An Early Exploratory Method to Avoid Local Minima in Ant Colony System

Thanet Satukitchai[1] and **Kietikul Jearanaitanakij**[2], Non-members

ABSTRACT

Ant Colony Optimization (ACO) is a famous technique for solving the Travelling Salesman Problem (TSP.) The first implementation of ACO is Ant System. Itcan be used to solve different combinatorial optimization problems, e.g., TSP, job-shop scheduling, quadratic assignment. However, one of its disadvantages is that it can be easily trapped into local optima. Although there is an attempt by Ant Colony System (ACS) to improve the local optima by introducing local pheromone updating rule, the chance of being trapped into local optima still persists. This paper presents an extension of ACS algorithm by modifying the construction solution phase of the algorithm, the phase that ants move and build their tours, to reduce the duplication of tours produced by ants. This modification forces ants to select unique path which has never been visited by other ants in the current iteration. As a result, the modified ACS can explore more search space than the conventional ACS. The experimental results on five standard benchmarks from TSPLIB show improvements on both the quality and the number of optimal solutions founded.

Keywords: Ant Colony Optimization, Ant Colony System, Travelling Salesman Problem, Local Optima, Exploration

1. INTRODUCTION

Ant Colony Optimization (ACO) [1] is a swarm intelligence that simulates behavior of ant colony in nature. Ants have ability to find the shortest path between their anthill and a food source. It uses indirect communication via a chemical substance called pheromone. While searching the path from a nest to a food source, an ant deposits pheromone on a path that it moves through. The pheromone intensity then attracts other ants to follow that path. ACO can be successfully applied to solve computational problem which can be reduced to a task of searching for an optimal tour within a graph, e.g., the Travelling Salesman Problem [2] (TSP.) TSP is a famous NP-hard combinatorial optimization problem. The objective is to find the shortest possible Hamiltonian cycle - the cyclic tour that composes of all cities, each of which is visited exactly once.

According to a survey article by M. Dorigo et al. [3], a list of successful implementations of Ant Colony Optimization is shown in Table 1, in chronological order. Due to the space limitation, some of them related to ACS are mentioned in this paper. The first ACO algorithm is Ant System (AS), published by Dorigo et al. [4]. The major drawback of AS is that, due to its simple pheromone updating mechanism, it can get stuck in local optima easily. There are a number of AS successors trying to overcome this weakness. One of the most successful solutions is Ant Colony System (ACS) [8, 9]. Although it introduces the local pheromone update mechanism which can reduces the chance of local optima, the construction solution phase of ACS is still not perfect

Table 1: *Successful Ant Colony Optimization Algorithms in Chronological Order.*

Algorithm	Year
Ant System (AS) [4-6]	1991
Elitist AS [4, 5]	1992
Ant-Q [7]	1995
Ant Colony System [8, 9]	1996
Max-Min AS [10-12]	1996
Rank-Based AS [13, 14]	1997
ANTS [15]	1999
BWAS [16]	2000
Hyper-Cube AS [17]	2001

Other research works improve the performance of ACS by inserting the exploratory mechanism. Beer et al. [18] introduce the antennation - an analogous process of real ant. It is a method of direct interactions between ants by using their antennae. When ants meet at a node, they share information about the search space. The information is used to force the diversification of ant paths. Malisia and Tizhoosh [19] proposed several extensions of ACS that based on the idea of opposition to expand the area of the solution space. Aljanaby et al. [20] use the concept of multiple colonies for ACS. Each colony has its own pheromone and they are working cooperatively by using an interaction technique that directs each colony to search toward the different areas of search space.

We propose another modified version of ACS by

Manuscript received on August 5, 2015.
Final manuscript received on November 1, 2015.
[1,2] The authors are with Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520 Thailand, E-mail: alphanet.lek@gmail.com and kjkietik@kmitl.ac.th

adding an exploratory mechanism in the solution construction phase to improve both the quality and the quantity of good solutions founded. Since ACS tends to favor exploitation over exploration, it is likely that the algorithm will be trapped in local optima. We introduce the exploratory step that forces ants to expand their search space. When an ant selects the next city for moving to, it is forced to move through a unique edge that has never been visited by any ant, if possible. Moreover, the exploratory step for each ant is restricted in order to prevent an excessive dispersion of search that may cause a bad quality solution. The experimental results on five symmetric TSP benchmarks from TSPLIB [21] show that our proposed method produces better quality of solutions and finds more optimal solutions than conventional ACS and another related method. The rest of this paper is organized as follows. Section 2 introduces the background of ACS. Section 3 describes the details of our proposed method. The experimental results and the comparisons with other algorithms are discussed in section 4. Finally, conclusions are provided in section 5.

