Proceedings of the 7th
World Congress on Intelligent Control and Automation
June 25 - 27, 2008, Chongqing, China

An improved ant colony optimization algorithm with embedded genetic algorithm for the traveling salesman problem

Fanggeng Zhao
Department of Logistics Engineering
1.University of Science & Technology Beijing
2.Vehicle Management Institute
1.Beijing, China
2.Bengbu, Anhui Province, China
zhaofanggeng@yahoo.com.cn

Jinyan Dong
Department of Vehicle Management
Vehicle Management Institute
Bengbu, Anhui Province, China
djy-djy@sohu.com

Sujian Li and Jiangsheng Sun
Department of Logistics Engineering
University of Science & Technology Beijing
Beijing, China
{lichaorong, sunjs}@263.net

***Abstract*—In this paper we proposed an improved ant colony optimization algorithm with embedded genetic algorithm to solve the traveling salesman problem. The main idea is to let genetic algorithm simulate the consulting mechanism, which may have more chances to find a better solution, to optimize the solutions found by the ants. In the proposed algorithm, we employed a new greedy way of solution construction and designed an improved crossover operator for consultation in the embedded genetic algorithm. Experimental results showed that the proposed algorithm could find better solutions of benchmark instances within fewer iterations than existing ant colony algorithms.**

Index Terms –Ant colony optimization; Genetic algorithm; Traveling salesman problem

I. INTRODUCTION

The ant colony optimization (ACO) algorithm is a multi-agent system in which the behavior of each ant is inspired by the foraging behavior of real ants to solve optimization problem. The main idea of ACO is to use the equivalent of the pheromone trail used by real ants as a medium for cooperation and communication among a colony of artificial ants. ACO algorithm has been successfully applied to several NP-hard combinatorial optimization problems, such as the traveling salesman problem(TSP) [9], the quadratic assignment problem (QAP) [14], the vehicle routing problem (VRP) [13] and the job-shop scheduling problem (JSP) [3], since it was introduced by Dorigo [9]. Among these problems, TSP plays an important role in ACO algorithm because almost all researches on ACO have been tested on this problem. TSP was chosen mainly for three reasons [5]: (I) it is a problem to which the ACO is easily adapted, (II) it is one of the most studied NP-hard problems in combinatorial optimization, and (III) it is easier comprehend. Thus, we here focused on TSP as application domains for the proposed algorithm.

Because the first ACO algorithm, named Ant System, was not competitive with state-of-the-art algorithms for TSP, many variants, including AS_{elite}, AS_{rank}, *MM*AS, and cAS et al, of the Ant System have been proposed by researchers. And till now, ACO algorithm is still confronted with how to improve its performance.

In this study, we present an improved ant colony optimization with embedded genetic algorithm (GA) for the TSP to improve the performance of ACO algorithm. In this algorithm, the embedded genetic algorithm is used to simulate the consulting mechanism among ants in order to optimize the solutions found by them, and then the pheromone trails are updated. The adoption of embedded GA not only speeds up the search, but improves the quality of solution.

The remainder of this paper is organized as follows. Section 2 provides a briefly review of the ACO algorithms that have been proposed for the TSP. Sections 3 and 4 present the proposed algorithm and its computational results respectively. Finally, Section 5 draws conclusions from this study.

II. ACO FOR THE TRAVELING SALESMAN PROBLEM

The traveling salesman problem is the first application of ACO algorithm. The general TSP can be represented by a complete graph $G = (N, A)$ with N being the set of 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 $n = |N|$ nodes of G exactly once.

