Improved Ant Colony System using Subpath Information for the Traveling Salesman Problem

Minyoung Yun and Inkyeom Kim

Department of Information and Communications Engineering,
Sungkyul University, Anyang, Korea
{alabama, kik}@sungkyul.edu

Abstract. Ant Colony System (ACS) applied to the traveling salesman problem (TSP) has demonstrated a good performance on the small TSP. However, in case of the large TSP, ACS does not yield the optimum solution. In order to overcome the drawback of the ACS for the large TSP, the present study employs the idea of subpath to give more information to ants by computing the distance of subpath with length 3. In dealing with the large TSP, the experimental results indicate that the proposed algorithm gives the solution much closer to the optimal solution than does the original ACS. In comparison with the original ACS, the present algorithm has substantially improved the performance. For a certain graph, the solution performance has been enhanced up to 72.7 % by utilizing the proposed algorithm.

1 Introduction

Ant System(AS) is a meta-heuristic algorithm proposed by Dorigo et al.[1] that has been inspired by the foraging behavior of ant colonies. Real ants are capable of finding the shortest path from a food source to their nest by exploiting pheromone information. Ant System was applied to the complex combinatorial optimization problems such as the traveling salesman problem (TSP) and the quadratic assignment problem (QAP). Currently many ongoing research activities has been performed to investigate many different discrete optimization problems like vehicle routing, sequential ordering, graph coloring, and routing in communication networks.

In the present study, the Ant Colony System has improved the efficiency of the existing ant system and it has been applied to analyze TSP. In context with the Ant Colony System, the ants acting like agents perform parallel search for the TSP and find a good solution. During this process, the ants are able to exchange information each other indirectly but globally by using pheromone [5]. Each ant constructs the path for TSP with the iterative procedure to select a next visiting city by jointly utilizing informations on the greedy heuristic and the past experience. Several meta-heuristic search algorithm are applied to find an optimal solution for TSP which is well known as a NP-hard problem.

The TSP can be expressed by a complete weighted graph $G = (V, E)$. Here V is a set of vertices and $|V| = n$, it represents all cities that the sales person has to

Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 185-194

visit. The E denotes a set of edges. Each edge $(i, j) \in E$ has a weight d_{ij} which represents a distance between any two cities i and j $(i, j \in V)$. Consequently, the TSP can be converted to a Hamiltonian circuit problem which find a shortest path from a starting city by visiting each city only once and returning to the starting city on a complete weighted graph. The TSP is classified as symmetric TSP and asymmetric TSP. In the asymmetric TSP, the distance of the paired vertices (i, j), d_{ij}, could be different for the circulating direction. In other word, there exists at least one edge which satisfies $d_{ij} \neq d_{ji}$. In the symmetric TSP, $d_{ij} = d_{ji}$ is satisfied for every edges in E.

The original ACS algorithm is capable of finding an optimal solution for the small size of TSP. The original ACS uses information on distance of adjacent neighbors only. However, in case of the large TSP, ACS does not yield the optimum solution. In order to overcome the drawback of the ACS for the large TSP, the present study employs the idea of subpath to give more information to ants by computing the distance of all possible subpath with length to construct a tour for a solution. In dealing with the large TSP, the experimental results indicate that the proposed algorithm gives the solution much closer to the optimal solution than does the original ACS. For a certain graph, the solution performance has been enhanced up to 72.9 % by utilizing the proposed algorithm. In comparison with the original ACS, the present algorithm has considerably improved the performance. The detailed discussion has been made for the existing and proposed algorithm for the ant colony optimization to solve the large TSP with a symmetry.

