A Memory-based Ant Colony Algorithm for the Bipartite Subgraph Problem

Rong-Long Wang and Li-Qing Zhao

Graduate School of Engineering, University of Fukui, Bunkyo 3-9-1, Fukui-Shi, Japan 910-8507

Summary
An innovative ant colony algorithm called memory-based ant colony algorithm is proposed to solve the bipartite subgraph problem. In the proposed algorithm, artificial ant has memory of solution found previously, and can use it to construct a new solution. Besides, in the proposed algorithm two kinds of pheromone and two kinds of heuristic information are also adopted to reinforce the search ability. The proposed algorithm is tested on a large number of instances and compared with other algorithms. The experimental results show that the proposed algorithm is superior to its competitors.

Key words:
Bipartite subgraph problem, ant colony optimization, NP-complete problem, combinatorial optimization problems

1. Introduction

The bipartite subgraph problem [1] is an important problem from combinatorial optimization. It has many important applications in modeling matching problem, modern coding theory, and communication network, printed circuit board, and computer science [2]. It was proved to be a NP-complete problem [3] [4]. It is well known that there is no tractable algorithm to solve NP-complete problem, which motivates to find better algorithms that yield better approximate solutions. Many approximate algorithms have been proposed for bipartite subgraph problem. An algorithm for solving the largest bipartite subgraphs in triangle-free graph with maximum degree three has been proposed [5]. Grotschel and Pulleyblank [6] defined a class of weakly bipartite graphs such that the convex hull of incidence vectors of bipartite subgraphs is defined by odd cycle inequalities. Barahona [7] characterized another class of weakly bipartite graphs. Based on the Hopfield neural network [8], Lee et al. [9] proposed a maximum neural algorithm for the bipartite subgraph problem. Unfortunately, the maximum neural network that is based on the steepest descent method sometimes causes the excessive fixation of the state of the neural network and easily converges to the oscillation state of local minimum because of the limitation of its ability [10]. Wang et al. [11] proposed a parallel algorithm using gradient ascent learning algorithm of the Hopfield network, the result of which is superior to that of Lee et al. Global search methods such as simulated annealing can also be applied to the problem, but they are generally very slow [12] [13].

Currently, in the optimization field, swarm intelligence algorithms have been shown to be good problem solvers with various application domains. These mainly consist of the particle swarm optimization (PSO) and the ant colony optimization (ACO) [14]. Especially ACO algorithms have been shown to be very successful for solving problems from combinatorial optimization. The inspiring source of ACO algorithms are real ant colonies. More specifically, ACO is inspired by the ants' foraging behavior. At the core of this behavior is the indirect communication between the ants by means of chemical pheromone trails, which enables them to find short paths between their nest and food sources. This characteristic of real ant colonies is exploited in ACO algorithms in order to solve, for example, discrete optimization problems. In the original ACO algorithms, only one kind of pheromone is used. However for some combinatorial optimization problems, it is not efficient to show the quality of solution using only one kind of pheromone. We have proposed a two-state ant colony algorithm for the minimum graph bisection problem [15], in which two kinds of pheromone and two kinds of heuristic information are adopted to reinforce the search ability.

In this paper, we proposed a memory-based ant colony algorithm for solving the bipartite subgraph problem. In the proposed algorithm, artificial ant has memory of solution found previously, and can use it to construct a new solution. Besides, the idea of using two kinds of pheromone and two kinds of heuristic information proposed in our previous work [15] is also adopted in the proposed algorithm. The proposed algorithm is tested on a large number of instances and compared with other algorithms. The experimental results show that the proposed algorithm works remarkably well and is superior to its competitors.