In 1991, Dorigo et al [9] proposed the first ACO algorithm named Ant System (AS). In AS, artificial ants construct solutions (tours) of the TSP by moving from one city to another. The algorithm executes t_{max} iterations, in the following indexed by t. During each iteration, m ants build a tour executing n steps in which a probabilistic decision (state transition) rule is applied. In practice, when in node i the ant chooses the node j to move to, and the arc (i, j) is added to the tour under construction. This step is repeated until the ant has completed its tour. According to the way pheromone trails are updated, AS algorithm can be divided into three types: *ant-density, ant-quantity, and ant-cycle* [2,4,9,10]. In ant-density and ant--quantity ants deposit pheromone while building a solution, while in ant-cycle ants deposit pheromone after they have constructed a complete tour. Numerous experiments run

978-1-4244-2114-5/08/$25.00 © 2008 IEEE.

on benchmark instances[4,9,10] indicate that the performance of ant-cycle is much better than that of the other two algorithms. Although the performance of AS is limited compared with other heuristics especially on large TSP problems, it is the prototype of a number of ant algorithms that have found many interesting and successful applications.

AS_{elite} is the first improvement of AS [1,7,9], in which the best ant has more "weight" in contributing to the pheromone trails than other ants. Bullnheimer et al [1] proposed a rank-based version of AS, named AS_{rank}. Different from AS_{elite}, both the best ant and some other ants that provide good solutions can add pheromone trails in AS_{rank}, and likewise, the "better" ants have the higher "weight". Computing results of benchmark instances show the effectivities of these measures.

In order to avoid the stagnation in which all ants are stuck within a local optimum, Stützle and Hoos [18,19, 20] presented the *Max-Min* AS (*MM*AS), which differs from AS in three key aspects: (I) only one single ant (the one which found the best solution in the current iteration or the one which found the best solution from the beginning of the trial) adds pheromone after each iteration, (II) pheromone trail values are limited to an interval $[\tau_{min}, \tau_{max}]$, and (III) pheromone trails are initialized to be τ_{max}. In this way, *MM*AS combines an improved exploitation of the best solutions found during the search with an effective mechanism for avoiding early search stagnation [20], and performs much better than AS.

Another important improvement of AS that applied to solve TSP is the Ant Colony System (ACS) [6,8,12]. Similar to *MM*AS, ACS adds the pheromone trails of the arcs on the best tour from the beginning of trail, however, it uses pseudo-random-proportional rule to select next city when the ants construct tours. In addition, ACS uses local pheromone trails update mechanism and exploits *candidate list*, a data structure that provides additional local heuristic information. Tests on benchmark instances indicate that these measures can not only improve the quality of the solutions but also reduce the computing time. Based on the modifications mentioned above, local search methods, such as 2-opt and 3-opt, are integrated into ACO algorithms [6,10,18]. Computing results show that the application of local search methods can significantly improve the solution quality.

Marcin and Tony [16] combined the genetic algorithm (GA) and ACS, and proposed two hybrid algorithms to improve the performance of ACS. Although their first algorithm that uses a population of genetic ants modified by a GA, called ACSGA-TSP, can not outperform the ACS algorithm, it has the advantage of quick convergence and low variability. The second algorithm in [16], named Meta ACS-TSP algorithm, uses GA to optimize the parameter settings used in the ACS algorithm. Experiments using their second algorithm result in a new serial of appropriate parameter values instead of those used by Dorigo and Gambardella [6]. Similar work that uses GA to optimize the parameter settings of ACS can be found in [11]. In addition, Hao et al [21] introduced a new adaptive parameter control strategy that uses particle swarm optimization (PSO) to solve the parameter setting problem.

More recently, Tsutsui [22] proposed a variant of ACO algorithm called the cunning Ant System (cAS). In cAS, a ant constructs a solution according to both the pheromone density and a part of a solution from a previous iteration, and this cunning action can reduce premature stagnation and improve the performance of ACO algorithm.

III. The Proposed Algorithm for the TSP

In this section, we present the proposed algorithm with embedded GA for the TSP. The main idea of this algorithm is to let the ant consult with each other and thereby enhance the share mechanism of information, which may have more chances to find a better solution. The crossover operator of GA can combine the genes in the parents, and the offspring of crossover can be seen as the consulting result of the parents to some extent. So we use heuristic crossover operator as the method for ants to consult with others. The pseudo-code of proposed algorithm is shown in Fig. 1, where [*i/2*] represents the rounded value of *i/2*. In the proposed algorithm, after initialized the parameters that needed in the algorithm, *m* ants begin to construct the solutions which will constitute the population for GA. Then, the improved GA embedded into ACO to optimize the solutions constructed by the ants, and the pheromone trails are updated according to S^{gb} (the best solution from the beginning of the trial). The program will be iterated until the termination condition satisfied. In the reminder of this section, we introduce the main operators used in the proposed algorithm.