2. ANT COLONY SYSTEM

The conventional ACS algorithm for solving TSP is explained in this section.

Initially, m ants are randomly distributed to n cities. They concurrently create their tours (solutions of TSP) by iteratively applying the state transition rule.

2.1 ACS State Transition Rule

When an ant k is at a city r, it selects the next city s to move by using the *pseudorandom proportional rule* as the following equation.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\} & \text{, if } q \le q_0 \\ \text{rondomly choose from } J_k(r) \text{ with probability } p_k(r,s) & \text{, otherwise} \end{cases} \quad (1)$$

Where

- $J_k(r)$ is a set of neighbors of the city r which an ant k never moves to,
- $\tau(r,u)$ is an amount of pheromone on an edge that connects from city r to city u,
- $\eta(r,u)$ is a heuristic function (inverse of distance) of an edge that connects from city r to city u,
- β is a parameter for bias value of heuristic function,
- q is a uniformly distributed random number in range [0, 1],
- q_0 is an exploitation probability parameter $(0 \le q_0 \le 1)$.

In equation (1), when $q \prec q_0$, ant chooses the next city by greedy selection, i.e., it chooses the city that has the largest product between pheromone amount τ and heuristic function η. Otherwise, it randomly chooses a city from a set of neighborhood $J_k(r)$ with the probability function p_k, as shown in equation (2). This equation implies that the city which is connected by an edge with a high level of pheromone and a short distance will be more likely to be selected.

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)] \cdot [\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [\tau(r,u)] \cdot [\eta(r,u)]^{\beta}} & \text{, if } s \in J_k(r) \\ 0 & \text{, otherwise} \end{cases} \quad (2)$$

2.2 Local Pheromone Updating Rule

During a process of building a tour, an ant moves through an edge and decreases the pheromone level on that edge by the local updating rule as the following equation.

$$\tau(r,s) \leftarrow (1-\rho) \cdot \tau(r,s) + \rho \cdot \tau_0 \quad (3)$$

Where τ_0 is an initial pheromone value and ρ is an evaporation rate $(0 \le \rho \le 1)$.

2.3 Global Pheromone Updating Rule

After all ants complete their tours, only pheromone on edges of the global best-so-far tour will be increased by the following global updating rule.

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s) \quad (4)$$

Where

$$\Delta\tau(r,s) = \begin{cases} 1/L_{gb} & \text{, if}(r,s) \in \text{global best tour} \\ 0 & \text{, otherwise} \end{cases} \quad (5)$$

- α is a deposit rate $(0 \le \alpha \le 1)$
- L_{gb} is the length of global-best tour

2.4 Summary Procedure of ACS

The steps of ACS algorithm can be summarized in Fig. 1.

Set parameters, Initialize pheromone
Loop // each loop is called an iteration
  Each ant is randomly positioned at a starting city
  Loop // each loop is called a step
    Each ant selects the next city by state transition rule (1) and then applies a local pheromone updating rule (3)
  Until all ants have built a complete solution
  Apply a global pheromone updating rule (4)
Until termination condition is met

Fig.1: *The Procedure of ACS Algorithm.*

3. THE PROPOSED METHOD

In ACS, there are two main factors on an edge that attract ants to move through. The first factor is the heuristic value which was calculated from the distance of an edge. It is hard to improve this factor since the distance of an edge is constant. Another factor is the pheromone level on an edge. The pheromone level is decreased when an edge is visited by an ant, and increased when an edge belongs to the current global best-so-far tour.

According to the state transition rule (1), ACS tends to favor exploitation over exploration. More precisely, the q_0 parameter, which represents the probability of greedy selection, is usually defined nearby 1, e.g., 0.9. At the beginning, the pheromone levels of all edges are set to an identical small-positive real number. Therefore, during early iterations, ACS tends to create tours by emphasizing only on the distance of each edge. The tour that composes of edges which have a fairly short distance can be chosen as a sub-optimal solution, which may in turn mislead as the global best-so-far tour. Consequently, the pheromone level along that tour will increase in later iterations. Due to the premature convergence problem of ACS, if there is no ant that can find a better best-so-far tour within a limited number of iterations, all ants tend to generate similar tours. This could lead to a local optima phenomenon.

Our proposed method modifies the solution construction phase of the ACS algorithm which is the phase that an ant starts to choose the next city. The basic idea is to force ants to be more exploratory during the early stage by considering only the distances of all unvisited paths connected to the current city. This modification will expand the area coverage of the search space and reduce the possibility of stagnation in local optima.