Manuscript received March 5, 2010
Manuscript revised March 20, 2010

2. Problem Formulation

For a given N-Vertex M-edge undirected graph $G=(V,E)$, if the vertex set V can be partitioned into two subsets V_1 and V_2, such that each edge of G is incident to vertex in V_1 and to a vertex in V_2, then the graph G is called bipartite graph, and be denoted by $G=(V_1,V_2,E)$. Given a graph $G=(V,E)$ with a vertex set V and an edge set E, the goal of bipartite subgraph problem is to find a bipartite subgraph with the maximum number of edges, such that

$V_1 \cup V_2 = V$, $V_1 \cap V_2$ In other words, the goal of bipartite subgraph problem is to remove the minimum number of edges from a given graph such that the remaining graph is a bipartite graph. This problem can be formulated as follows: For a given graph with N vertices and M edges, its vertex set can be represented using vector $=(x_1, x_2, \ldots, x_N)$, where x_i ($i = 1, 2, \ldots, N$) expresses #i vertex is partitioned into the subset V_1 or V_2. A value of 1 for the x_i implies that #i vertex is partitioned into subset V_1, and a value of 0 denotes that it is partitioned into subset V_2. Therefore, the bipartite subgraph problem can be mathematically transformed into the following optimization problem:

$$\text{Maximize ; } C=\sum_{ij} |x_i-x_j| \qquad (1)$$

Where d_{ij} is 1 if there is an edge from vertex #i to vertex #j, otherwise, 0.

3. Ant Colony Optimization

Ant Colony Optimization (ACO) [16] is one of the most recent techniques for solving combinatorial optimization problems, and has been unexpectedly successful in recent years. ACO is based on the indirect communication of a colony of simple agents, called artificial ants, mediated by artificial pheromone trails. It implements a randomized construction heuristic that makes probabilistic decisions as a function of artificial pheromone trails, which are determined by the pheromone intensity and the heuristic information based on the input data of the problem to be solved, if they are available. The first ACO algorithm, called Ant System (AS) [16] was proposed by Dorigo in 1992. Since then, the ACO algorithms attracted the attention of more researchers.

Firstly, AS was applied to the Traveling Salesman Problem (TSP). Given a set of n towns, the TSP can be stated as the problem of finding a minimal length closed tour that visits each town once. We call d_{ij} the length of the path between towns i and j, and Let $\tau_{ij}(t)$ called pheromone be the intensity of trail on edge (i, j) which connects i and j at time t. Each of m ants decides independently on the city to be visited next based on the intensity of pheromone trail τ_{ij} and a heuristic value η_{ij} , until the tour is completed. Each ant is placed on a random start city, and builds a solution going from city to city, until it has visited all of them. The probability by which an ant k in a city i chooses to go to a city j next is given by:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{k \in J^k}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}(t)]^{\beta}} & \text{if } j \in J^k \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

Where the heuristic value $\eta_{ij} = 1/d_{ij}$, the parameters α and β determine the relative influence of pheromone and heuristic, and J^k is the set of cities that remain to be visited by ant k positioned on city i. Once all ants have built a tour, ants perform following pheromone update rule:

$$\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \qquad (3)$$

Eq. (3) consists of two parts. The left part makes the pheromone on all edges decay. The speed of this decay is defined by ρ, the evaporation parameter. The right part increases the pheromone on all the edges visited by ants. The amount of pheromone an ant k deposits on an edge (i,j) is defined by $L^k(t)$, the length of the tour created by that ant at iteration t.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^k(t)} & \text{if edge } (i,j) \text{ is used by ant } k \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

In this way, the increase of pheromone for an edge depends on the number of ants that use this edge, and on the quality of the solutions found by those ants.

Even though the original AS algorithm achieved encouraging results for the TSP problem, it was found to be inferior to state-of-the-art algorithms for the TSP as well as for other problems. Therefore, several extensions and improvements of the original AS algorithm were introduced over the years. ACS [17] has been introduced to improve the performance of AS. It differs in three main aspects from ant system. First, in ACS ants choose the next city using the pseudo-random-proportional action choice rule: when located at city i, ant k moves with probability q_0 to city l for which $\tau_{il}(t)\cdot[\eta_{il}]^{\beta}$ is maximal. With probability $(1-q_0)$ an ant performs a biased exploration of edges according to Eq. 2. Second, in ACS only the global best ant is allowed to add pheromone. The most interesting contribution of ACS is the introduction of a local pheromone update in addition to the pheromone update performed at the end of the construction process (called *offline* pheromone update). The effect of the local updating rule is to make an already chosen edge less desirable for a following ant. The Max-Min AS [18] is a direct improvement over AS. The main modifications introduced by Max-Min AS with respect to AS are the following. First, to exploit the best solution found, after iteration only the best ant which can be either the *iteration-best* or the *best-so-far*, is allowed to add pheromone. Second, to avoid search stagnation, the allowed range of the pheromone trail strengths is limited to the interval [τ_{min}, τ_{max}]. Last, the pheromone trails are initialized to the upper trail limit, which causes a higher exploration at the start of the algorithm. Another improvement over AS is the *rank-based* version of Ant System (AS_{rank}) [19]. In AS_{rank}, always the global-best tour is used to update the pheromone trails. Additionally, a number of best ants of the current iteration are allowed to add pheromone. To this aim the ants are sorted by tour length, and the quantity of pheromone an ant may deposit is weighted according to the rank r of the ant. Only the (w-1) best ants of each iteration are allowed to deposit pheromone. The global best solution, which gives the strongest feedback, is given weight w. The r_{th} best ant of the current iteration contributes to pheromone updating with a weight given by max {0, w-r}.