2 Ant Colony Optimization Algorithms

The Ant Colony Optimization(ACO) algorithm is easily applicable to handle the TSP. In the ACO algorithm, the pheromone trails consist of the connecting edges and τ_{ij} represents the measure of possibility to visit a city j directly from a city i. The heuristic information is expressed as $\eta_{ij} = 1/d_{ij}$. The values of τ_{ij} and η_{ij} are stored at pheromone matrix and heuristic matrix, respectively. For each ant, tours are constructed by the following procedure : (1) choose a start city in random fashion and place an ant; (2) according to values of τ_{ij} and η_{ij}, construct a path by adding a city that the ant has not visited yet; and (3) after all cities have been visited, go back to the starting city and complete one path. After all ants have completed their tour, they may deposit a certain amount of pheromone according to the tour they have constructed [7,8].

The Ant System(AS) is a initially developed ACO algorithm and it is quite easy to apply the TSP. However, due to the simple pheromone updating rule, there is a certain tendency the AS leads to the local optima situation. Therefore, the AS gives the optimal solution only for the small TSP. To improve the performance, several extensions of the AS was devised. These extensions include elitist AS, rank-based AS, and MAX-MIN AS. The main difference between the original AS and these extensions is the way to update the pheromone[4]. The ACS algorithm is the framework of this study and it is the ACO algorithm by adopt-

ing the basic idea of the AS. Its performance has been improved by overcoming the drawbacks of the AS. The ACS has been applied to various combinatorial optimization problems and it has demonstrated a good performance.

The ACS proposed by Gambardella and Dorigo[9] differs from the AS in the following features:

1. By using a more aggressive action choice rule, compared to the AS, the ACS more actively exploits the search informations accumulated by the ants.
2. Pheromone evaporation and pheromone deposit take place only on the edges belonging to the best-so-far tour.
3. Each time an ant uses an edge (i, j) to move from city i to city j , it removes some pheromone from the edge to increase the room for selecting the alternative paths.

In the initial stage of the ACS with a given graph $G = (V, E)$ and $|V| = n$, m ants ($m \leq n$) are placed on m cities in random fashion. According to the tour construction rule, each ant repeatedly chooses a next visiting city and constructs a path. In this process, whenever an edge is added to a path, the local pheromone updating rule is applied to update the pheromone on each edge. When the path is constructed, the local search is applied to improve the constructed path. Then the pheromone is updated only at the global optimal path with the minimum length among all paths constructed so far. Figure 1 shows the ACS algorithm for the TSP.

```
algorithm: ACS for TSP {

   Initialize Data;
   while (not terminate) {
      place m ants at m cities;
      repeat (for each ant)
           apply tour construction rule to build a trail;
           apply local pheromone updating rule;
      until (construct a solution)
      apply local search;
      apply global pheromone updating rule;
   }
}
```

Fig. 1. Algorithm: ACS for TSP

2.1 Tour Construction Rule

If ant k is located at city i, then a next visiting city j can be chosen according to the pseudo-random proportional rule, given by equation (1).

$$j = \begin{cases} \arg \max_{l \in N_i^k} \{\tau_{il}[\eta_{il}]^\beta\}, & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{1}$$

where β is a parameter which determines the relative importance of pheromone τ_{ij} versus heuristic information η_{ij}, ${N_i}^k$ is the set of the remaining cities to be visited by ant k positioned on city i. q is a random variable uniformly distributed in $[0, 1]$, q_0 is a parameter to satisfy the range, $0 \leq q_0 \leq 1$, and J is a random variable selected by the following probability distribution.

$$p_{ij}^k = \frac{[\tau_{ij}][\eta_{ij}]^\beta}{\sum_{l \in N_i^k}[\tau_{il}][\eta_{il}]^\beta} \qquad \text{if} \quad j \in N_i^k \tag{2}$$

In the equation (2), the probability to select an edge (i, j) in a path is dependent on the amount of pheromone, τ_{ij} and heuristic information, η_{ij}. Each ant select a city j as a next visiting city which has a large level of pheromone and a short distance. If $\beta = 0$, the selection of a next city depends only on the pheromone level, τ_{ij}. Therefore, in the general situations, $\beta > 1$, according to reference[4], a good performance is achieved at $2 \leq \beta \leq 5$.