A. The construction of solution

In the proposed algorithm, we employed a new greedy way to construct solutions. Initially, *m* ants are placed on *m* cities randomly. Then, in each construction step, each ant (in city *i*) has to choose an unvisited city *j* as the next destination. At

```
Procedure: The proposed algorithm
Begin
  Initialize parameters;
  While (termination condition not satisfied) do
     The ants construct m solutions which constitute the population P for GA;
     Evaluate the solutions;
     i = m;
     While (i >=2) do
        For ( j=0; j< [i/2]; j++)
           Select two solutions, e.g. p1 and p2, from the population;
           Recombine the selected solutions using heuristic crossover operator
             to yield c(j);
           Mutate c(j) with mutation probability pm;
           Let the best solution in { p1, p2, c(j) } enters new_population;
        End for;
        i =[i/2];
        Replace the population with new_population;
     End while;
     Save the best solution found from the beginning as S^gb;
     Update the pheromone trails;
  End while;
End;
```

Fig. 1 The proposed algorithm in pseudo code.

time t, an ant k that located at city i selects the next city in the light of the following rules:

(I) ant k first chooses the target city from the candidate list[17] of city i, the static data structure that includes cl closest cities from city i, and each city j in the candidate list will be selected with a probability[10]:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{k\in S_k}[\tau_{ik}(t)]^{\alpha}\cdot[\eta_{ik}]^{\beta}} & if \quad j\in S_k\\ 0 & otherwise\end{cases}, \qquad (1)$$

where $\tau_{ij}(t)$ is the value of pheromone trail on edge (i, j) at time t, η_{ij} ($\eta_{ij}=1/d_{ij}$) is the visibility of j from i, α and β are parameters that determine the relative importance of pheromone trail and the visibility, and S_k is the intersection of the candidate list of city i and the set of cities that ant k has not visited yet.

(II) If $S_k=\Phi$, ant k chooses the next city j in the set of cities that ant k has not visited to move to by the following the state transition rule:

$$j=\arg\max_{s\in allowed_k}\left\{[\tau(i,s)]^{\alpha}\cdot[\eta(i,s)]^{\beta}\right\} \qquad (2)$$

where $allowed_k$ is the set of cities that ant k has not visited yet.

B. The embedded genetic algorithm

In our proposed algorithm, the GA is embedded into the ACO with the purpose of simulating the consulting process between ants. The m solutions constructed by ants constitute the initial population $P=\{p_1, p_2, \ldots, p_m\}$ for GA. The population size will be halved each cycle and the GA will be stopped till only one solution is left in the population. In the embedded GA, the fitness value of a solution p_i is defined as follows:

$$fitness(p_i)=M-L_i \qquad (3)$$

where L_i is the length of p_i and M is defined as:

$$M=\lambda\cdot\max_{i}\{L_i\} \qquad (i=1,2,\ldots,m) \qquad (4)$$

where λ is a coefficient is set to be 1.15 in our experiments. After being evaluated, the solutions will be selected according to roulette wheel selection for crossover operator.

1) Heuristic crossover operator

In this paper, we design an improved heuristic crossover operator to produce the offspring. The improved heuristic crossover operator utilizes both adjacency information and heuristic information (including distances and pheromone trails between cities) among genes to yield the offspring.

Initially, the improved crossover operator randomly selects a city as the first city of the offspring. Then the operator chooses the nearest city among the immediate predecessor and successor of the first city, called *neighbors*, in the two parents as the second city of offspring. If all the cities in *neighbors* have been visited, the next city will be selected according to the following formula:

$$j=\arg\max_{k\in N}\{\tau(o_i,k)\} \qquad (5)$$

where N is the set of cities that have not appeared in o. The process is repeated until all cities are visited. Let n be the total number of cities, o be offspring, and o_{i-1} be the ith city in o. The pseudo-code of the improved crossover operator is shown in Fig. 2.