In addition to the ACO algorithms outlined above, there are some variants when applying ACO to various problems. For a more comprehensive overview that covers the application of ant-based algorithms to combinatorial optimization problems we refer the interested reader to [20][21].

4. A Memory-based Ant Colony Algorithm for Solving the Bipartite Subgraph Problem

Artificial ants in ACO algorithms can be seen as probabilistic construction heuristics that generate solutions iteratively by taking into account accumulated past search experience: pheromone trails and heuristic information on the instance under solution. In the existing ACO algorithms, each ant starts with an empty solution and constructs a complete solution iteratively. It is a process that the solution is constructed gradually from empty solution. In other word, the solution is incomplete in the whole process of iteration.

In this section, we propose a memory-based ant colony algorithm for the bipartite subgraph problem. In the memory-based ant colony algorithm, artificial ant has memory of solution found previously, and can use it to construct a new solution. Firstly a randomly initial solution is given; then, the initial solution is adjusted gradually until the end condition is achieved. Thus, the solution is always complete in the whole process, and the heuristic information is update once the solution is updated. The principle of updating heuristic is that awards and penalties are executed by taking into account accumulated past search experience after a solution is updated. Besides, the idea of using two kinds of pheromone and two kinds of heuristic information proposed in our previous work [15] is also adopted in the proposed algorithm. The details of this algorithm are as follows.

In the proposed ACO algorithm, each ant has two states in which the ant deposits two kinds of pheromone ($\tau^1$and τ^0) on vertex. Pheromone τ^1 and τ^0 indicate the learned desirability of partitioning the vertex into subset V_1 and V_2 respectively. Besides, two kinds of heuristic information η^1 and η^0 are also associated to a vertex to indicate the heuristic desirability of partitioning the vertex into subset V_1 and V_2 respectively. In the proposed algorithm, ant selects a vertex with high associated pheromone trail τ^1 and heuristic value η^1 for subset V_1, and then selects a vertex for subset V_2 using τ^0, η^0 in each iteration. In iteration t an ant k selects *#i* vertex for subset V_1 and V_2 with probabilities $p_k{}^1(i)$, $p_k{}^0(i)$ respectively.

$$P_k^1(i) = \begin{cases} \dfrac{[\tau_i{}^1(t)]^{r_1}[\eta_i{}^1(t)]^{r_2}}{\sum_{j\in J^k}[\tau_j{}^1(t)]^{r_1}[\eta_j{}^1(t)]^{r_2}} & \text{if } i \in J^k \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

$$P_k^0(i) = \begin{cases} \dfrac{[\tau_i{}^0(t)]^{r_1}[\eta_i{}^0(t)]^{r_2}}{\sum_{j\in J^k}[\tau_j{}^0(t)]^{r_1}[\eta_j{}^0(t)]^{r_2}} & \text{if } i \in J^k \\ 0 & \text{otherwise} \end{cases} \qquad (6)$$

Where $\tau_i{}^1$, $\tau_i{}^0$ are pheromone trail and $\eta_i{}^1$, $\eta_i{}^0$ are heuristic information in vertex *#i*. J^k is the set of vertexes that remain to be partitioned by ant k, the parameters r_1 and r_2 determine the relative influence of pheromone and heuristic information.

