Ant Colony System with Heuristic Function for the Travelling Salesman Problem

[1]Mustafa Muwafak Alobaedy, [2]Ku Ruhana Ku-Mahamud
[1, 2] *School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010 Sintok, Kedah, Malaysia, new.technology@hotmail.com, ruhana@uum.edu.my*

Abstract

Ant colony system which is classified as a meta-heuristic algorithm is considered as one of the best optimization algorithm for solving different type of NP-Hard problem including the travelling salesman problem. A heuristic function in the Ant colony system uses pheromone and distance values to produce heuristic values in solving the travelling salesman problem. However, the heuristic values are not updated in the entire process to reflect the knowledge discovered by ants while moving from city to city. This paper presents the work on enhancing the heuristic function in ant colony system in order to reflect the new information discovered by the ants. Experimental results showed that enhanced algorithm provides better results than classical ant colony system in term of best, average and standard of the best tour length.

Keywords: *Ant Colony Optimization, Ant Colony System, Heuristic Function, Traveling Salesman Problem*

1. Introduction

Biological ants have the ability to discover the shortest route from the nest to the source of food [1]. Although they do not have an advanced vision system [2], they have the ability to communicate with the environment. Ants use a chemical substance called a "*pheromone*" to communicate with the environment and between each other [3]. Pheromone substance has an evaporation property which is a powerful mechanism to update the route information. While an ant moves looking for food, it deposits a pheromone along the path. The following ant will, more likely, select the route with richer pheromones. This mechanism will make the ant choose the shortest path. In 1992, Marco Dorigo proposed the first Ant Colony Optimization (ACO) algorithm to search for an optimal solution in graphs to solve optimization problems such as the travelling salesman problem, job scheduling and network routing [1]. The variants of ACO are: (i) Ant System (AS) [4] [5] [6]. (ii) The first improvement on the ant system, called the Elitist strategy for Ant System (EAS) [7]. The improvement was done by providing strong additional reinforcement to the arcs belonging to the best tour found since the start of the algorithm. (iii) Rank-Based Ant System (AS_{rank}), another improvement over ant system is the rank-based version of ant system introduced by [8]. In $AS_{rank,}$ each ant deposits an amount of pheromone that decreases with its rank. This is similar to EAS, where the best-so-far ant always deposits the largest amount of pheromone. (iv) Max-Min Ant System (MMAS) is another variant of ACO, this algorithm has four direct improvements over AS [9] [10]. MMAS strongly exploits the best tours found, limits the possible range of pheromone trail values to the interval [t_{min}, t_{max}], the pheromone trails are initialized to the upper pheromone trail limit, and pheromone value is reinitialized each time the system approaches stagnation or when no improved tour has been generated for a specific number of iterations. (v) Ant Colony System (ACS), this improvement has been introduced by [3] [11] to improve the performance of AS.

In the ACS algorithm, ants apply exploitation and exploration mechanisms when they select the next node to move to. In addition, ACS applies local pheromone updates and global pheromone updates to direct the search for the next iteration. The global update is calculated based on the quality of the best solution so far while the local update applies an evaporation concept. ACS is used to solve the Travelling Salesman Problem (TSP) [3] [11] [12]. TSP is one of the most typical NP-Hard problems in the optimization field [13]. In TSP there are a set of cities {C_1, C_2, C…, C_N} and for each pair of cities, the distance is *d (Ci, Cj)*. The solution for this problem is to find the shortest tour, which is to find a permutation π of the cities that minimizes the quantity $\sum_{i=1}^{N-1} d\big(c\,\pi(i), c\,\pi(i+1)\big) + d\big(c\,\pi(N), c\,\pi(1)\big)$. Such a problem is an optimization problem. Therefore, an approximation

algorithm is required to find near optimal solutions rather than optimal solutions. ACS algorithm is one of those algorithms used to solve the travelling salesman problem.

Variants of ACO algorithm have been derived and extended to exploit the search history without losing the chance of exploring new areas of the search space. Among them ACS algorithm appears to be promising to extend the framework of ACO. It provides good opportunity to explore wide area of the search space in reasonable time. All variants of ACO algorithm have some similarity in their foundation such as utilize the heuristic information and pheromone value. The solution is based on constructing. All the ACO algorithms apply pheromone evaporation. Ant colony optimization algorithm has the ability to hybrid with other heuristic and meta-heuristic algorithms to solve different types of NP-hard problems. In [14], hybrid ant colony and genetic algorithm was used to solve vehicle scheduling problem. Another study conducted by [15] presented ant system-assisted genetic algorithm for solving travelling salesman problem. In addition, hybrid ant colony algorithm was used in task scheduling problem by [16]. Ant colony optimization algorithm also used on resource allocation by [17].