3.1 Required Additional Structures

The following additional structures are required before running the proposed method.

I. The Edge's Flag

Every edge (j, k) has a flag p_j, k that keeps the status whether that edge has been passed by any ant or not. The initial value of p_j, k is false.

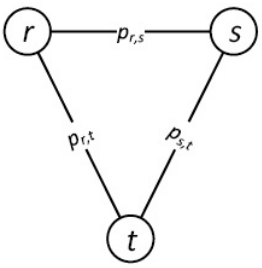

Fig.2: *Edge's Flag.*

II. The Exploratory Steps

Each ant a has a memory e_a to counts the number of its own exploratory steps. The initial value of e_a is zero.

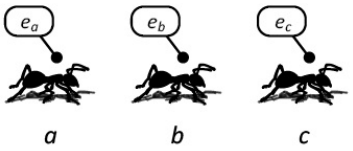

Fig.3: *Ant's Exploratory Steps.*

II. The Exploratory Steps

The global constant parameter σ limits the number of ant's exploratory steps. A proper value of parameter σ can be determined from the preliminary runs, which will be described in the later section.

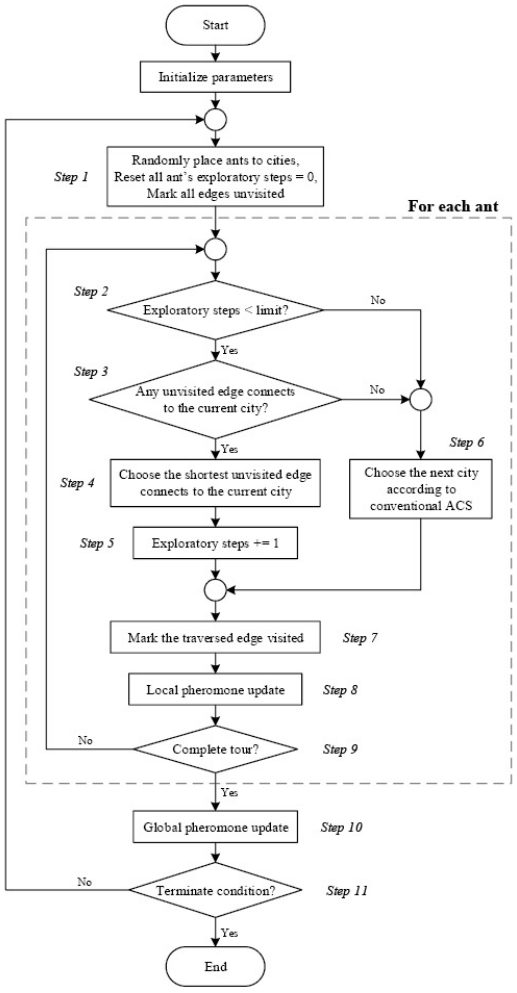

Fig.4: *Flow Chart of the Proposed Method.*

3.2 Procedure of the Proposed Method

The proposed method can be described by the following algorithm along with the flowchart in Fig.4.

Step 1. The p flags of all edges are reset to false, the exploratory steps (ea) of all ants are reset to zero and all ants are randomly placed to starting cities.
Step 2. **If not** ($ea < \sigma$) where σ is a limit of exploratory steps.
Then go to step6.
Step 3. **If** there is **no** $s \in J_a(r)$ such that $p_{r,s}$=false.
Where
r is the current city of an ant a,
$J_a(r)$ is set of neighbors of city r that an ant a has not visited yet.
Thengotostep6.
Step 4. An ant a chooses an edge (r, s) which has the shortest distance.
Step 5. Increase the exploratory steps of an ant a by one ($e_a = e_a + 1$) and gotostep7.
Step 6. An ant a chooses the next city by using ACS's state transition rule (1).
Step 7. Mark $p_{r,s} = true$, where (r, s) is an edge that an ant a just moves through.
Step 8. Apply local pheromone updating rule(3).
Step 9. **Repeat** steps 2 to 8 **until** all ants have completed their tours.
Step 10. Apply global pheromone updating rule (4).
Step 11. **Repeat** steps 1 to 10 **until** the termination condition is met.

