Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Self-Adaptive Ant Colony System for the Traveling Salesman Problem

Wei-jie Yu, Xiao-min Hu and Jun Zhang
Department of Computer Science
SUN Yat-sen University
Guangzhou, P.R.China, 510275
junzhang@ieee.org

Rui-Zhang Huang
Department of Industrial and System Engineering
The Hong Kong Polytechnic University
Kowloon, Hong Kong
mfrzh@polyu.edu.hk

***Abstract*—In the ant colony system (ACS) algorithm, ants build tours mainly depending on the pheromone information on edges. The parameter settings of pheromone updating in ACS have direct effect on the performance of the algorithm. However, it is a difficult task to choose the proper pheromone decay parameters α and ρ for ACS. This paper presents a novel version of ACS algorithm for obtaining self-adaptive parameters control in pheromone updating rules. The proposed adaptive ACS (AACS) algorithm employs *Average Tour Similarity* (*ATS*) as an indicator of the optimization state in the ACS. Instead of using fixed values of α and ρ, the values of α and ρ are adaptively adjusted according to the normalized value of *ATS*. The AACS algorithm has been applied to optimize several benchmark TSP instances. The solution quality and the convergence rate are favorably compared with the ACS using fixed values of α and ρ. Experimental results confirm that our proposed method is effective and outperforms the conventional ACS.**

***Keywords*—Ant colony system (ACS), adaptive parameters control, traveling salesman problem**

I. INTRODUCTION

Ant colony optimization (ACO) [1] is a new metaheuristic inspired by the foraging behavior of real ants. These ants find the shortest path from their nest to a food source by using an indirect form of communication called stigmergy. When the ants are walking, they lay a certain amount of pheromone on the ground and decide a direction to move by detecting the pheromone previously deposited by other ants. Supposing that all ants walk at approximately the same speed, more ants will visit the shorter paths than the longer paths. Therefore, a greater amount of pheromone is accumulated on the shorter paths and thus attracts more new ants to follow. Based on this positive feedback effect, all ants will eventually choose the shortest path.

ACO algorithms have been successfully applied to many applications, such as the traveling salesman problem (TSP) [2][3], workflow scheduling problem [4], job scheduling problem (JSP) [5], protein folding problem [6], continuous optimization problem [7]-[9], and other industrial problems [10]-[12]. The traveling salesman problem (TSP) is one of the most studied NP-hard problems of combinatorial optimization. An algorithm called ant system (AS) inspired by the ants' behavior was first applied to TSP by Dorigo and his colleagues [2]. However, the AS is not suitable for solving large TSP problems. Thus, the successor of AS—ant colony system (ACS) [3] was developed for achieving improvements on the performance.

How to properly update the pheromone level is the key of the ACS algorithm. In the ACS algorithm, pheromone updating rules include global updating and local updating. The values of α and ρ, which are the two corresponding parameters of the pheromone updating rules, have significant effect on the performance of the ACS. Many studies have been done in the literature for analyzing the influence of proper parameter values [3][13][14] and attempting to find the best parameter values of ant-based algorithms [15]-[17]. As described in [18], fine-tuning the best parameter values for solving an optimization problem is time-consuming. Moreover, the best parameter values are problem-dependent, which means that the best parameter values for an instance may not be as good for the other instances. Various methods have been proposed for searching the best ACO parameter values, such as computational intelligence algorithms (e.g., genetic algorithms [15][16] and particle swarm optimization [17]) and experimental methods in [18] and [19]. However, in these methods, the evaluation of each combination of parameter values needs to run the ACO algorithm several times. The computation for finding the best parameter values by these methods is thus quite heavy.

In this paper, we propose an adaptive ACS (AACS) algorithm, which is characterized by an online parameter tuning mechanism for adapting the optimization situation. Similar to the work in [20], the parameter values in AACS are not fixed but dynamically changed during the search process. We use the *Average Tour Similarity* (*ATS*), which measures the differences among the ants' tours, as the indicator of the optimization state in the ACS. In the initial state, the value of the global pheromone decay parameter α is maintained to be large and the value of the local pheromone decay parameter ρ is small. On the contrary, when the mature state is detected, the value of α is reduced whereas the value of ρ is increased.

The rest of this paper is organized as follows. Section II presents the structure of the ACS algorithm for solving TSPs. Section III describes the adaptive version of the ACS algorithm in detail. Section IV shows the experimental results of AACS

compared with the conventional ACS. Conclusions are made in Section V.