In this paper, enhancement of the ACS algorithm to solve the TSP is presented. The rest of the paper is organized as follows. Section II describes the structure of the ACS algorithm while Section III presents the proposed enhanced ACS algorithm. Section IV discusses the implementation of the proposed algorithm in solving TSP and Section V presents the experimental results. Concluding remarks and future works are presented in Section VI.

2. Ant Colony System

Ant Colony System (ACS) is introduced by Dorigo and Gambardella [3] [11] to improve the performance of AS. ACS differs in three main aspects from ant system. First, ACS uses a more aggressive action choice rule than AS. Second, pheromone is added only to moves belonging to global-best solution. Third, each time an ant moves on a path, it removes some pheromone from that path. The three main phases of the ACS algorithm constitute the ants' solution construction, global pheromone trail update and local pheromone trail update. ACS algorithm starts the tour construction when ant moves from node to node. Ant will choose the node using one of the two rules. First rule called pseudorandom proportional rule which is based on exploitation mechanism. Second rule uses exploration mechanism which is based on the probability distribution used in AS. The tuning between exploitation and exploration is controlled by a parameter. ACS algorithm applies global pheromone trail update where only one ant (the best-so-far ant) is allowed to add pheromone after all ants finished constructing their tours. In addition, ACS algorithm applies local pheromone trail update. In this update, all ants apply local pheromone update rule immediately after moving on arcs during the tour construction using the evaporation concept.

Dorigo and Gambardella [3] used the ant colony system to solve the travelling salesman problem. The algorithm in their study consists of three parts. The first part of the algorithm deals with the exploitation of the environment experience discovered by the ants using an aggressive action choice rule. That is, when ant k wants to move from city *i* to city *j*, it will choose the city using a rule called the *pseudorandom proportional* rule, given by:

$$j = \begin{cases} \text{argmax}_{l \in N_i^k} \{ [\tau_{il}] [\eta_{il}]^{\beta} \}, & \text{if } q \leq q0; \\ J, & \text{otherwise} \end{cases}$$

where q is a random variable uniformly distributed in [0, 1], q0 ($0 \leq q0 \leq 1$) is a parameter, and J is a random variable selected according to the probability distribution given by the following formula:

$$P_{ij}^k = \frac{[\tau_{ij}] \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta}}, \text{ if } j \in N_i^k.$$

The second part of the algorithm deals with a global update which is the mechanism of pheromone evaporation and pheromone deposit on the arcs of the best-so-far tour. In ACS only one ant (the best-so-far ant) is allowed to add pheromones after completing the iteration. The update is implemented using the following equation:

$$\tau_{ij} \leftarrow (1-P)\tau_{ij} + p\Delta\tau_{ij}^{bs},\ \forall(i,j) \in T^{bs}$$

where P is the evaporation rate, and $\Delta\tau_{ij}^{bs} = 1/ C^{bs}$. The third part of algorithm deals with local updates which occur each time an ant moves on the arc (i, j) to move from city to city, this process will remove some pheromones from the arc to increase the probability of exploring another path. In local update the ants apply the update rule immediately after moving on the arc (i, j) during the tour construction using the following rule:

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0$$

where ξ, $0 < \xi < 1$, and τ_0 are two parameters. The value of τ_0 is equal to the initial value for the pheromone trail. In the ACS algorithm, the updating functions focus on pheromones only and neglect the heuristic value in the whole process of the algorithm.

Enhancement of the ACS algorithm has been proposed by many researchers to solve the optimization problem such as those in [18] [19] [20] [21] [22]. They conducted different studies to enhance the ACS algorithm in order to solve TSP. In their works, they propose many ideas to increase the algorithm performance. However, all the studies focus on local and global pheromone update functions. In general, problems are modeled as graphs that consist of nodes and edges. The ACS algorithm utilizes the values between the edges as a heuristic value for the calculation of probability to choose the next node. However, the heuristic value is not updated at any time during execution. Such a condition is a contradiction to the concept of heuristics. According to [23] the word "heuristic" comes from Greek and means "to know", "to find", "to discover" or "to guide an investigation". Therefore, an update function for heuristic value is needed to reflect the new information discovered by the ants. Hence, this paper presents the work that updates the heuristic value based on the tour quality to improve the algorithm performance.