2) Mutation operator

After crossover operator, the offspring will be mutated with the probability p_m to avoid stagnation. In the embedded GA, we use 3-exchange mutation operator instead of 2-exchange described in [15]. The 3-exchange operator randomly selects 3 cities first, and then simply exchanges the places of them. Obviously, there are 5 possible combinations besides the offspring itself, and the best one will be selected as the result of mutation.

C. Pheromone trail and its updating

Like the *MM*AS in [20], the proposed ACO algorithm imposes explicit limits τ_{min} and τ_{max} on the minimum and maximum pheromone trails to avoid stagnation, and the initial values of pheromone trails are set to be τ_{max}. In addition, τ_{min} and τ_{max} are defined as follows:

$$\tau_{\max}=\frac{1}{1-\rho}\cdot\frac{1}{f(S^{gb})} \qquad (6)$$

$$\tau_{\min}=\tau_{\max}/2n \qquad (7)$$

where $0<\rho<1$ is the trail persistence, $f(S^{gb})$ is the cost of the globally best solution , and n is total number of cities.

After the solutions are optimized by GA, the pheromone trails are updated on the basis of the globally best solution (S^{gb}). According to the *MM*AS proposed in [20], the pheromone update is done as follows:

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}^{gb} \qquad (8)$$

where $\Delta\tau_{ij}^{gb}=\begin{cases}1/f(S^{gb}) & if(i,j)\in s^{gb}\\ 0 & otherwise\end{cases}$ (9).

Equation (8) dictates that only those edges belonging to the globally best solution will be reinforced.

Finally, the termination condition is set as the number of generation exceeds the permitted generation number.

```
Procedure: The improved crossover operator
Begin
 i = 0;
 Select a random city as initial city o0 of the o;
 While ( i < n ) do
       Search the positions of oi in parents p1 and p2 respectively;
       If ( all the neighbors of oi in p1 and p2 have appeared in o)
        Select the next city j according to (5):
       Else
        Select the nearest city that has not appeared in o from the neighbors
          of oi as the next city j;
       End if;
       i = i+1;
       oi ← j;
 End while;
 on ← o0;
End;
```

Fig. 2 The proposed crossover operator in pseudo code.

IV. COMPUTATIONAL RESULTS AND ANALYSIS

In this section we present numerical results for our proposed algorithm and compare them with results from previous ACO algorithms, including the *MM*AS, ACS, Meta ACS-TSP and cAS. For the proposed algorithm, various parameters have to be set. In the following experiments, the parameter settings used are: $\rho = 0.8$, $\alpha = 1$, $\beta = 2$, $m = 35$, $p_m = 0.1$, $cl = 20$. Since values for different parameters are set heuristically, we perform some additional tests and verify that, within a moderate range, there is no significant difference in the outcomes. All the TSP instances used here are taken from the TSPLIB benchmark library.

A. Performance of the ACO algorithm with embedded GA

Computational results of the proposed algorithm are presented in Table I, and the computations are run for 1000 iterations. In Table I, *opt* indicates the known optimal solution value of each instance, *best* shows the length of the best solution found together with its deviation from the known optimum in percent, *average* and *worst* display the same information for the average and the worst of 20 independent runs (10 independent runs for the 3 larger instances). From Table I we can see that the proposed algorithm can find the known optimal solutions of smaller benchmark instances tested here in at least one of the runs. Both the *average* and *worst* deviations from the optimum are less than 1%. Moreover, for instances kroA100 and lin105, all these 20 experiments have found the optimal solutions.

In Table II we compare the average results of proposed algorithm to those obtained by *MM*AS, ACS, and cAS. The results of MMAS are taken from [20], and those of ACS, Meta ACS-TSP, cAS are from [12], [16] and [22] respectively. Table II illustrates that the results obtained by the proposed algorithm are better than, at least the same to, those obtained by *MM*AS, ACS, Meta ACS-TSP and cAS.