2.2 Local Pheromone Trail Update

Unlike the AS, the ACS uses a local pheromone updating rule. Whenever an ant constructs a tour of the TSP and select an edge, the pheromone level for a selected edge is updated by applying the local updating rule equation (3).

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{3}$$

where ξ is the variable to satisfy the range, $0 \leq \xi \leq 1$. According to numerical experiment, the best performance is achieved at ξ =0.1 [4]. The value of τ_0 represents the initial pheromone level and the best performance is obtained at $\tau_0 = 1/(nC^{mn})$, where n is the number of cities in the TSP and C^{mn} is the length of a path constructed by the nearest-neighbor heuristic. In other word, the pheromone level at each edge is initialized by the length of a path which is constructed by the greedy method. By applying the equation (3), whenever an ant selects an edge (i, j), its pheromone level, τ_{ij} at a selected edge is reduced. As a result, the once selected edge has the much lower probability to be selected by the following ants. This treatment increases the probability to select the edges that have not been visited yet and it prevents from a stagnation behaviour which is a certain tendency to repeatedly choose an once selected edge. In other words, ants do not converge to the generation of a common path. In this study, we only consider symmetric TSP such that $\tau_{ij} = \tau_{ji}$.

2.3 Local Search

The local search is basically included in the ACO algorithm. After all ants have completed to find their own path, the locally optimum solution can be obtained by 2-opt or 3-opt procedure which exchange two or three edges involved in the constructed path. If this local search is applied to construct a path of the TSP, the ACO algorithm together with a local search can improve the solution constructed by an ant[10]. In this proposed algorithm, a 3-opt method is employed.

2.4 Global Pheromone Trail Update

In the procedure of a Global Pheromone Trail Update, the pheromone update is allowed only for the most optimum path among all constructed paths, according to the equation (4).

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \qquad \forall(i,j) \in T^{bs} \tag{4}$$

where $\Delta\tau_{ij}^{bs}$ is the amount of pheromone to be added to edge (i,j) in the optimal path. C^{bs} represents the length of global optimal solution. Thus, the relation between the pheromone level and the optimum path length is expresses as $\Delta\tau_{ij}^{bs} = 1/C^{bs}$. The parameter ρ is the pheromone evaporation rate. The deposited pheromone is decreased with increasing the pheromone evaporation rate, ρ. In experiments, the best performance is obtained at $\rho = 0.1$.

3 Proposed Algorithm

The ACS algorithm adopts a global pheromone update as well as a local pheromone update. If the global pheromone update is used, the information about the best path among all constructed paths is delivered to ants which start to search for the solution. On the other hand, the local pheromone update decreases the pheromone on the edge which is just visited by ants. Therefore, this procedure increases the probability to select the edges that have not been visited yet and it can avoid a stagnation behaviour and increase a room to find a optimum path. However, in case of a graph having a large number of vertices, it is difficult to find the optimal path by only using the heuristic method.

In the ACS, m ants are placed randomly on m cities and start to search for the optimal path. During the searching process using the local pheromone updating rule, if the current visiting city is highly probable to be included in an optimal path, the probability to find an optimum path is definitely increased by constructing the path based on the current visiting city.

In the present study, to give the more precise information to ants for constructing a optimum path, the value of η_{ij} in equation (1) is not determined just by using the distance of adjacent neighbor and it is determined by using the subpath s_w with length w, $1 \leq w \leq n$. Here w refers the number of edges including subpath. Using the information about the length of subpath s_w, we precompute the length of all city i based subpaths which can be constructed from w number

of edges, $(i,j)(j,k)...(x,t)$. Then the next visiting city is selected as a city which is located at the subpath with the minimum length among all possible subpaths. In other words, in the searching process of a next visiting city j from the city i. the tour path is constructed by selecting a city which minimize the value of s_w. According to this algorithm, the equation (1) can be modified as follows:

$$j = \begin{cases} \arg \ \max_{l \in N_i^k} \{\tau_{il}[\eta_{il}{}^{sw}]^{\beta}\}, & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{5}$$