3. The Enhanced Ant Colony System Algorithm

The enhanced ACS algorithm integrates a new heuristic function that will update the heuristic value every time the ants find a better solution in the iteration. This is done to reflect the new status of the solution. After the ant constructs its solution, a global update process will be applied to update the best-so-far solution. This event will change the environment for the next iteration. A function will be triggered at this moment to reflect this change and thus a new heuristic value will be obtained. The new information will be applied to the best-so-far edge. The pseudo-code for the new heuristic function is shown in Fig. 1.

```
Step 0: for each path in the best tour do step 1 to 2
        Step 1: if path i (i = 1, 2, n) is not updated before do step 2
                Step 2: ηi = ηi + (δ/ best-so-far tour)     // δ is parameter from (0-10)
End
```

Figure 1. Pseudo-code for the new heuristic function

By applying this function, the heuristic values will change according to the quality of the best-so-far solution. Best solution will increase the heuristic value and vice versa. The parameter δ will determine how much the influence of the updating value should be applied to the heuristic value. If $\delta = 0$ then no update will occur which reflects the heuristic value in the classical ACS. The heuristic value on each edge will be updated only once during the whole process if it is belongs to the best-so-far edge. This condition will eliminate the issue of stagnation that may occur if the heuristic value updates more than once. After conducting many experiments, $\delta = 0.5$ is found to produce good results. However, this depends on the problem domain and dimensions.

The Enhanced Heuristic Function in Ant Colony System (EHF_ACS) algorithm flowchart is depicted in Fig. 2. While Fig. 3 shows the difference between EHF_ACS algorithm and the classical

ACS is the application of the enhanced heuristic function after global pheromone update activity is performed.