TABLE I
RESULTS OF THE PROPOSED ALGORITHM RUNS ON TSP INSTANCES

Instance	opt	best	average	worst
eil51	426	426 (0.0%)	426.20 (0.047%)	427 (0.235%)
eil76	538	538 (0.0%)	538.20 (0.037%)	539 (0.186%)
kroA100	21282	21282 (0.0%)	21282.00 (0.0%)	21282 (0.0%)
lin105	14379	14379 (0.0%)	14379.00 (0.0%)	14379 (0.0%)
ch130	6110	6110 (0.0%)	6121.95 (0.196%)	6155 (0.736%)
d198	15780	15781 (0.006%)	15800.25 (0.128%)	15826 (0.292%)
lin318	42029	42029 (0.0%)	42125.30 (0.229%)	42163 (0.319%)
pcb442	50778	50919 (0.278%)	50944.10 (0.327%)	50976 (0.390%)
att532	27686	27858 (0.621%)	27909.30 (0.807%)	27962 (0.997%)

TABLE II
THE COMPARISON OF EXPERIMENTAL RESULTS WITH OTHER ALGORITHMS

Instance	proposed algorithm [a]	MMAS [b]	ACS [b]	Meta ACS-TSP [c]	cAS [b]
eil51	426.2	427.6	428.1	428.5	426.2
eil76	538.2	--	--	542.5	--
kroA100	21282.0	21320.3	21420.0	21513	21282.0
d198	15800.25	15972.5	16054.0	--	15954.1

[a] Results after 1000 iterations.
[b] Results after $n \cdot 10000$ iterations, where n is the number of cities, and over 25 runs.
[c] Results after 1000-3000 iterations.

Among these algorithms, Meta ACS-TSP used GA to optimize the parameters of ACO. This is a different way to combine the ACO and GA. The comparison in Table II demonstrates the superiority of our method. Another important thing should be pointed out is that our algorithm find these better results through much fewer iterations (the computing times are not given in those literatures).

B. The influence of embedded GA on the performance of ACO algorithm

To illustrate the role of the embedded GA, we compared the evolutional processes of the best solution obtained by the proposed algorithm and the algorithm without embedded GA in Fig. 3. Fig. 3 clearly shows that the embedded GA can significantly speed up the searching process, and improve the quality of final solutions.

Hence, these results strongly suggest that the embedded GA successfully simulates the consulting mechanism among ants, and thereby improves the performance of ACO algorithm.

V. CONCLUSION

In this paper, an improved ACO algorithm with embedded GA for the TSP is proposed. Within the proposed algorithm, ACO algorithm is used to construct solutions that constitute the initial population for genetic algorithm, while the embedded GA is used to simulate the consulting mechanism among the ants to optimize the solutions. In the embedded GA, we design a new heuristic crossover operator that utilizes adjacency information, distances and pheromone trails between cities to yield offspring. The experiments on benchmark instances of TSP have shown that the embedded GA can significantly improve the performance of ACO, and the proposed algorithm outperforms existing ACO algorithms for TSP.

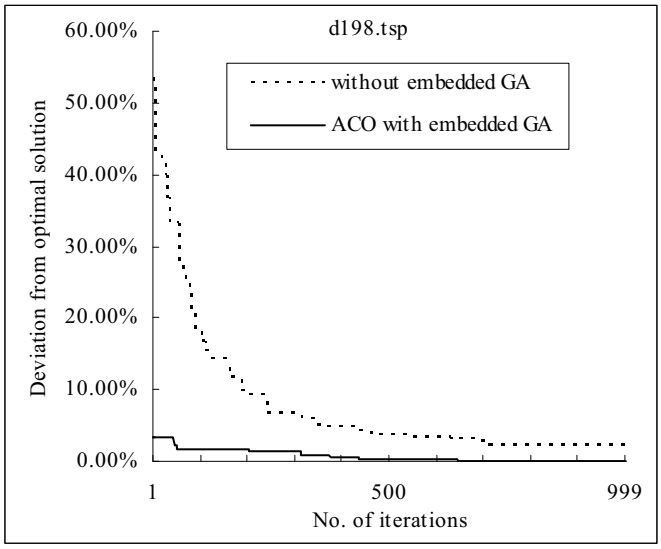

Fig. 3 Comparison of evolutionary process between the proposed algorithm and the algorithm without embedded GA.