3.3 Ant's Move

The state transition rule of the proposed method is different from the conventional ACS. An additional procedure which wraps up the original ACS's transition rule (1) has been inserted as a new rule (6). It represents steps 2-6 of the procedure in the previous subsection.

$$s = \begin{cases} \arg\max\limits_{u \in I_a(r)} \{[\eta(r,u)]^\beta\} & \text{, if } e_a \text{ and } I_a(r) \neq \varnothing \\ \text{choose from ACS state} & \text{, otherwise} \\ \text{transition rule} \end{cases} \quad (6)$$

Where $I_a(r) = \{u \in J_a(r) : p_{r,u} = false\}$

This additional procedure provides more explorations to ACS. If the number of exploratory steps does not exceed the limit and there exists available edges that have not been passed by any ant, the ant will choose the edge which has the best heuristic value. Otherwise it will use the conventional ACS state transition rule (1).

Fig. 5 illustrates an example of ant's decision in case of the number of exploratory steps does not exceed the limit. Suppose that the ant is at city r and there are three available adjacent cities s, t, and u. Since edge (r, u) is already passed by another ant in this iteration, as the flag $p_{r,u}$ was set to be true, the city u is eliminated from available choices. The ant then selects city t as the next city since its distance $d(r, t)$ is shorter than $d(r, s)$.

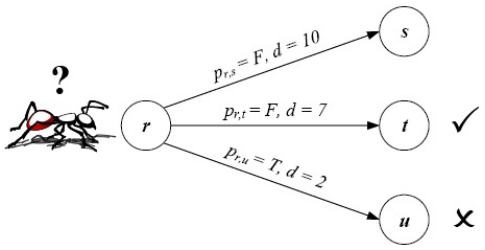

Fig.5: *An Example of Ant's Move.*

4. EXPERIMENTS

4.1 Benchmarks

We select five symmetric TSP benchmarks, which have various numbers of cities, from TSPLIB [21]. Table 2 shows the descriptions consisting of name, the number of cities, and the optimal tour length (in real value) of each benchmark.

Table 2: *TSP Benchmarks.*

Benchmark	Number of cities	Optimal Tour Length
Eil51	51	428.87
Berlin52	52	7544.37
Eil76	76	544.37
KroA100	100	21285.44
D198	198	15808.65

4.2 Preliminary Runs

Our proposed method has an additional parameter σ (exploratory steps limit) that requires an assignment before running. In order to roughly estimate a suitable value of the parameter σ, we perform the preliminary runs on all benchmarks. These preliminary runs spend less number of iterations and trials than the actual runs.

Table 3: *The possible combinations for each key size.*

σ	Benchmarks				
	Eil51	Eil76	Berlin52	KroA100	D198
1	434.98	559.90	7653.57	21680.41	16484.83
2	432.46	556.84	7693.45	21629.27	16364.61
3	431.10	556.88	7699.62	21607.75	16420.42
4	431.80	557.51	7702.03	21544.91	16495.41
5	433.46	558.83	7720.79	21704.88	16434.90

In preliminary process, we run our method for 25 trials, each with 500 iterations for Eil51, Eil76 and Berlin52, and each with 1000 iterations for KroA100. For D198, the setting is 15 trials, each with 1000 iterations.

In general, the number of trials and iterations in preliminary run is approximately 10-15 percent of the

Table 5: *ACS Parameters Setting.*

Measurement	Method	Eil51	Berlin52	Eil76	KroA100	D198
Average	ACS	431.59	7638.79	553.75	21532.59	16138.39
	Proposed	430.00	7626.81	550.37	21423.88	16077.29
Median	ACS	430.24	7544.37	553.54	21414.80	16112.92
	Proposed	428.98	7544.37	550.11	21349.44	16063.86
Min	ACS	428.87	7544.37	545.95	21285.44	15971.93
	Proposed	428.87	7544.37	544.37	21285.44	15873.92
Max	ACS	439.04	7988.19	568.13	22687.70	16916.18
	Proposed	436.73	7944.50	563.44	22308.24	16420.43
#Optimal found	ACS	5	62	0	1	0
	Proposed	12	65	6	15	0

Table 4: *ACS Parameters Setting.*

Parameter	Description	Value
m	The number of ants	10
q_0	Exploitation probability	0.9
α	Global update's parameter	0.1
ρ	Local update's parameter	0.1
β	Weight bias of heuristic function	2
τ_0	Initial pheromone value	$1/(n \cdot L_{nn})$

value used in the actual run. Table 3 shows suitable values of the parameter σ for each benchmark. It is worth to note here that we gradually increase σ and observe the total tour length in the preliminary run. If we do not notice the new best tour length for some considerable value of σ, we stop the preliminary run and return the value of σ which gives the shortest tour in each benchmark.

Other ACS parameters used in our proposed method are conformed to those used in the conventional ACS [7], as shown in Table 4; where L_{nn} is the tour length produced by the nearest neighbor heuristic and n is the number of cities.