II. ACS ALGORITHMS FOR THE TSP

A. Framework of the ACS Algorithm

The TSP can be represented by a graph $G = (N, E)$, where N is the set of cities and E is the set of edges between cities. It is a problem of finding a shortest closed tour that visits every city once. The ACS algorithm for a TSP is described as follows [3]:

Step 1) Initialization: The parameters in the ACS are initialized and all ants are randomly placed on cities as the beginning of the tours.

Step 2) Building Tour: Each ant builds a tour by repeatedly applying a *pseudo-random-proportional* rule. When located at city r, ant k chooses the next city s to move to according to the rule given by

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

where $\tau(r,u)$ is the pheromone level between city r and city u, $\eta(r,u)$ is the inverse of the distance between city r and city u, $J_k(r)$ is the set of unvisited cities of ant k, β ($\beta > 0$) is the heuristic factor which determines the relative importance of pheromone versus distance, q is a random number uniformly distributed in [0,1], q_0 ($0 \le q_0 \le 1$) is a parameter, and S is a random variable selected according to the probability distribution given by

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

The above procedure is repeated until every ant has visited all of the cities. Then all ants move back to the initial cities.

Step 3) Local Pheromone Updating: During the construction of a tour, ants use a local updating rule to change the pheromone level of the edge (r, s) that they have passed as

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

where ρ ($0 < \rho < 1$) is the local pheromone decay parameter, τ_0 is the initial pheromone level.

Step 4) Global Pheromone Updating: The global pheromone updating occurs once all ants have completed their tours. Only the edges (r, s) belonging to the best tour have the pheromone level updated as

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

where α ($0 < \alpha < 1$) is the global pheromone decay parameter. The value of $\Delta\tau(r,s)$ is determined by (5)

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

where L_{ib} is the length of the best tour found in this iteration.

The iteration is then repeated from step 2) again until the algorithm meets an ending condition.

B. Discussion on the Pheromone Decay Parameters

Cooperation among ants plays a key role in the ACS algorithm. Real ants coordinate their behaviors via stigmergy, a form of indirect communication mediated by modifying the environment [1]. In the ACS algorithm, modifications of the environment are achieved by the pheromone updating.

Based on the description of the ACS algorithm, the pheromone decay parameters α and ρ have direct influence on the effect of pheromone in guiding ants' solution construction, moreover, on the performance of the algorithm.

The global pheromone updating in ACS is intended to reinforce the shortest tour by depositing a great amount of pheromone on the corresponding edges. α is the decay parameter for the global pheromone updating rule. If the value of α is larger, a greater amount of new pheromone will be added on the best tour in each iteration. Although the search will be more direct with a larger α, the optimization is easily trapped into local optima. On the contrary, if the value of α is small, the global searching ability of the algorithm is improved, but the convergence rate is reduced.

In contrast, the effect of the local pheromone updating is to make the edges less desirable for the other ants. ρ is the decay parameter for the local pheromone updating rule. If the value of ρ is large, the tour just visited by ants will be weaken by the high pheromone evaporation. Therefore, the search randomness is improved, but the search is less direct. Conversely, if the value of ρ is small, the algorithm converges faster, but it may easily lead to premature convergence.

In the conventional ACS, the values of α and ρ are fixed throughout the optimization process. There are two drawbacks. First, the best values of α and ρ are instance-dependent. The best setting of α and ρ for solving an instance may not be as good when applied to another instances. Second, the importance of reinforcing the pheromone on a high quality tour or decreasing the desirability of an edge is different during the optimization process.

According to the above analysis, if the parameter values are not set properly, the optimization may stagnate from achieving the optimum. Using proper values of α and ρ in different optimization states not only helps find high quality tours but also accelerates the search speed. Hence, an adaptive version of the ACS is proposed in the next section.

III. ADAPTIVELY ADJUSTING α AND ρ

A. Strategies for Adjusting the Values of α and ρ

Instead of using fixed values of α and ρ, we propose a novel adaptive method to adjust the values of α and ρ according to the optimization state. Equations (6) and (7) present the adjustment of the values of α and ρ in a general form.

$$\alpha = F_{\alpha}(\Phi), \tag{6}$$

$$\rho = F_{\rho}(\Phi), \tag{7}$$

where Φ is an indicator of the optimization state in the ACS algorithm. Based on the state Φ, $F_{\alpha}()$ and $F_{\rho}()$ make suitable strategies for adjusting the values of α and ρ.