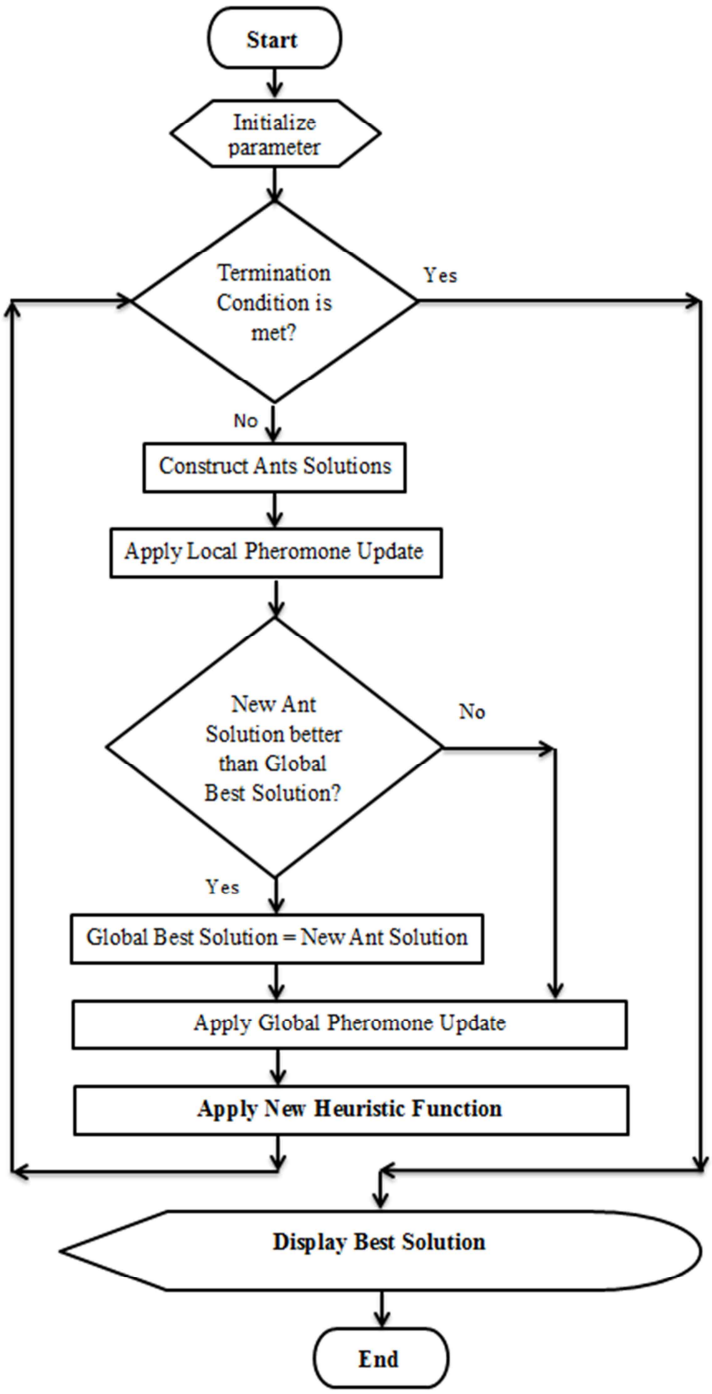

Figure 2. EHF_ACS algorithm flowchart

```
Procedure EHF_ACS
    Initialize parameters
    While (termination condition not met) do
        Construct Ants Solutions
        Apply Local Pheromone Update
            If (New Ant Solution better than Global Best Solution)
                Global Best Solution = New Ant Solution
        Apply Global Pheromone Update
        Apply New Heuristic Function
    End - While
End - Procedure
```

Figure 3. The EHF_ACS algorithm

4. EHF_ACS Algorithm for Travelling Salesman Problem

One of the most topical problems in optimization filed is travelling salesman problem. This is due to its simplicity in definition and constraints it has. William J. Cook wrote: "The TSP is an easy enough task from one perspective: there are only finitely many possible routes through a given set of cities" [24]. In addition, Alan Turing quotes: "Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit (by car), how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily (given the methods I know) find a solution" [24]. Therefore, TSP was chosen in this study to verify our enhanced algorithm.

In order to apply EHF_ACS to the travelling salesman problem, several initializations will have to be performed as follows:

(i) Calculate distance between cities using Euclidean distance method.
(ii) Calculate heuristic values using heuristic function (1/distance).
(iii) Initialize pheromone on all paths using the method (1 / (No of cities * nearest neighbor solution).
(iv) Set the variables: alpha (α) = 1, beta (β) = 2, delta (δ) = 0.5, q = 0.9, evaporation rate (p) = 0.1, $0 < p \leq 1$, number of ants = 10, and number of iteration = 10000.

If $\alpha = 0$, the closest cities are more likely to be selected. This corresponds to the classic stochastic greedy algorithm with multiple starting points since ants are initially randomly distributed over the cities. If $\beta = 0$, only pheromone amplification is at work. In other word, only pheromone is used without any heuristic bias. This generally leads to poor results and in particular, for values of $\alpha > 1$ it leads to the rapid emergence of a stagnation situation. This is a situation in which all the ants follow the same path and construct the same tour, which, in general, is strongly suboptimal.

$\delta = 0.5$ is used in all experiments. If $\delta = 0$, then no change in heuristic value will happen. If $\delta > 10$, the ant will be bias to this path which will affect the behavior of the algorithm. The parameter p is used to avoid unlimited accumulation of pheromone on any trails and it enables the algorithm to ''forget'' bad decisions previously taken. No evaporation is applied if $p = 0$, and all pheromone has evaporated if $p = 1$.

Ants will be randomly distributed to cities after the initialization process. All ants will move concurrently and each ant will start building a solution which is a function of the distance between the cities. Each time an ant moves from a city to the next city, the pheromone on that connection (edge) will be evaporated using a local update mechanism. After all ants have constructed their solutions, the best solution will be selected based on the shortest tour. The best solution will be saved as global best solution if it is better than the current global best solution. A global update will be applied at this step using the global best solution. The benefit from global update is to increase the probability of selecting the same city (or the edge) for the next iteration.

The function of the local update is to reduce the probability of selecting the same city (or edge) for the following ant. Local update helps to reduce stagnation problems when sometimes the ACS algorithm does not show a convergence behavior. In other words, ants do not converge to the generation of a common path [3] [11]. Local update also helps to increase the exploration mechanism.

