Eil50 (50-city)	425 (N/A)	428 (N/A) [25000]	426 (427.86) [100000]	443 (N/A) [68512]	425 (427.96) [1830]	441 (435.13) [5000]	425 (427.85) [458]
Eil75 (75-city)	535 (N/A)	545 (N/A) [80000]	542 (549.18) [325000]	580 (N/A) [173250]	535 (542.31) [3480]	564 (566.88) [5000]	535 (542.30) [500]
KroA100 (100-city)	21282 (N/A)	21761 (N/A) [103000]	N/A (N/A) [N/A]	N/A (N/A) [N/A]	21282 (21285.44) [4820]	23278 (23599.12) [5000]	21282 (21285.44) [593]

Table 2 Average deviation from optimal value.

Problem	kroA100	kroB100	kroC100	kroD100	kroE100
Tabu search	0.26%	0.37%	0.42%	0.15%	0.45%
The proposed algorithm	0.01%	0.11%	0 %	0.03%	0.08%

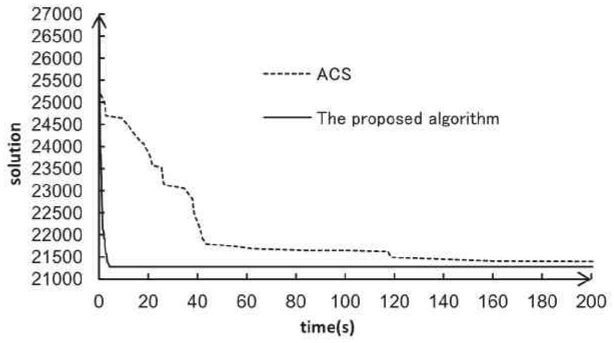


Fig. 8 Variation of the solution during the evolution procedure.

AS is from [14], and those of GA, EP and SA are from [9]. From the Table 1, it is clear that the proposed algorithm outperforms other algorithms in the solution quality. From the table we can also know that the proposed algorithm converges in hundreds of iterations comparing with other ACO algorithms and GA algorithm always performs several thousand search iterations for finding a solution. The reason why the proposed algorithm can find a good solution within small searching iterations can be considered that in the proposed algorithm, the memory of the ants present the better balanced intensification and diversification in searching process. In addition, in order to evaluate the proposed method from the aspect of the calculation cost, we compared the proposed method with ACS on real calculation time, where KroA100 is used. Figure 8 shows the result of the simulations. As shown in this figure, the proposed method can converge to a good solution within less time comparing with the other algorithm.

To evaluate the performance of the proposed algorithm comparing with other techniques, we also performed extended comparison with tabu search algorithm [24] on some 100-city problems. The simulation results are shown in Table 2, where the average deviations from the optimal value are listed. From this table, we can see that the performance of our proposed method is quite good and seems suitable for obtaining good solutions to the TSP problems.

7. Conclusions

An improved ACO algorithm with memory for efficiently solving combinatorial optimization problems have been proposed in this paper. In the proposed ACO algorithm, each ant searches the solution not only according to the pheromone and heuristic information but also based on the memory which is from the solution of the last iteration. The proposed algorithm was applied to the TSP, and to verify the effect of the memory, several TSP benchmark problems were simulated. From the simulation results, we find that the improved ACO algorithm has very high performance in searching solution comparing with other compared algorithms.

References

[1] T. Stutzle and H.H. Hoos, "MAX-MIN ant system," Future Gener. Comput. Syst., vol.16, no.8, pp.889–914, 2000.
[2] L.M. Gambardella, E. Taillard, and M. Dorigo, "Ant colonies for the quadratic assignment problem," J. Operational Research Society, vol.50, pp.167–176, 1999.
[3] T. Teich, M. Fischer, A. Vogel, and J. Fischer, "A new ant colony algorithm for the job shop scheduling problem," Proc. Genetic and Evolutionary Computation Conference GECCO2001, p.803, 2001.
[4] M. Reimann, K. Doerner, and R.F. Hartl, "D-ants: Savings based ants divide and conquer the vehicle routing problems," Computers & Operations Research, vol.31, pp.563–591, 2004.
[5] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," Future Gener. Comput. Syst., vol.16, no.8, pp.851–871, 2000.
[6] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," in New Ideas in Optimization, pp.11–32, McGraw-Hill, 1999.
[7] M. Dorigo, G. Di Caro, and L.M. Gambardella, "Ant algorithms for discrete optimization," Artificial Life, vol.5, no.3, pp.137–172, 1999.
[8] M. Dorigo, "Optimization, learning and natural algorithms," Italian PhD dissertationPolitecnico di MilanoMilan, 1992.
[9] M. Dorigo and L.M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., vol.1, pp.53–66, 1997.
[10] M. Dorigo and L.M. Gambardella, "Ant colonies for the traveling salesman problem," Bio Systems, vol.43, no.2, pp.73–81, 1997.