REFERENCES

[1] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the ant system - a computational study," Central European J. Oper. Res. Economic, vol. 1, pp. 25–38, Jan. 1999.
[2] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," in Proceedings of ECAL91 European Conference of Artificial Life, Paris, 1991, pp. 134-144.
[3] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system for job-shop scheduling problem," Belgian Journal of Operation Research, Statistics and Computer Science, vol. 34, pp. 39-53, Jan. 1994.
[4] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.
[5] M. Dorigo, and G. Di Caro, *New Ideas in Optimization*, McGraw-Hill Press, 1999, pp. 11-32.
[6] M. Dorigo, and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Trans. Evolut. Comput, vol. 1, pp. 53–66, Jan. 1997.
[7] M. Dorigo, and L. M. Gambardella, "A Study of Some Properties of Ant-Q," Universite Libre de Bruxelles, Beligium. Technical Report IRIDIA 1996-4, 1996.
[8] M. Dorigo, and L. M. Gambardella , "Ant Colonies for the Traveling Salesman Problem," BioSystems, vol. 43, pp. 73-81, 1997.
[9] M. Dorigo, V. Maniezzo, and A. Colorni , "Positive feedback as a search strategy," Politecnico di Milano, Dipartimento di Elettronica, Milan, Italy . Tech. Rep. 91-016, 1991.
[10] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," IEEE Trans. Systems Man Cybernet. B, vol. 26, pp. 29–42, 1996.
[11] D. Gaertner, and K. Clark, "On optimal parameters for ant colony optimization algorithms," in Proceedings of the International Conference on AI 2005, Las Vegas, February, 2005.
[12] L. M. Gambardella, and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in Proc. the IEEE Int. Conf. on Evo. Comp, 1996. pp. 622–627.
[13] B. Bullnheimer, R. F. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem. Meta-Heuristics: Advances and Trends," in Local Search Paradigms for Optimization. Kluwer, Boston, 1998, pp. 109-120.
[14] L. M. Gambardella, E. Taillard, and M. Dorigo, "Ant Colonies for QAP," IDSIA, Lugano, Switzerland, Tech1Rep1IDSIA 97 - 4 ,1997.
[15] M. Herdy, "Application of evolution strategy to discrete optimization problem," in Proceedings of the 1st Conference on Parallel Problem Solving from Nature, Springer-Verlag, Lecture Notes in Computer Science, vol. 496, 1991, pp. 188-192.
[16] L. P. Marcin, and W. Tony, "Using genetic algorithms to optimize ACS-TSP," in Proc of 3rd Int Workshop ANTS, Brussels, 2002, pp. 282-287.
[17] G. Reinelt, "The traveling salesman: computational solutions for TSP applications," Springer-Verlag, 1994.
[18] T. Stützle, and H. Hoos, "The MAX–MIN ant system and local search for the traveling salesman problem," in Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, USA, 1997, pp. 309–314.
[19] T. Stützle, and H. Hoos, "Improvements on the Ant System: Introducing MAX-MIN ant system," in Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Springer-Verlag, 1997, pp. 245–249.
[20] T. Stützle, and H. Hoos, "MAX-MIN Ant System," Future Generation Computer Systems, vol. 16, 2000, pp. 889–914.
[21] Z. F. Hao, R. C. Cai, and H. Huang, "An adaptive parameter control strategy for ACO," in Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 2006, pp. 13-16.
[22] S. Tsutsui, "Ant colony optimization with cunning ants," Transactions of the Japanese Society for Artificial Intelligence, vol. 22, pp. 29-36, Jan. 2007.