4.3 Comparisons between Our Proposed Method and the Conventional ACS

The comparison between the proposed method and the conventional ACS algorithm [7] is discussed in this subsection. In order to have a fair comparison, the ACS algorithm and the ACS part in our proposed method are implemented without local search and candidate list.

In actual runs, we execute each algorithm 100 trials, each with 5000iterations, for Eil51, Eil76, Berlin52 and KroA100. For D198, the number of trials and iterations are 70 and 5000, respectively. Table 5 indicates that the proposed method achieves better quality of solutions in terms of both average and median value of the tour length. Moreover, the number of optimal solutions founded in the proposed method is also more than those of the conventional ACS in all benchmarks, except for the benchmark D198 which neither method reaches the optimal solution.

4.4 Visual Comparisons on the Shortest Tour

In order to perceive the visual comparisons between the conventional ACS and the proposed method, the samples of the shortest tours for all benchmarks are shown in Fig.6-13.

It is worth to note that we draw the shortest tour for each benchmark by choosing the tour serving as the median of the tour lengths among all trials. Both ACS and the proposed method can find the optimal solutions at the median for Eil51 and Berlin52 benchmarks. Therefore, they can share the same shortest tour for both benchmarks, as shown in Fig.6-7.

Since these two benchmarks contain a small number of instances, it is not surprising that both ACS and the proposed method can achieve the optimal solutions at the median.

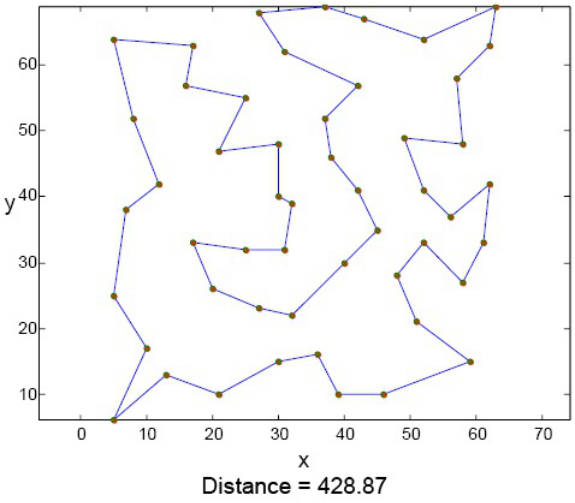

Fig.6: *The Shortest Tour of Both Methods on Eil51.*

Fig. 8 illustrates the shortest tour of ACS on Eil76 benchmark. Eil76 is a representation of the small-size benchmark which contains 76 cities. It also has an important characteristic for simulating the combinatorial optimization problems, i.e., most neighboring states are separated by similar distance. The conventional ACS achieves a fair tour distance (549.87),

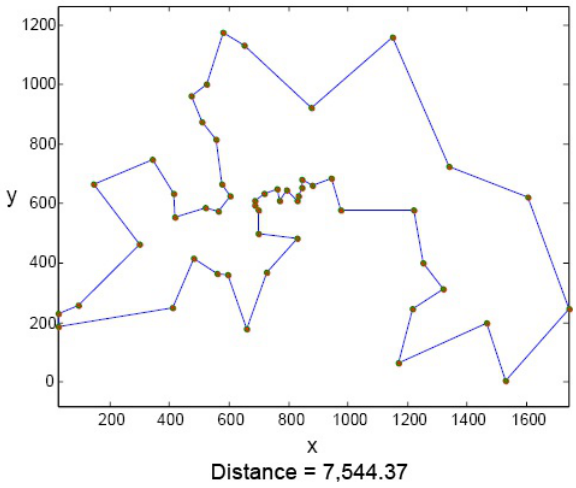

Fig.7: *The Shortest Tour of Both Methods on Berlin52.*

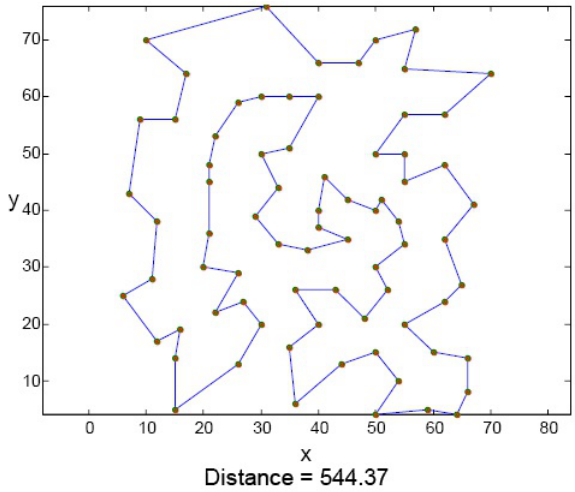

Fig.9: *The Shortest Tour of the Proposed Method on Eil76.*

comparing to the optimal tour length (544.37). However, ACS never reaches the optimal solution for Eil76 benchmark, as shown in Table 5. In contrast, Fig. 9 shows the optimal tour length produced from the proposed method. As we have noted in the previous paragraph, we show the shortest tour at the median of each method. This indicates a better performance of the proposed method which can produces the optimal solution at the median.

