NEW HEURISTIC FUNCTION IN ANT COLONY SYSTEM ALGORITHM

Ku Ruhana Ku-Mahamud[1], Aniza Mohamed Din[2], Yuhanis Yusof[3], Massudi Mahmuddin[4], and Mustafa Mufawak Alobaedy[5]

[1]*Universiti Utara Malaysia, Malaysia,* ruhana@uum.edu.my
[2]*Universiti Utara Malaysia, Malaysia,* anizamd@uum.edu.my
[3]*Universiti Utara Malaysia, Malaysia,* yuhanis@uum.edu.my
[4]*Universiti Utara Malaysia, Malaysia,* ady@uum.edu.my
[5]*Universiti Utara Malaysia, Malaysia,* new.technology@hotmail.com

ABSTRACT. NP-hard problem can be solved by Ant Colony System (ACS) algorithm. However, ACS suffers from pheromone stagnation problem, a situation when all ants converge quickly to one sub-optimal solution. ACS algorithm utilizes the value between nodes as heuristic value to calculate the probability of choosing the next node. However, the heuristic value is not updated throughout the process to reflect new information discovered by the ants. This paper proposes a new heuristic function for the Ant Colony System algorithm that can reflect new information discovered by ants. The credibility of the new function was tested on travelling salesman and grid computing problems. Promising results were obtained when compared to classical ACS algorithm in terms of best tour length for the travelling salesman problem. Better results were also obtained for the grid scheduling problem in terms of makespan and utilization.

Keywords: Ant colony system, heuristic function, traveling salesman problem, grid scheduling.

INTRODUCTION

Marco Dorigo presented the first ACO algorithm in 1992, to solve optimization problems such as travelling salesman problem, job scheduling and network routing (Dorigo & Stutzle, 2004). Other variants of ACO are (i) Ant System (AS) (Colorni et al., 1991). (ii) Elitist Ant System (EAS) (Dorigo, 1992), the first improvement on ant system in providing strong additional reinforcement to arcs belonging to the best tour found since the start of the algorithm. (iii) Rank-Based Ant System (AS_{rank}) (Bullnheimer, 1999), another improvement over AS where 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) (Stutzle, 1997; Stutzle & Hoos, 1997) that has four direct improvements over AS. 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 specific number of iterations. (v) Ant Colony System (ACS) (Dorigo & Gambardella, 1997a; 1997b). This variant allows the ants to apply exploitation and exploration mechanisms when they select the next node to move. In addition, ACS applies local pheromone update and global pheromone update to direct the search for next iteration. Global update is calculated based on the quality of the best solution so far while the local update apply evaporation concept.

In grid environment, the scheduler is responsible to find suitable resource to process specific job. Jobs are submitted to the grid system by users. The scheduler will assign jobs to resources according to the scheduling algorithm adopted by the grid system. Job scheduling problem can be presented as a directed graph (digraph) with nodes and directed edges. Resources and jobs are represented as nodes and the directed edges represent the processing cost and pheromone.

Job scheduling problem is different from traveling salesman problem where in traveling salesman problem, the graph is complete and undirected because every pair of distinct vertices is connected by a unique edge. In job scheduling, graph the connections (edges) are only between resources and jobs and directed from resources to jobs. There is no connection (edge) between resources and resources or jobs and jobs.

RESEARCH FRAMEWORK

The research framework starts with formulating a heuristic update function to reflect the change in the heuristic information of ACS algorithm. The second stage is the development of the enhanced ACS algorithm followed by the testing of the performance of the enhanced algorithm in solving the travelling salesman problem. This is followed by the development of the simulator and finally, testing the performance of the enhanced ACS algorithm in solving grid job scheduling problem. Figure 1 depicts the framework for this research.

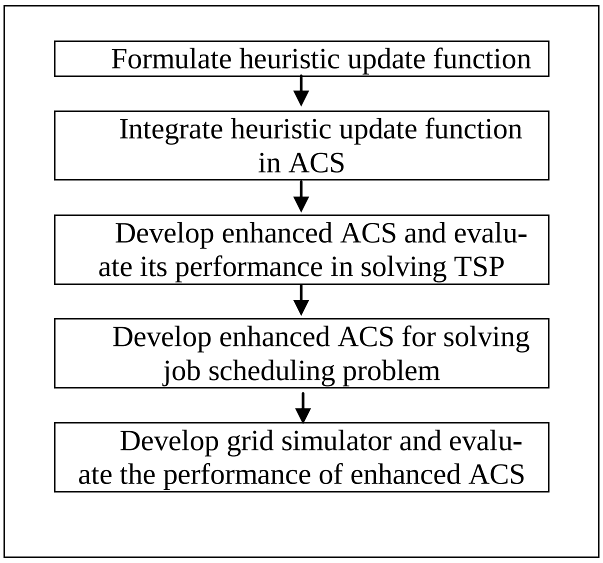
Formulate heuristic update function
Integrate heuristic update function in ACS
Develop enhanced ACS and evaluate its performance in solving TSP
Develop enhanced ACS for solving job scheduling problem
Develop grid simulator and evaluate the performance of enhanced ACS

Figure **Error! No text of specified style in document.**1. Research framework

NEW HEURISTIC FUNCTION

ACS algorithms utilize the values between the edges to use it as heuristic value for the calculation of probability to choose the next node. However, heuristic value is not updated at any time during execution. Such situation contradicts the concept of heuristic. A new a function for heuristic value is needed to reflect the new information discovered by the ants. The pseudo-code for the function is as shown in Figure 2.

```
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: $\eta_i = \eta_i + (\delta /$ best-so-far tour$)$  // $\delta$ is parameter from (0-10)
End
```

Figure 2. Pseudo-code for the new heuristic function.