Through numerical procedure of this algorithm, we need to make the list of the nearest neighbor first, and then we have to find the minimum distance between the neighboring cities in the list. For instance, if we assume the subpath length $w = 3$, we first compute the distance of subpath $(i,j)(j,k)(k,t)$, $d_{ij} + d_{jk} + d_{kt}$ for every city j adjacent to the current city i. Then the nearest neighbor list, l_i is arranged by sorting with an ascending order. As illustrated in the Figure 2, this procedure marginally increases the total execution time owing to its preprocessing treatment. As implied in the local pheromone updating method represented by equation (3), it is quite important how to evaluate a initial value of pheromone, τ_0 because it continuously influences the tour construction process. In the original ACS, a initial value of pheromone is obtained by $\tau_0 = 1/(nC^{mn})$. Here C^{mn} is the length of path which is constructed by the Greedy method. On the other hand, in the present proposed algorithm, a initial value of pheromone is evaluated by the following expression (6) which has a governing parameter, C^{sw}.

$$\tau_0 = 1/(nC^{sw}) \tag{6}$$

Since C^{sw} is generally smaller than C^{mn}, every path in this proposed algorithm has the much higher level of initial pheromone. Moreover, using equation (3) together with a initial value of pheromone, τ_0 governed by the subpath information C^{sw}, the present proposed algorithm can search the adjacent neighbors more precisely.

If the original ACS is applied to the TSP with large number of cities, it is very difficult to find an optimal solution. In the search procedure of optimal solution for the TSP, if the correct and optimal algorithm is applied, the shortest path is quickly found at the beginning of the search process. Otherwise, the optimum path could be constructed by gradually improving the solution through the numerous iterations. However, in case of a large TSP, it is nearly impossible to construct a optimal path by applying the iterative search for all possible paths. Therefore, it is necessary to include cities in a path which are quite probable to construct the optimal path. By setting the initial value of pheromone large, the proposed algorithm constructs the path much closer to a optimal solution at the initial stage than does the original ACS algorithm. In other words, the proposed algorithm can increase a probability to find an optimal path at the beginning of path construction stage by choosing a nearest neighbor with the subpath information. However, it is necessary to note that, if the length of subpath, w,

is set to a quite large value, it is susceptible to be a local minima. If the length of subpath, w, is set to a small value, then there is no difference with original ACS and the algorithm performance is greatly reduced. Therefore, it is very important to set s_w with the proper value. Figure 2 shows the schematics of proposed algorithm. Here the italictype parts represent the major improvements against the original ACS.

```
algorithm: Proposed ACS for TSP {

  preprocessing steps:
      construct a distance matrix;
      construct a nearest neighbor list by s_w;
  Initialize Data;
  while (not terminate) {
      compute τ_0 with s_w : τ_0 = 1/(nC^sw);
      place m ants at m cities;
      repeat (for each ant)
          apply tour construction rule to build a trail;
          apply local pheromone updating rule;
      until (construct a solution)
      apply 3-opt local search;
      apply global pheromone updating rule;
  }
}
```

Fig. 2. Proposed Algorithm

4 Experimental Results and Discussion

The proposed algorithm has been implemented into the aco-code in reference [11]. For the validation, we used the graphs in the TSPLIB library [12]. The experiments on the proposed algorithm have been performed at Enterprise RedHat 2.1 (PentiumIV 1.7 GHz, 768MB). For each test, we have chosen the parameters which were proved to yield the optimal solution from the previous experiments. These problem parameters are given as $\xi = 0.1, \rho = 0.1, \beta = 2, q_0 = 0.9$ and $m = 10$. The initial value of pheromone in the equation (6) is evaluated by using $\tau_0 = 1/(nC^{s3})$ and the information of subpath is obtained from s_3. For each ant, 100 seconds of CPU time are allocated for one search process and the path search is repeated 10 times. For each graph, the optimum and averaged value is obtained from the results of 10 executions.