In the initial state, a great amount of pheromone is deposited on the shorter tours for attracting more ants to follow. Thus, the value of α should be large and the value of ρ should be small. On the contrary, the shorter tour should be weakened by pheromone evaporation to avoid being trapped in local optima in the mature state. Thus, the value of α should be small and the value of ρ should be large. Fig. 1 and Fig. 2 illustrate the strategies for adjusting the values of α and ρ in different optimization states, respectively.

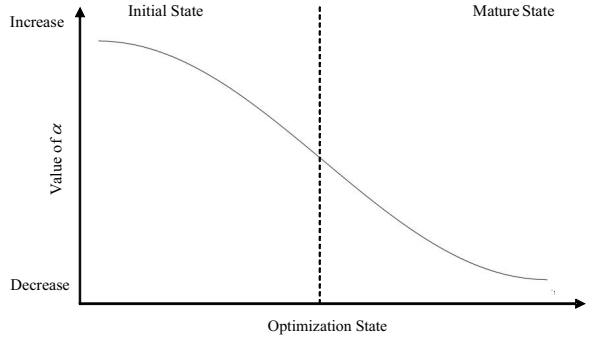


Figure 1. Adjusting the value of α in different optimization states

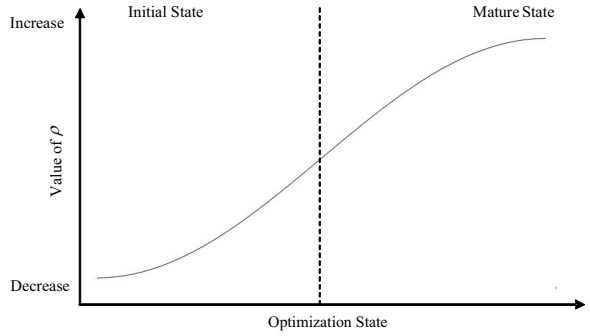


Figure 2. Adjusting the value of ρ in different optimization states

B. Computation of the Average Tour Similarity

As shown in Fig. 3, the flowchart of the proposed adaptive ACS (AACS) algorithm, the adaptive adjustment of α and ρ occurs after all ants have completed their tours in each iteration. In the AACS, we use the *Average Tour Similarity* (*ATS*) to measure the optimization state.

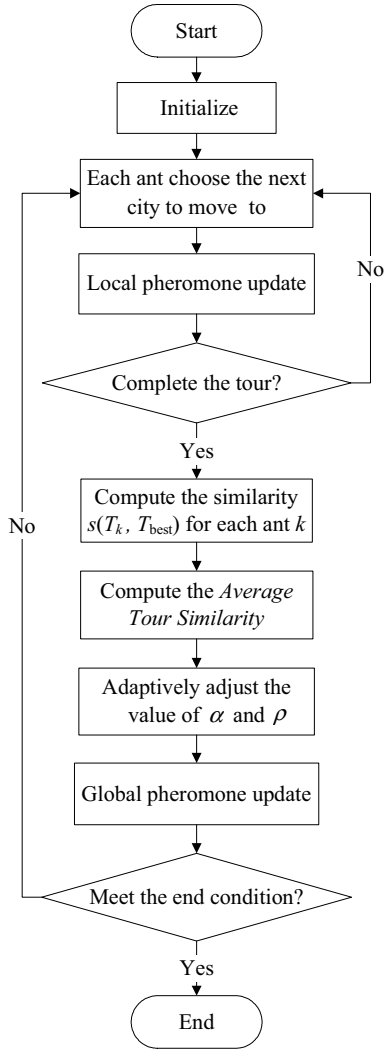


Figure 3. The flowchart of the Adaptive ACS (AACS) algorithm

Step 1) Compute the similarity $s(T_k, T_{\text{best}})$ between the tour T_k of every ant k and the best tour T_{best} in the current iteration, $k = 1,2,...,m$. m is the number of ants. We define $s(T_k, T_{\text{best}})$ as the number of edges contained in T_k as well as in T_{best}.

$$s(T_k, T_{\text{best}}) = | T_k \cap T_{\text{best}} |, \tag{8}$$

Because the number of edges contained in a tour equals to the number of cities, we have $s(T_k, T_{best}) \in \{0,1,...,n\}$.