[11] T. Stutzle and M. Dorigo, "A short convergence proof for a class of ACO algorithms," IEEE Trans. Evol. Comput., vol.6, no.4, pp.358–365, 2002.
[12] B. Bullnheimer, R.F. Hartl, and C. Strauss, "A new rank based version of the ant system: A computational study," Central European Journal for Operations Research and Economics, vol.7, no.1, pp.25–38, 1999.
[13] T. Stutzle and H. Hoos, "The MAX-MIN ant system and local search for the traveling salesman problem," Proc. 1997 IEEE International Conference on Evolutionary Computation ICEC 97, vol.16, no.8, pp.309–314, 1997.
[14] R.-L. Wang and X.-F. Zhou, "Ant colony optimization with genetic operation and its application to traveling salesman problem," IEICE Trans. Fundamentals, vol.E92-A, no.5, pp.1368–1372, May 2009.
[15] G. Reinelt, "A traveling salesman problem library," ORSA Journal on Computer, vol.3, no.4, pp.376–384, 1991.
[16] G. Laporte, "A concise guide to traveling salesman problem," J. Operational Research Society, vol.61, no.1, pp.35–40, 2010.
[17] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Revista Da Escola De Enfermagem Da U S P, vol.44, no.2, p.340, 1979.
[18] L.M. Gambardella and M. Dorigo, "HAS-SOP: Hybrid Ant System for the Sequential Ordering Problem," Technical Report IDSIA 11-97, pp.1–22, 1997.
[19] V. Maniezzo, "Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem," INFORMS Journal on Computing, vol.11, pp.358–369, 1998.
[20] H. Bersini, C. Oury, and M. Dorigo, "Hybridization of Genetic Algorithms," Tech. Rep. no.IRIDIA 95-22, 1995.
[21] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and travelling salesman: The genetic edge recombination operator," Proc. 3rd International Conference on Genetic Algorithms, pp.133–140, 1989.
[22] D.B. Fogel, "Applying evolutionary programming to selected traveling salesman problems," Cybernetics and Systems: An International Journal, vol.24, no.1, pp.27–36, 1993.
[23] F.-T. Lin, C.-Y. Kao, and C.-C. Hsu, "Applying the genetic approach to simulated annealing in solving some NP-hard problems," IEEE Trans. Syst. Man. Cybern., vol.23, no.6, pp.1752–1767, 1993.
[24] M.P. Hansen, "Use of substitute scalarizing functions to guide a local search based heuristic: The case of moTSP," J. Heuristics, vol.6, no.3, pp.419–431, 2000.

Rong-Long Wang received a B.S. degree from Hangzhe teacher's college, Zhejiang, China and an M.S. degree from Liaoning University, Liaoning, China in 1987 and 1990, respectively. He received his D.E. degree from Toyama University, Toyama, Japan in 2003. From 1990 to 1998, he was an Instructor in Benxi University, Liaoning, China. In 2003, he joined University of Fukui, Fukui Japan, where he is currently an associate professor in Department of Electrical and Electronics Engineering. His current research interests include intellectual information technology, soft computing, and optimization problems.

Li-Qing Zhao received the B.S. and M.S. degrees in Micro-electronics from Nan-Kai University, Tianjin, china, in 2006 and 2009, respectively. During 2006 to 2009, she was focus on LCD research. From 2009 she is working toward her Ph.D degree at University of Fukui, Fukui, Japan. Her main research interests include ant colony optimization, neural network, genetic algorithm and combinatorial optimization problems.

Xiao-Fan Zhou received a B.S. degree in Electronic Science and Technology from Nanchang Hangkong University, Jiangxi, China in 2007, and an M.S. degree in Electrical and Electronics Engineering from University of Fukui, Fukui, Japan in 2010. During 2006 and 2010, he was focused on ant colony optimization and genetic algorithm. From 2010 he is working toward his Ph.D. degree at University of Fukui, Fukui, Japan. His main research interests include genetic algorithm, ant colony optimization and combinatorial optimization problems.