Table 1 shows the results obtained from the original ACS and the proposed algorithm for the graphs with less than 1000 cities for the length of subpath 3, s_3. Here, 'Instance' represents the graph name in TSPLIB and 'Known optimal' represents the known length of the optimal path for the corresponding

graph. The 'Best' and 'Average' of original ACS denote the optimal and averaged lengths calculated by the Dorigo's algorithm[8]. On the other hand, the 'Best' and 'Average' of proposed ACS corresponds to the optimal and averaged lengths computed by the proposed algorithm. The 'NNChangeRate' in the last column represents the changing rate in the next visiting city which is determined by the proposed algorithm with s_3, versus to the original ACS. As shown in the experimental results, in case of the graphs with small number of cities, the original and proposed algorithm can find the optimal solution within the fairly short period.

Table 1. Experimental results for the graphs with less than 1000 cities

Instance	*Known*	*Original ACS*		*Proposed ACS*		*NNChange*
	Optimal	Best	Average	Best	Average	Rate(%)
att 532	27686	27686	27704.28	27686	27705.88	36.47
d 198	15780	15780	15780.19	15780	15780.1	23.23
lin 318	42029	42029	42086.48	42029	42087.58	19.18
pcb 442	50778	50778	50835.83	50778	50831.57	13.57
rat 783	8806	8806	8819.88	8806	8821.01	24.65
d 1291	50801	50801	50874.87	50801	50863.21	7.20

In case of the graph att532, experimental results obtained by the proposed algorithm indicate that the changing rate of the next visiting city is more than 35%, compared to the original ACS. This situation can be occurred when the graph has the more complexity and the large number of edges. Since the generated subpaths in this complex graph situation are rapidly increased, the possibility to change the next visiting city becomes higher. On the other hand, in case of the graph d1291 having more than 1000 cities and simple edge connection among cities, the NNChangeRate is only 7% because the probability to change the next visiting city becomes lower for the simple graph situation.

However, in case of the graph with more than 1000 cities and high complexity, it is quite seldom to find an optimal solution by employing the original ACS. Table 2 illustrates the experimental results of the graphs with more than 1000 cities. As shown in Table 2, in case of the large graph, the proposed algorithm finds the solution much closer to the optimal solution than does the original ACS. The 'Improved Rate' at the rightmost column represents the improvement rate of the searching path constructed by the proposed algorithm, compared to the original ACS. This improvement rate is evaluated by the relation, 100 - {(c-a)/(b-a) * 100}. Experimental results simulated by the proposed algorithm indicate that the only 0.5% improvement is obtained for the graph, rl 1889 and the 72.7% improvement is for the graph, r1 5915. Even if the improvement rate has a certain level of sensitivity for the specific graph, experimental results for most of the large graphs show that the proposed algorithm yields more than 30% of improvement. These experimental results suggest that, in dealing with

the large and complex graph, the proposed algorithm is much better than the original ACS in terms of efficiency and performance improvement.

Table 2. Experimental results of the graphs with more than 1000 cities

Instance	*Known*	*Original ACS*		*Proposed ACS*		*NNChange*	*Improved*
	Optimal(a)	Best(b)	Average	Best(c)	Average	Rate(%)	Rate(%)
d 1655	62128	62153	62357.89	62147	62352.75	11.0	34.0
fnl 4461	182566	186492	186986.05	186361	187032.61	29.75	3.4
pcb 3038	137694	139098	139749.38	138933	139661.64	26.37	21.8
rl 1889	316536	317349	319232.81	317345	318849.68	20.12	0.5
rl 5915	565530	576654	581050.39	575837	581286.21	21.05	72.7
u 1432	152970	153204	153579.93	153131	153612.97	1.47	31.2
vm 1748	336556	336765	337531.19	336679	337641.23	25.69	41.2
pr 2392	378032	378838	380344.11	378654	380418.47	20.03	32.8