In the iteration, pheromone trails are updated. We use C (shown in Eq. (1)) to characterize the pheromone update process. The pheromone trails of the global-best ant and several other random ants in current iteration are captured with certain weight to update the pheromone. For example, if the number of random ants is N_r, the pheromone update is as follows:

$$\tau_i^1(t+1) = \rho\tau_i^1(t) + N_r\Delta\tau_i^{1,gb}(t) + \sum_{r=\text{random No.}}\Delta\tau_i^{1,r}(t) \qquad (7)$$

$$\tau_i^0(t+1) = \rho\tau_i^0(t) + N_r\Delta\tau_i^{0,gb}(t) + \sum_{r=\text{random No.}}\Delta\tau_i^{0,r}(t) \qquad (8)$$

where ρ is a coefficient such that (1 - ρ) represents the evaporation of trail between time t and t+1, $\Delta\tau_i^{1,gb}(t)$, $\Delta\tau_i^{0,gb}(t)$ represent the pheromone update of the global-best ant, and $\Delta\tau_i^{1,r}(t)$, $\Delta\tau_i^{0,r}(t)$ represent pheromone update of the random ants.

The amount of pheromone an ant k deposits on vertex *#i* is defined by:

$$\Delta\tau_i^{1,k}(t) = \begin{cases} \frac{C_k}{Edge\_Num - C_k} & \text{if vertex} \# i \epsilon V_1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\Delta\tau_i^{0,k}(t) = \begin{cases} \frac{C_k}{Edge\_Num - C_k} & \text{if vertex} \# i \epsilon V_2 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where C_k is defined in Eq. (1), *Edge_Num* is the number of edges in graph $G=(V,E)$.

The heuristic information can play a significant role in the performance of ACO algorithms. Original ACO algorithm uses static heuristic information, in which the heuristic information can be computed only once, when initializing the algorithm. Then it remains the same throughout the whole run of the algorithm. In the proposed method, we use dynamic heuristic information. The dynamic heuristic information depends on the partial solution constructed and it is computed at each construction step of each ant. Once a vertex is partitioned according to the probabilities (shown in Eq. 5 and Eq. 6), the heuristics information is updated by executing awards and penalties. Awards and penalties is the foundation in the proposed memory-based ant colony algorithm, and how to execute the awards and penalties is very important.

In the proposed method for the bipartite subgraph problem, the heuristic information is defined as followings:

$$\eta_i^1 = \frac{[EdgetoV_2]_i}{[EdgetoV_1]_i} \tag{11}$$

$$\eta_i^0 = \frac{[EdgetoV_1]_i}{[EdgetoV_2]_i} \tag{12}$$

where $[EdgetoV_1]_i$ is the edge number which the $\#i$ vertex connects to the vertex in group V_1, $[EdgetoV_2]_i$ is the edge number which the $\#i$ vertex connects to the vertex in group V_2. The method to execute the awards and penalties is as follows. For the $\#r$ vertex (d_{ir}=1) which has been partitioned just now, if the value of x_r is changed from 1 (in V_1) to 0 (in V_2) according the probabilities, then add α_1 to $[EdgetoV_2]_i$ as awards, and subtract α_2 from $[EdgetoV_1]_i$ as penalties, otherwise subtract α_2 from $[EdgetoV_2]_i$, add α_1 to $[EdgetoV_1]_i$. We call α_1 award parameter and α_2 penalty parameter. Thus, we have:

$$[EdgetoV_1]_i' = \begin{cases} [EdgetoV_1]_i + \alpha_1 & \text{if } x_r(0 \to 1) \\ [EdgetoV_1]_i - \alpha_2 & \text{if } x_r(1 \to 0) \end{cases} \tag{13}$$

$$[EdgetoV_2]_i' = \begin{cases} [EdgetoV_2]_i + \alpha_1 & \text{if } x_r(1 \to 0) \\ [EdgetoV_2]_i - \alpha_2 & \text{if } x_r(0 \to 1) \end{cases} \tag{14}$$

The awards and penalties are executed after each vertex is partitioned in the current iteration, and then update the heuristic information according to Eq. (11) and Eq. (12) for the next iteration.