The heuristic function will start immediately after the global update in order to update the heuristic value. Fig. 4 shows the EHF_ACS algorithm implementation for TSP.

```
Procedure EHF_ACS for TSP
    Step 0: Read TSP file (Coordinate points X & Y);
    Step 1: Calculate distance;
    Step 2: Calculate heuristic values;
    Step 3: Initialize pheromone;
    Step 4: Set Algorithm parameters;
    Step 5: Initialize ants array;
    Step 6: While (termination condition not met) do steps 7-15
        Step 7: For each ant (ant[i], i = 0, 1, …, m) do step 8;
            Step 8: Create Thread;  /// one thread for each ant to move in parallel
        End - For
        Step 9: For each Thread (Thread i, i = 0, 1, …, m) do Steps 10-11;
            Step 10: Construct ant[i] Tour;
            Step 11: Apply Local Pheromone Update;
        End - For
        Step12: If (Best Ant_i tour is shorter than Global Best tour) Do Step 13
            Step 13: Global Best tour = Best Ant_i tour;
        Step 14: Apply Global Update Pheromone;
        Step 15: Apply New Heuristic Function;
    End – End step 6
End - Procedure
```

Figure 4. EHF_ACS algorithm for TSP

The quality of each ant solution is measured using the tour length. In this case, the shorter tour length the better is the solution quality. After completing all iterations, the heuristic value is updated for each edge that has not been updated before.

5. Experimental Results on Travelling Salesman Problem

Eight data sets from TSPLIP with different sizes were used in the experiments to test the performance of the algorithm. The Core i7-2600 CPU @ 3.4 GHz machine with 8 GB RAM was used in conducting the experiments. The ACS algorithm and EHF_ACS algorithm were implemented using C# programming language. The multi-thread concept was used in the implementation of the algorithms to enable the parallel movement of ants while constructing their tours.

Experiments were performed to test the validity of the ACS algorithm implementation, and comparison of results was completed with [3] [11]. The settings of the parameters are as follows:

$$\alpha = 1, \beta = 2, \delta = 0.5, q = 0.9, m = 10, p = 0.1, \tau_0 = 1/(N*nn),$$

where τ_0 is the initial pheromone value, N = number of cities and nn = nearest neighbour.

The simulator development process starts with the development of the classical ant colony system algorithm for TSP problem from TSPLIP. Fig. 5 shows the console application for solving 51 cities problem. In order to validate the simulator, experiments on different TSP data set were conducted and the results are compared with previous study from [3] [11]. After validating the simulator, the enhanced ant colony system algorithm for travelling salesman problem is implemented in the simulator.