After ant constructed 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. The new function will be triggered 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 every time the ants find a better solution in the iteration. 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 one time during the whole process if it belongs to the best-so-far edge. This condition will eliminate the issue of stagnation that may occur if the heuristic value is updated more than one time. After conducting many experiments, it is found that $\delta = 0.5$ will produce good results. However, this depends on the problem domain and dimensions. The new heuristic function has been integrated in the classical ACS algorithm thus enhancing it. The enhanced ACS algorithm is known as EHF_ACS.

In EHF_ACS, ants will be randomly distributed to nodes (cities or resources) 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 in the case of TSP or the processing all jobs using all resources for the grid scheduling case. Each time an ant moves from a node to the next node, the pheromone on that connection (edge) will be evaporated based on local update mechanism. The benefit of local update function is to reduce the probability of selecting the same resource (node) by the following ant. Local update also helps to reduce stagnation problem and increase exploration mechanism when sometime ACS algorithm does not show a convergence behavior (i.e., ants do not converge to the generation of a common path) (Dorigo & Gambardella, 1997a; 1997b). After all ants have constructed their solutions, the best solution will be selected based on the shortest tour or makespan and utilization criteria. The best solution will be stored 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 node (or the edge) for the next iteration. The heuristic function will start immediately after the global update in order to update the heuristic value.

EXPERIMENTAL RESULT

Experiments were conducted to evaluate the performance of the enhanced ACS algorithm on two problem domain. The domains are the TSP and grid computing.

Several initializations will have to be performed before EHF_ACS algorithm can be applied to TSP. The initializations are: (i) calculate distance between cities using Euclidean distance method. (ii) calculate heuristic values using heuristic function (1/distance). (iii) initialize pheromone on all paths with values equal to 1 / (No of cities * nearest neighbor solution. (iv) set alpha (α), beta (β), delta (δ), q, p, number of ants (m), and number of iteration. The tour length is used to measure the quality of each ant where 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.

In the experiment for TSP, eight data sets from TSPLIP with different sizes were used. Results were compared to classical ACS algorithm) (Dorigo & Gambardella, 1997a; 1997b). Parameters initializations are as follows: $\alpha = 1$, $\beta = 2$, $\delta = 0.5$, $q = 0.9$, $m = 10$, $p = 0.1$, Initial pheromone (τ_0) = 1/ (N*nn), where N = number of cities and nn = nearest neighbour, and number of iteration = 10000.

Comparisons of results between the proposed algorithm and best known solution and ASC results from previous studies (Wei-Jie et al., 2009; Aljanaby et al., 2008; Wei-Jie & Jun, 2010) are depicted in Table 1. It can be seen that the proposed 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 proposed algorithm are better than ACS and for the best tour results the proposed algorithm is at par with ACS. The mean and SD shows the robustness of the proposed algorithm and its ability to guide the ants to quickly converge to the best solution. Each data set was run for five times to calculate the mean and SD. All the experiments using EHF_ACS produced good solution with minor differences between the runs.

Table 1. Performances of ACS and EHF_ACS Algorithm on TSP

TSP instance	Optimum	Source	ACS			EHF_ACS			|Mean ACS – Mean EHF_ACS|%
			Mean	SD	Best	Mean	SD	Best	
att48	33522	[21]	35595	a---	33780	**33614.4**	**43.135**	**33587**	5.56 %
eil51	426	[17]	428.21	2.05	426	**428**	**1.095**	**426**	0.05 %
st70	675	[22]	682.50	2.82	677	**677.2**	**0.748**	**676**	0.78 %
eil76	538	[17]	541.55	2.97	538	545.2	**1.469**	543	0.67 %
rat99	1211	[22]	1219.60	6.45	1211	**1212.6**	**0.8**	**1211**	0.57 %
kroA100	21282	[20]	21441.30	112.13	21315	**21297.2**	**11.51**	**21282**	0.67 %
eil101	629	[17]	640.67	5.86	630	**633**	**2.449**	631	1.20 %
rat195	2323	[22]	2352.76	15.79	2334	**2347**	**4.147**	2342	0.24 %

a. SD is not calculated in the original study.

In order to apply EHF_ACS to job scheduling problem, several initializations will have to be performed on i) list of jobs and resources, ii) array for the pheromone and iii) the variables alpha (α), beta (β), delta (δ), p, q, number of ants and number of iteration.