Step 2) Compute the *Average Tour Similarity* (*ATS*) according to the equation given by

$$ATS = \frac{1}{m} \cdot \sum_{k=1}^{m} s(T_k, T_{\text{best}}). \quad (9)$$

We also have $ATS \in \{0,1,...,n\}$. The normalized value of *ATS* is denoted as

$$\overline{ATS} = ATS / n, \quad (10)$$

where $\overline{ATS}$ ranges from zero to one.

Fig. 4 shows the development of the normalized value of the *ATS* throughout the optimization process on the benchmark TSP instance eil76. The Y-axis represents the value of $\overline{ATS}$ and the X-axis represents the number of iterations. According to the figure, the value of $\overline{ATS}$ is relatively small in the initial state. Then the value of $\overline{ATS}$ becomes larger, showing the transition from the initial state to the mature state.

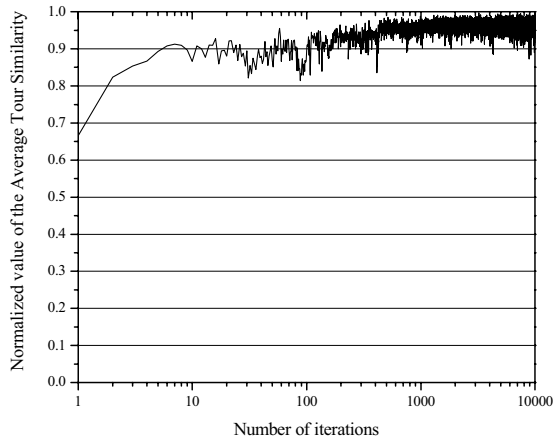


Figure 4. Development of the normalized value of the *Average Tour Similarity* (*ATS*) throughout the optimization process on the TSP instance eil76

The figure indicates that ants built tours that have more and more edges in common with the best tour during the optimization process. This is achieved by applying the *pseudo-random-proportional* transition rule and the global pheromone updating rule in ACS. Thus, the probability with which ants choose the same shorter tours increases when the optimization process transfers from the initial state to the mature state, which leads to providing a greater amount of pheromone with the best tour that can attract more and more ants to visit.

C. Realization of Adaptively Adjusting Values of α and ρ

The values of α and ρ are adjusted adaptively according to the value of $\overline{ATS}$, which is the indicator of the optimization state. The realization of (6) and (7) for the adaptive adjustment of the values of α and ρ is implemented as

$$\alpha = a_\alpha \cdot \overline{ATS} + b_\alpha, \quad (11)$$

$$\rho = a_\rho \cdot \overline{ATS} + b_\rho. \quad (12)$$

In the initial state, i.e., the value of $\overline{ATS}$ is small, we need a relatively large value of α and a relatively small value of ρ to accelerate the convergence speed. When the value of $\overline{ATS}$ increases to a high value, indicating the optimization enters the mature state, a relatively small value of α and a relatively large value of ρ are needed to weaken the attraction of short edges.

According to the correlation between $\overline{ATS}$ and parameters α and ρ, the value of a_α is negative while the value of a_ρ is positive. The constants b_α and b_ρ are used for keeping the adjusted values of α and ρ in a proper interval.

IV. Experimental Comparisons

In order to demonstrate the efficiency of the Adaptive ACS (AACS) proposed in this paper, we compare the conventional ACS [3] with the AACS on three symmetric TSP instances, including eil51, eil76 and eil101 from the TSPLIB. Table I tabulates the size and the optimal tour length of each instance. The objective is to find the optimal tour length fast and accurately.

TABLE I. Benchmark TSP Instances

Instance	*Number of cities*	*Optimum tour length*
eil51	51	426
eil76	76	538
eil101	101	629

A. Parameters and Settings

In both the conventional ACS and the proposed AACS algorithms, the following parameters are set the same for all of the instances: $m = 10$, $\beta = 3.5$, $q_0 = 0.9$, and $\tau_0 = \frac{1}{n \cdot L_{nn}}$, where L_{nn} is the tour length computed by nearest neighbor heuristic and n is the number of cities. The conventional ACS uses a fixed value 0.1 for α and ρ according to [3]. In the proposed AACS algorithm, the values of α and ρ are adjusted adaptively. Each instance is tested 100 times independently. The maximum number of iterations is defined as 5000.