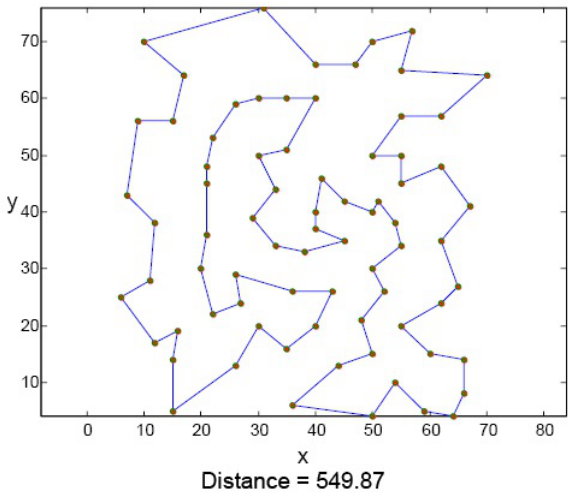

Fig.8: *The Shortest Tour of ACS on Eil76.*

The next benchmark for a visual comparison is KroA100. This benchmark contains 100 cities and has a long optimal tour length (21285.44). The shortest tours of ACS and the proposed method on KroA100 benchmark are shown in Fig. 10 and Fig.11, respectively. Both methods are competitive; indeed achieve the solutions which are very close to the optimal tour length. KroA100 is considered as the challenging benchmark since it consumes a long running time in finding the optimal solution. As the experimental results shown in Table 5, ACS can find the optimal tour length only 1 time out of 100 trials, while the proposed method produces the optimal solution more often than ACS, i.e., 15 times out of 100 trials.

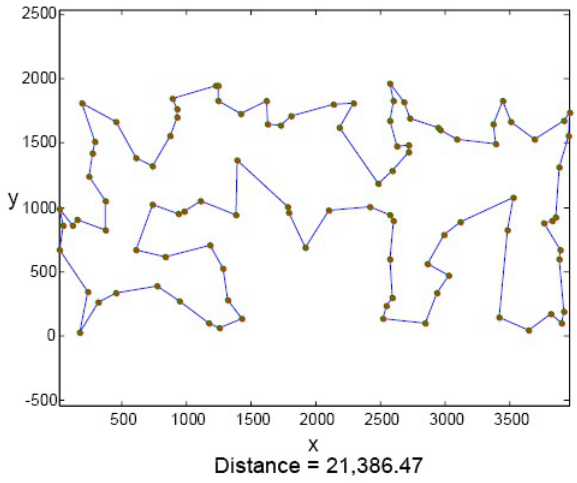

Fig.10: *The Shortest Tour of ACS on KroA100.*

The last visual comparison is taken on D198 benchmark, as shown in Fig. 12 and 13. D198 benchmark not only contains a larger number of cities, i.e., 198, but also has various kinds of the instance distributions. Besides several groups of instances clustered at the middle-top of the map, there are a number of small groups distributed around, including one instance positioned at faraway place, i.e., the origin. The overview of the shortest tour produced from both ACS and the proposed method look similar except for paths which are in the congested area. In addition, the tour distances of both methods are not significantly different since the distances are approximately close to the optimal tour length which is 15808.65. An interesting fact from this comparison is that the