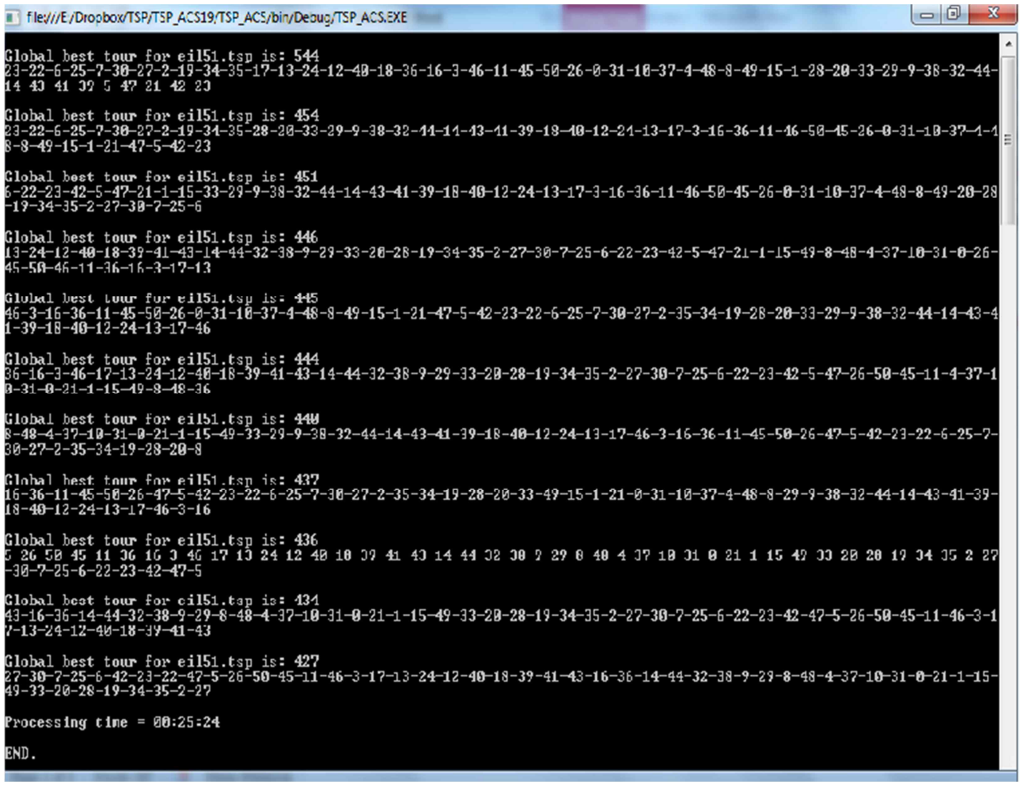

Figure 5. 오류! 지정한 스타일은 사용되지 않습니다.. Console application for TSP

Experiments were then performed to test credibility of the enhanced algorithm. TSPLIB data sets were used in the experiments. Table 1 shows the comparisons of results of the enhanced algorithm with Best Known Solution and ASC results from previous studies [21][25][26][27].

The results show that the enhanced algorithm produces better solutions quality in terms of best and mean tours, and smaller standard deviation (SD). Seven (7) mean tour results obtained by the enhanced algorithm are better than ACS and for the best tour results, the enhanced algorithm is at par with ACS. The mean and SD shows the robustness of the enhanced algorithm and its ability to guide the ants to quickly converge to the best solution. Each data set was run five times to calculate the mean and SD. All the experiments using EHF_ACS produced good solutions with minor differences between the runs.

Table 1. Performance of EHF_ACS algorithm on TSP

TSP instance	Optimum	ACS			EHF_ACS			|Mean ACS – Mean EHF_ACS|%
		Mean	SD	Best	Mean	SD	Best	
att48	33522	35595	*---	33780	**33614.4**	**43.135**	**33587**	5.56 %
eil51	426	428.21	2.05	426	**428**	**1.095**	**426**	0.05 %
st70	675	682.50	2.82	677	**677.2**	**0.748**	**676**	0.78 %
eil76	538	**541.55**	2.97	**538**	545.2	**1.469**	543	0.67 %
rat99	1211	1219.60	6.45	1211	**1212.6**	**0.8**	**1211**	0.57 %
kroA100	21282	21441.30	112.13	21315	**21297.2**	**11.51**	**21282**	0.67 %
eil101	629	640.67	5.86	**630**	**633**	**2.449**	631	1.20 %
rat195	2323	2352.76	15.79	**2334**	**2347**	**4.147**	2342	0.24 %

a. SD is not calculated in the original study.

The mean tour length differences between EHF_ACS and ACS are summarized in Table 2. The results show that the enhanced algorithm was able to improve the quality of solutions in terms of tour length as high as 5.5 % in att48 instances. The classical ACS algorithm produced better results than the enhanced algorithm when ei176 instance was used. However, the enhanced algorithm produces better results for standard deviation in all instances which implied that better results are produced by the enhanced algorithm. Fig. 6 represents the mean tour length of all instance using ACS and EHF_ACS. The enhanced algorithm was able to find shorter mean tour than the classical ACS in seven instances.

Table 2. Summary of mean tour differences

Instances	Differences in Mean Tour Length	% in Difference
att48	1980.6	5.56 %
eil51	0.21	0.05 %
st70	5.3	0.78 %
eil76	3.65	0.67 %
rat99	7	0.57 %
kroA100	144.1	0.67 %
eil101	7.67	1.20 %
rat195	5.76	0.24 %