B. Results and Analysis

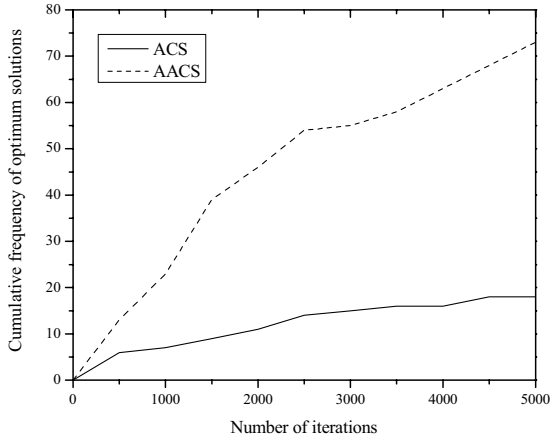
Table II summarizes the results of the conventional ACS and AACS on geometric instances of the symmetric TSP. The average tour length, the best solution, the worst solution, and the standard deviation generated by each algorithm are listed. The "*Opt-times*" column represents the number of times the optimum solutions found.

TABLE II. COMPARISON BETWEEN THE CONVENTIONAL ACS AND THE PROPOSED AACS

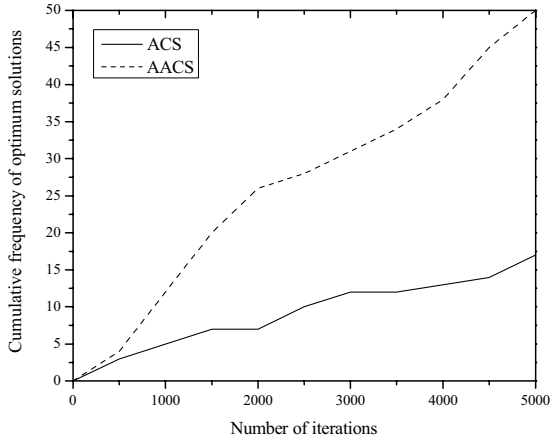
Instance	*Algorithm*	*Average*	*Best*	*Worst*	*Std.dev.*	*Opt-times*
eil51	ACS	428.21	426	435	2.05	18
	AACS	426.51	426	430	0.89	73
eil76	ACS	541.55	538	550	2.97	17
	AACS	538.91	538	543	1.17	50
eil101	ACS	640.67	630	655	5.86	0
	AACS	635.85	629	647	4.75	8

It can be seen from Table II that AACS performs better than ACS. As smaller average tour lengths and much larger values of opt-times than those of ACS are obtained by AACS, AACS has a stronger ability to find better solutions than ACS. In addition, AACS is more stable than ACS because the standard deviation of the tour lengths (denoted as "*std.dev.*") generated by AACS is smaller than ACS.

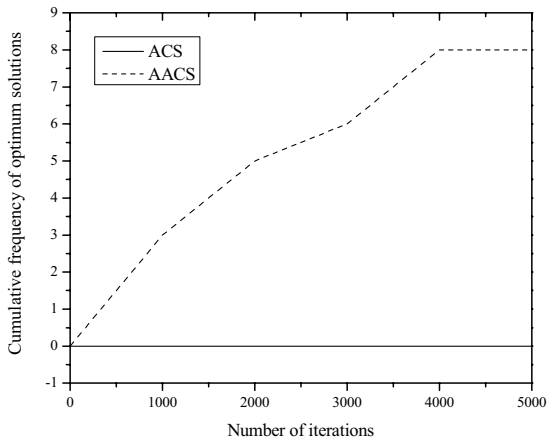
Fig. 5 compares the cumulative frequency of obtaining the optimal solutions in every 500 iterations of ACS with that of AACS. One can observe that AACS searches the optimum solutions much faster than ACS on all the three TSP instances.



(a)



(b)



(c)

Figure 5. Comparison of ACS and AACS for cumulative frequency of optimum solutions obtained in every 500 iterations. (a) TSP instance eil51. (b) TSP instance eil76. (c) TSP instance eil101.

V. CONCLUSION

In this paper, a self-adaptive method to adjust values of the pheromone decay parameters α and ρ for ACS has been proposed. The values of α and ρ are adapted to the *Average Tour Similarity* (*ATS*), which is an indicator of the optimization state.