In general, the proposed algorithm shows a good performance for the most of the graphs. However, as shown in Table 2, the performance is still sensitive to the characteristics of each graph. Since the original ACS basically adopts the greedy heuristic algorithm to search for an nearest neighbor, the search process to find a optimal path could be highly influenced by the distance from the nearest neighbor. Thus, the performance of the ACS algorithm could be improved by changing the value of parameters according to the size of graph or number of edges in the graph. In case of the graph u1432, NNChangeRate is just 1.47% but the solution obtained by the proposed algorithm is up to 31.2%. This result implies that, in this particular graph, the performance can be significantly improved by changing few cities in visiting order. In contrast to the graph u1432, the graph fnl 4461 is another extreme case. In case of the graph fnl 4461, NNChangeRate is nearly 30% and the improved rate is only 3.4%. Since a changing rate of the nearest neighbor list is quite high according to the information on subpath w_3, it can be speculated that a graph fnl 4461 could have the much higher complexity. Therefore, in this type of a complex graph, any meta-heuristic algorithms may yield the similar trend for the solution of TSP. The experimental results suggest that the proposed ACS algorithm could be improved by varying the subpath length, w according to the characteristics of graphs.

5 Conclusion

In this study, we propose an algorithm which improve the performance of the original ACS for the TSP. For the construction of tour, the original ACS search the adjacent cities first, then select a city with the minimum distance as the next visiting city. However, in order to optimally choose the next visiting city, the proposed algorithm uses the information on subpath such that the distance

of all possible subpaths with length w are precomputed and select a city having the much higher probability to construct a optimal path. If the length of subpath, w is long, there is a possibility for stagnation. Therefore, it is quite crucial to select the proper subpath length, w. In the ACS, the information on subpath s_w highly influences the initial value of pheromone, τ_0. Since the value of τ_0 is continuously used in the updating process of local pheromone, it eventually influences the tour construction process.

In case of the graphs with small number of cities, the original and proposed algorithm can find the optimal solution within the fairly short period. For the large TSP, with the same CPU time, the proposed algorithm finds the solution much closer to the optimal solution than does the original ACS. Even if the improvement rate has a certain level of sensitivity for the specific graph, experimental results for most of the large graphs show that the proposed algorithm enhances the improved rate more than 30%. For a certain graph, the solution performance has been improved up to 72.7% by utilizing the proposed algorithm. The experimental results suggest that the proposed ACS algorithm could be improved by varying the subpath length, w according to the characteristics of graphs.

References

1. M. Dorigo, V. Maniezzo and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1) : 1-13, 1996.
2. M. Dorigo and L. M. Gambardella. Ant Colonies for the Travelling Salesman Problem. *BioSystems*, 43:73-81, 1997.
3. S. Lee and T. Chung. A Study about Additional Reinforcement in Local Updating and Global Updating for Efficient Path Search in Ant Colony System. *Journal of Korean Information Processing*, 10(B):237-242, 2003.
4. M. Dorigo, G. D. Caro and L. M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3):137-172, 1999.
5. O. Gomez and B. Baran. Reasons of ACO's Success in TSP. *Proceedings of 4th International Workshop in Ant Colony Optimization and Swarm Intelligence*, LNCS 3172: 226-237, 2004.
6. L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. *Proceedings of IEEE International Conference on Evolutionary Computation*, IEEE-EC 96: 622-627, 1996.
7. M. Dorigo and T. Stutzle. Ant Colony Optimization. MIT Press, 2003.
8. E. Bonabeau, M. Dorigo and G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 1999.
9. M. Dorigo and L. M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53-66, 1997.
10. L. Bianchi, J. Knowles and N. Bowler. Local search the probabilistic salesman problem: correction to the 2-p-opt and 1-shift algorithms. Technical Reports, IDSIA-18-03, Dalle Molle Institute of Artificial Intelligent, Switzerland, 2003.
11. http://www.aco-metaheuristic.org/aco-code/
12. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/