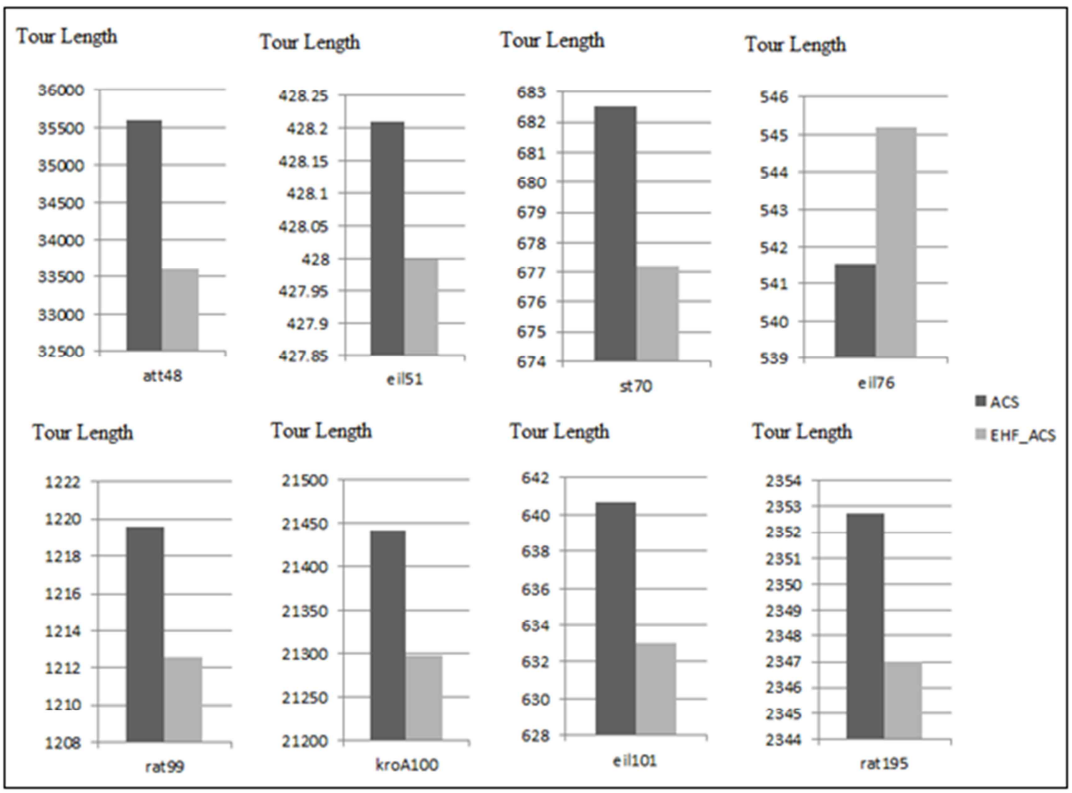

Figure 6. Mean tour length of all instance using ACS and EHF_ACS

6. Conclusion

The enhanced heuristic function presented in this paper was able to reflect the new heuristic information obtained during the implementation of the ant colony system algorithm to solve the travelling salesman problems. This new information represents the heuristic experience gained by the ants while moving along the paths between cities. The new information will guide ants in their decision for the next iteration. . No stagnation will occur when the updating activity is fixed to only one time for each edge in the whole process. The enhanced algorithm outperforms ant colony system algorithm in term of shortest tour, mean, and standard deviation. Future work can focus on improvement in the data structure where better solutions can be obtained within a shorter time.

7. Acknowledgment

The authors wish to thank the Ministry of Higher Education Malaysia for funding this study under Fundamental Research Grant Scheme, S/O code 11980 and RIMC, Universiti Utara Malaysia, Kedah for the administration of this study.

8. References

[1] M. Dorigo and T. Stutzle. Ant colony optimization. Cambridge, Massachusetts, London, England: MIT Press, 2004.

[2] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraula and E. Bonabeau. Self-Organization in Biological Systems. New Jersey: Princeton University Press, 2003.

[3] M. Dorigo and L.M. Gambardella. “Ant Colonies for the Travelling Salesman Problem”. BioSystems Journal, Vol. 43(2), pp. 73-81, 1997.

[4] A. Colorni, M. Dorigo and V. Maniezzo. Distributed “Optimization by Ant Colonies”. Proceedings of the first European Conference on Artificial Life, Cambridge, pp. 134-142, 1991.

[5] M. Dorigo, V. Maniezzo and A. Colorni. “Positive Feedback as a Search Strategy”. Milano, Italy: Politecnico di Milano. (Report No. 91- 016), 1991.

[6] M. Dorigo, V. Maniezzo and A. Colorni. “Ant system: optimization by a colony of cooperating agents”. Journal of Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE, Vol. 26(1), pp. 29-41, 1996.

[7] M. Dorigo. “Optimization, learning and natural algorithms” (Unpublished doctoral dissertation), Politecnico di Milano, Italy. 1992.

[8] B. Bullnheimer, R. F. Hartl and C. Strauss. “A New Rank-Based Version of the Ant System: A Computational Study”. Central European for Operations Research and Economics, Vol. 7(1), pp. 25-38, 1999.