The adaptive ACS (AACS) algorithm has been implemented and tested on benchmark TSP instances. The experimental results show that the proposed AACS algorithm gives better performance than the one with fixed pheromone decay parameters settings in terms of both the solution quality and the searching speed.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation of China under Project 60573066, in part by the NSF of Guangdong under Project 5003346, in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, China, in part by the National Natural Science Foundation of China Joint Fund with Guangdong under Key Project U0835002, in part by the National High-Technology Research and Development Program of China, No. 2009AA01Z208. Jun Zhang is the corresponding author, e-mail: junzhang@ieee.org.

REFERENCES

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

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," IEEE Trans. Systems, Man, and Cybernetics-Part B, vol. 26, no. 1, pp. 29-41, February 1996.

[3] M. Dorigo and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Trans. Evolutionary Computation, vol. 1, no. 1, pp. 53-66, April 1997.

[4] W.-N. Chen and J. Zhang, "An ant colony optimization approach to a Grid workflow scheduling problem with various QoS requirements,"

IEEE Trans. Systems, Man, and Cybernetics-Part C, vol. 39, no. 1, pp. 29-43, 2009.

[5] J. Zhang, X.-M. Hu, X. Tan, J.-H. Zhong, and Q. Huang, "Implementation of an ant colony optimization technique for job scheduling problem," Transactions of the Institute of Measurement and Control, vol. 28, no. 1, pp. 93-108, 2006.

[6] X.-M. Hu, J. Zhang, J. Xiao, and Y. Li, "Protein folding in hydrophobic-polar lattice model: a flexible ant-colony optimization approach," Protein & Peptide Letters, vol. 15, no. 5, pp. 469-477, 2008.

[7] X.-M. Hu, J. Zhang, and Y. Li, "Orthogonal method ant colony search for solving continuous optimization problems," Journal of computer science and technology, vol. 23, pp. 2-18, January 2008.

[8] Y. Lin, J. Zhang, and J. Xiao, "A pseudo parallel ant algorithm with an adaptive migration controller," Applied Mathematics and Computation, vol. 205, pp. 677-687, 2008.

[9] J. Zhang, W.-N. Chen, J.-H. Zhong, Xuan Tan, and Yun Li, "Continuous function optimization using hybrid ant colony approach with orthogonal design scheme," SEAL'06, LNCS 4247, pp. 126-133, 2006.

[10] X.-M. Hu, J. Zhang, and H. Chung, "An intelligent testing system embeded with an ant colony optimization based test composition method," IEEE Trans. Systems, man, and Cybernetics-Part C. (in Press)

[11] J. Zhang, H. Chung, W. L. Lo, and T. Huang, "Extended ant colony optimization algorithm for power electronic circuit design," IEEE Trans. Power Electronics, vol. 24, no. 1, pp. 147-162, January 2008.

[12] X.-M. Hu, J. Zhang, and L.-M. Zhang, "Swarm intelligence inspried multicast routing: an ant colony optimization approach," Evo Workshops 2009, LNCS 5484, pp. 51-60, 2009.

[13] A. C. Zecchin, A. R. Simpson, H. R. Maier, and J. B. Nixon, "Parametric study for an ant algorithm applied to water distribution system optimization," IEEE Trans. Evolutionary Computation, vol. 9, no. 2, pp. 175-191, April 2005.

[14] X.-Y. Luo, F. Yu, and J. Zhang, "Study of parametric relation in ant colony optimization approach to traveling salesman problem," LNBI 4115, ICIC 2006, pp. 22-32, 2006.

[15] M. L. Pilat and T. White, "Using genetic algorithms to optimize ACS-TSP," ANTS 2002, LNCS 2463, pp. 282-287, 2002.

[16] H. M. Botee and E. Bonabeau, "Evolving ant colony optimization," Complex Systems, vol. 1, no. 2, pp. 149-159, 1998.

[17] D. Gómez-Cabrero, C. Mrmero, and D. N. Ranasinghe, "The travelling salesman's problem: a self-adapting PSO-ACS algorithm," In Proceedings of the Second Conference on Industrial and Information Systems, ICIIS 2007, pp. 479-484, 2007.

[18] B. Adenso-Díaz and M. Laguna, "Fine-tuning of algorithms using fractional experimental designs and local search," Operations Research, vol. 54, no. 1, pp. 99-114, 2006.

[19] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 11-18, 2002.

[20] J. Zhang, H. Chung, and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," IEEE Trans. Evolutionary Computation, vol. 11, no. 3, pp. 326-335, June 2007.