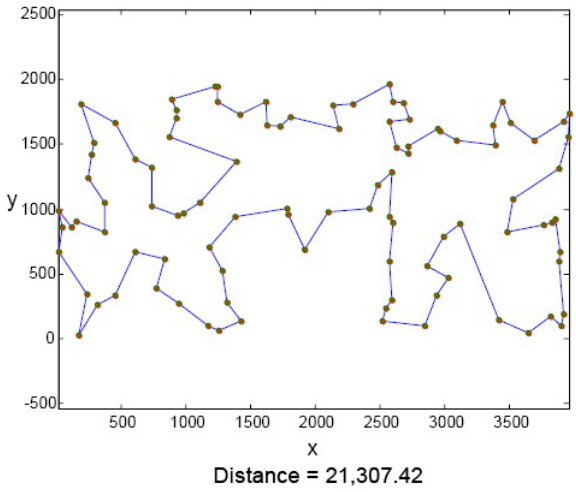

Fig.11: *The Shortest Tour of the Proposed Method on KroA100.*

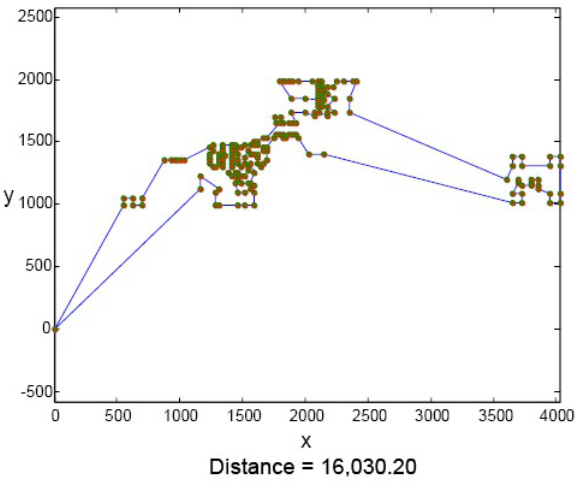

Fig.12: *The Shortest Tour of ACS on D198.*

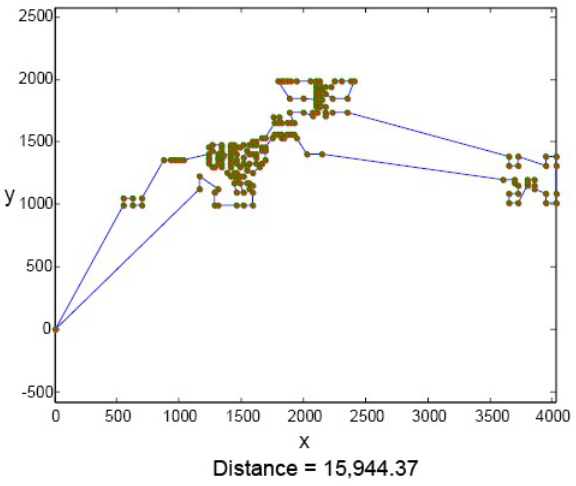

Fig.13: *The Shortest Tour of the Proposed Method on D198.*

proposed method does not make a major distinction to the tour produced by ACS. In fact, it explores a slightly better solution by performing a refined search on paths within the cluster of instances. The exploratory mechanism in the proposed method can improve the performance of ACS very well if a map does not contain an extreme distribution of the cities, e.g., D198. Otherwise their performances will be competitive.

4.5 Comparisons between Our Proposed Method and another ACS Extension

In this subsection, we compare the proposed method with another ACS extension that uses opposition-based approach called Opposite Pheromone per Node (OPN), proposed by Malisia et al. [19]. In order to be a fair comparison, all common parameters, including the number of trials and iterations, used in the proposed method are identical to those used in OPN. Other parameters are same as those used in the previous experiments.

Due to the different machine platform and implementation, we use the median ratio to compare the performance of two methods. The median ratio, in percentage unit, between the ACS-based algorithm x and conventional ACS is defined in (7), where *Optimal* is the optimal tour length of a particular benchmark (as shown in Table 2).

$$MedianRatio(x) = \frac{Median(ACS) - Median(x)}{Median(ACS) - Optimal} \times 100 \quad (7)$$

The reason of using median in (7) is that OPN uses median to benchmark their algorithm. Besides the median ratio, we also compare the number of optimal solutions found in addition to the number produced by the conventional ACS, i.e., #Additional Opt.

Table 6 shows that the proposed method has better performance in terms of median ratio in all benchmarks. Notice that the median ratio of OPN method for D198 is negative because its median is poorly larger than the conventional ACS's median. Furthermore, the proposed method achieves more additional number of optimal solutions found than OPN in most benchmarks.