A simulator was developed to test the performance of the proposed algorithm for the grid computing domain. Jobs were created with size varies from 500 – 1000 Million Instruction (MI) and the number of jobs is set between 10 and 100. The number of resources created for these experiments are seven resources with capacity varying from 50 – 250 Million Instruction per Second (MIPS) with load from 0 – 9.

In total, the number of experiments conducted is 100 for ACS and 100 for EHF_ACS. 10 instances created based on incremental number of jobs from 10-100 with interval of 10 jobs. Each 10 experiments conducted using the same instances to get the average and SD of each 10 experiments' results. Other parameters settings are as follows: number of ants = 7, $\alpha = 1$, $\beta = 2$, initial pheromone (t_0) = 0.0001, p = 0.5, q = 0.9, number of iteration = 1000 and δ = 0.5.

The performance metrics used to evaluate the proposed algorithm were makespan and utilization. Shorter makespan indicates faster performance in term of processing time while utilization criteria indicate the quality of jobs scheduling and load balancing policy. Contradiction between makespan and utilization may occur if there is limited resource. Therefore, load balancing provide fair distribution rather than equal distribution in order to get good performance. Table 2 presents the comparison between ACS and EHF_ACS in terms on makespan and utilization where better results are highlighted. It can be seen that for makespan, the proposed algorithm performed better than ACS algorithm in 8 out of 10 cases. As for utilization, the proposed algorithm performed better than ACS algorithm in 7 out of 10 cases.

Table 2. ACS and EHF_ACS performances on Grid Computing

	ACS				EHF_ACS			
Task	**Makespan**	**Utilization**	**Avg UT**	**SD**	**Makespan**	**Utilization**	**Avg UT**	**SD**
10	**12.52**	64.56	62.48	1.04	15.929	**77.715**	74.82	3.82
20	22.638	74.716	71.59	3.3	**20.541**	**90.14**	85.74	5.63
30	33.364	**92.396**	83.94	5.24	**29.395**	88.246	82.94	3.86
40	**35.566**	**88.658**	81.57	5.46	43.547	85.604	81.41	2.81
50	51.913	88.984	77.1	5.7	**42.575**	**92.96**	87.85	3.14
60	65.359	88.896	82.81	5.1	**50.327**	**94.369**	87.5	4.62
70	64.82	77.747	74.02	7.26	**59.836**	**89.643**	84.14	3.61
80	71.647	**96.154**	79.75	8.79	**65.747**	96.14	87.39	5.32
90	101.043	80.417	73.95	3.8	**73.134**	**94.106**	87.56	4.48
100	110.039	82.26	74.68	3.75	**80.284**	**95.472**	86.62	4.99

The above results show that the proposed new heuristic function hat has been integrated in ACS produces better results as indicated by smaller values for standard deviation in most cases.

CONCLUSION

The new heuristic function can produce a heuristic value that reflects the quality of the best-so-far solution. This is important because ants can be guided to choose a path that will prevent stagnation. The new heuristic function is integrated into the classical ACS algorithm thus enhancing it. The enhanced ant colony systems algorithm can be considered as a new member to the family of ant colony optimization algorithms. Results show that the proposed algorithm outperforms the original ant colony system algorithm. Future work can focus on improvement in the data structure where better solution can be obtained in a shorter time.

ACKNOWLEDGMENTS

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.

REFERENCES

Aljanaby, A., Ku-Mahamud, K.R., and Norwawi, N.M. (2008). Optimizing Large Scale Combinatorial Problems Using multiple Ant Colonies Algorithm Based on Pheromone Evaluation Technique. *International Journal of Computer Science and Network Security IJCSNS*, vol. 8(10), pp. 54 – 58.

Bullnheimer, B., Hartl, R.F., and Strauss, C. (1999). 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.

Colorni, A., Dorigo, M., and Maniezzo,V. (1991). Distributed Optimization by Ant Colonies. *Proceedings of the first European Conference on Artificial Life, Cambridge*, pp. 134 – 142.

Dorigo, M. and Stutzle, T. (2004). *Ant colony optimization*. Cambridge, Massachusetts, London, England: MIT Press.

Dorigo, M., and Gambardella, L.M. (1997a). Ant Colonies for the Travelling Salesman Problem. *BioSystems Journal*, Vol. 43(2), pp. 73–81.

Dorigo, M. (1992). *Optimization, learning and natural algorithms* (Unpublished doctoral dissertation), Politecnico di Milano, Italy.

Dorigo, M., and Gambardella, L.M. (1997b). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation,* Vol. 1(1), pp. 53 – 66.

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

Stutzle, T., and Hoos, H. (1997). MAX-MIN ant system and local search for the traveling salesman problem, *Proceedings of IEEE International Conference on Evolutionary Computation. Indianapolis*, pp. 309 – 314.

Wei-Jie, Y., Xiao-min, H., Jun, Z., and Rui-Zhang, H. (2009). 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.

Wei-Jie, Y., and Jun, Z. (2010). Pheromone-distribution-based adaptive ant colony system. *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, New York, USA,* pp. 31-38.