[9] T. Stutzle. “MAX-MIN ant system for quadratic assignment problems”. Germany: Intellektik Group, Department of Computer Science, Darmstadt University of Technology (Report No. AIDA-97-04), 1997.

[10] T. Stutzle and H. Hoos. “MAX-MIN ant system and local search for the traveling salesman problem,” Proceedings of IEEE International Conference on Evolutionary Computation. Indianapolis, pp. 309-314, 1997.

[11] M. Dorigo and L. M. Gambardella. “Ant colony system: a cooperative learning approach to the traveling salesman problem”. IEEE Transactions on Evolutionary Computation, Vol. 1(1), pp. 53-66, 1997.

[12] L. Gambardella and M. Dorigo. “Solving symmetric and asymmetric TSPs by ant colonies”. Proceedings of the IEEE Conference on Evolutionary Computation, Japan, pp. 622-627, 1996.

[13] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy kan, and D. B. Shmoys, The Traveling Salesman Problem, Eastbourne: John Wiley & Sons, 1985.

[14] S. Zhang, T. Ning and Z. Zhang. “A New Hybrid Ant Colony Algorithm for Solving Vehicle Scheduling Problem”, International Journal of Advancement in Computing Technology (IJACT), vol. 4, no. 5, pp. 17-23, 2012.

[15] G. Dong, X. Fu and H. Xue. "An Ant System-Assisted Genetic Algorithm For Solving The traveling Salesman Problem", International Journal of Advancement in Computing Technology (IJACT), vol. 4, no. 5, pp. 165-171, 2012.
[16] X. Wei. "Study of Ant Colony Hybrid Algorithm in Grid Task Scheduling", Advance in Information Science and Service Sciences (AISS), vol. 4, no. 5, pp. 325-331, 2012.
[17] Z. Yunxia. "Study on the Resource Allocation Algorithm Based on Ant Colony Optimization", Journal of Convergence Information Technology (JCIT), vol. 7, no. 16, pp. 214-223, 2012.
[18] S. Ilie and C. Badica. "Effectiveness of Solving Traveling Salesman Problem Using Ant Colony Optimization on Distributed Multi-Agent Middleware," Proceedings of the 2010 International Multiconference on Computer Science and Information Technology, Wisla, Poland, pp. 197-203, 2010.
[19] M. Lianming. "An Efficient Ant Colony System for solving the new Generalized Traveling Salesman Problem," Proceedings of the IEEE International Conference on Cloud Computing and Intelligent Systems, 2011, Beijing, China, pp. 401-412, 2011.
[20] L. Liqiang, S. Yang and D. Yuntao. "Cooperative Multi-ant Colony Pseudo-parallel Optimization Algorithm," Proceedings of the International Conference on Information and Automation, China, pp. 1269-1274, 2010.
[21] Y. Wei-Jie, H. Xiao-min, Z. Jun and H. Rui-Zhang. "Self-Adaptive ant colony system for the traveling salesman problem," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, Texas, USA, pp. 1399-1404, 2009.
[22] L. Xiaojiang, L. Jiapin and C. Min. "Ant colony algorithm for large scale TSP". Proceedings of the International Conference on Electrical and Control Engineering, Yichang, China, pp.573-576, 2011.
[23] R. Sarker, H. A. Abvbass and C. Newton. "Heuristic and optimization for knowledge discovery". United States of America: Idea Group Pub, 2002.
[24] W. J. Cook. "In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation", Princeton University Press, USA, 2011.
[25] A. Aljanaby, K. R. Ku-Mahamud and N. M. Norwawi. "Optimizing Large Scale Combinatorial Problems Using multiple Ant Colonies Algorithm Based on Pheromone Evaluation Technique." International Journal of Computer Science and Network Security, vol. 8(10), pp. 54-58, 2008.
[26] L. Kangshun, K. lanlan, Z. Wensheng and L. Bing. "Comparative analysis of genetic algorithm and ant colony algorithm on solving traveling salesman problem." Proceedings of the First IEEE International Workshop on Semantic Computing and Systems, Huangshan, China, pp. 72-75, 2008.
[27] Y. Wei-Jie and Z. Jun. "Pheromone-distribution-based adaptive ant colony system." Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, New York, USA, pp. 31-38, 2010.