Table 6: *Comparison between the Proposed Method and OPN.*

Benchmark	Measure	Proposed Method	OPN
Eil51	Median Ratio	91.97%	70.49%
	#Addtional Opt	7	2
Eil76	Median Ratio	37.40%	28.72%
	#Addtional Opt	6	3
KroA100	Median Ratio	50.53%	41.10%
	#Addtional Opt	14	6
D198	Median Ratio	16.12%	-1.96%
	#Addtional Opt	0	0

5. CONCLUSIONS

Ant Colony System (ACS) is successful nature-inspired algorithm that can be applied to the Travelling Salesman Problem. It imitates the pheromone mechanism of natural ants in order to construct the shortest solution. However, one of its disadvantages is that it has a chance of being trapped into local minima. We proposed the extension of ACS by enhancing the exploratory search. The proposed method forces ants to move differently in the early stage of iterations. A number of additional structures are inserted to memorize the number of exploratory steps, to limit the number of ant's exploratory steps, and to keep the visiting status for each edge. The experimental results on five symmetric TSPLIB benchmarks indicate better performance of the proposed method over the conventional ACS and OPN. The limitation of the proposed method is that it cannot significantly improve the shortest tour length for the problem which contains an extreme distribution of instances. One of the future works is to create a process which can determine a limit of exploratory steps (σ) automatically.

References

[1] M. Dorigo and T. Stützle, *Ant Colony Optimization*, MIT Press, Cambridge, 2004.

[2] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, NJ, 2007.

[3] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization: Artificial ants as a computational intelligence technique," *IEEE Computational Intelligence Magazine*, USA, Vol. 1, No.4, pp.28-39, 2006.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 26, No.1, pp. 29-41, 1996.

[5] M. Dorigo, *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italy, 1992.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Dipartimento di Elettronica, Politecnico di Milano, Italy, Tech. Rep. 91-016, 1991.

[7] L.M. Gambardella and M. Dorigo, "Ant-Q: A reinforcement learning approach to the travelling salesman problem," in *Proc. Twelfth International Conference on Machine Learning (ML-95)*, A. Prieditis and S. Russell, Eds., Morgan Kaufmann Publishers, pp. 252-260, 1995.

[8] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the travelling salesman problem," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No.1, pp.53-66, 1997.

[9] M. Dorigo and L.M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73-81, 1997.

[10] T. Stützle and H.H. Hoos, "Improving the Ant System: A detailed report on the $MAX-MIN$ Ant System," FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA-96-12, Aug. 1996.

[11] T. Stützle, *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications*, ser. DISKI. Infix, Sankt Augustin, Germany, vol. 220, 1999.

[12] T. Stützle and H.H. Hoos, "$MAX-MIN$ Ant System," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889-914, 2000.

[13] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the Ant System - a computational study," Institute of Management Science, University of Viena, Tech. Rep., 1997.

[14] 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.

[15] V. Maniezzo, "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem," *INFORMS Journal on Computing*, vol. 11, no. 4, pp. 358-369, 1999.

[16] O. Cordón, I.F. de Viana, F. Herrera, and L. Moreno, "A new ACO model integrating evolutionary computation concepts: The best-worst Ant System," in *Proc. ANTS 2000*, M. Dorigo et al., Eds., IRIDIA, UniversitLibre de Bruxelles, Belgium, pp. 22-29, 2000.

[17] C. Blum, A. Roli, and M. Dorigo, "HC-ACO: The hyper-cube framework for Ant Colony Optimization," in *Proc. MIC'2001-Metaheuristics International Conference*, vol. 2, Porto, Portugal, pp. 399-403, 2001.

[18] C. Beer, T. Hendtlass, and J. Montgomery, "Improving exploration in ant colony optimization with antennation," *IEEE World congress on Computational Intelligence*, Brisbane, Australia, pp. 1-8, 2012.

[19] A. R. Malisia and H. R. Tizhoosh, "Applying opposition-based ideas to the ant colony system," *IEEE Swarm Intelligence Symposium*, Honolulu, HI, pp.182-189, 2007.

[20] A. Aljanaby, K.R. Ku-Mahamud and N.M. Norwawi, "An Exploration Technique for the Interacted Multiple Ant Colonies Optimization Framework," *Interational Conference on Intelligent System, Modelling and Simulation (ISMS 2010)*, Liverpool, United Kingdom, pp.92-95, 2010.

[21] TSPLIB: `http://www.iwr.uni-heidelberg.`

de/groups/comopt/software/TSPLIB95/tsp/
.

Thanet Satukitchai received the B.E. degree in computer engineering from King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand in 2011. He is pursuing a M.E. degree in computer engineering at King Mongkut's Institute of Technology Ladkrabang. His research interests are artificial intelligence and swarm intelligence.

Kietikul Jearanaitanakij received the B.E. degree in computer engineering from King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand in 1995, and his M.S. degree in computer science from Oregon State University, USA in 1999. In 2008, he received the D.E. degree in electrical engineering from King Mongkut's Institute of Technology Ladkrabang. He now serves as an associate professor at King Mongkut's Institute of Technology Ladkrabang. His research interests include artificial intelligence and biologically inspired computing.