5. Algorithm

The following search procedure describes the proposed memory-based ant colony algorithm for solving the bipartite subgraph problem:

(1) Set parameters.
(2) Randomly generate initial solutions and initialize the pheromone $\tau^1$and τ^0.
(3) Update the pheromone trails $\tau^1$and τ^0 according to Eq. 7 and Eq. 8.
(4) Update probabilities using Eq. 5 and Eq. 6, and partition a vertex to V_1 or V_2 according to the probabilities
(5) Awards and penalties are executed and heuristic information is updated for all vertexes according to Eq. 11 and Eq. 12.
(6) If all vertexes are partitioned, go to next step, else go to step (4).
(7) Calculate the solution quality using Eq. 1.
(8) If termination condition is satisfied, terminate this procedure. Otherwise, go to the step (3). The termination condition is as follow:

$$(Best\ solution - Average\ solu \tag{15}$$

6. Simulation Results

In order to verify the proposed approach, we have tested the algorithm with a large number of randomly generated graphs [22] defined in terms of two parameters, n and p. The parameter n specifies the number of vertices in the graph; the parameter p, $0 < p < 1$, specifies the probability that any given pair of vertices constitutes an edge. In preliminary experiments we tried to find reasonable parameter settings for the proposed memory-based ant colony algorithm. The award and penalty parameters α_1, α_2 are very important for the proposed algorithm. A set of different values was used for the algorithm to verify which group is better. From our preliminary simulations, we find that α_1=6.00, α_2=5.00 can result good results. The other parameters setting used in simulations is as follows: ants number is 100, the parameters (r_1, r_2) = (1, 6) which decide whether pheromone or heuristic is important, the evaporation parameters ρ=0.5. To evaluate our results, we compared our results with the other existing algorithm including Marks et al.'s heuristic algorithm [23], Lee et al.'s neural network algorithm [9] and Wang et al.'s Hopfield neural network learning algorithm [11].

Table 1: The comparisons of simulation results produced by different algorithms

Vertexes	*Probability*	*Edges*	Marks et.al [23]	Lee. et.al [9]	Wang et.al [11]	SA [22]	Proposed algorithm
50	0.05	61	53	52	53	53	53
50	0.15	183	136	133	136	136	136
50	0.25	305	205	203	205	205	205
80	0.05	158	134	127	134	134	134
80	0.15	474	330	325	330	330	330
80	0.25	790	513	504	513	513	513
100	0.05	247	207	196	206	207	207
100	0.15	742	501	492	501	502	502
100	0.25	1235	778	761	779	779	779
150	0.05	558	423	402	421	419	423
150	0.15	1676	1077	1062	1074	1069	1077
150	0.25	2790	1692	1645	1693	1674	1699
200	0.05	995	722	685	713	714	722
200	0.15	2985	1871	1838	1864	1586	1874
200	0.25	4975	2954	2886	2941	2948	2955
250	0.05	1556	1104	1060	1100	1094	1107
250	0.15	4668	2856	2809	2856	2849	2872
250	0.25	7780	4516	4435	4510	4526	4534
300	0.05	2242	1530	1486	1524	1522	1540
300	0.15	6727	4062	3987	4059	4061	4073
300	0.25	11212	6440	6393	6435	6441	6458

Information on the test graphs as well as all results is shown in table (1). From table (1) we can know that the proposed approach outperformed the other compared algorithms for the bipartite subgraph problem. Besides, because the simulated annealing (SA) is a well known search method and has good local search ability, we also compared the proposed algorithm with SA. For SA, we used the scheme proposed by Johnson et al [22], because the problem discussed by Johnson et al. is similar to the bipartite subgraph problem and very good solution was reported in their work. In the bipartite subgraph problem, the neighbors of a solution $\vec{x} = \{x_i \square \{0,1\}; i = 1, \ldots , N\}$ can be obtained from $\vec{x}$ by modifying the partition of a single vertex. Simulation results are also shown in table (1). From the table (1), we can know that the performance of the proposed memory-based ant colony algorithm is also better than SA.

7. Conclusions

We have proposed a memory-based ant colony algorithm for efficiently solving the bipartite subgraph problem. In the proposed algorithm, artificial ant has memory of solution found previously, and can use it to construct a new solution. The proposed algorithm is evaluated by performing a large number of simulations. The simulation results showed that the proposed algorithm works remarkably well and is superior to its competitors to solve the bipartite subgraph problem.

References

[1] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer, "Some simplified NP-complete graph problem," Theor, Comput. Sci., vol.1, pp.237-267, 1976

[2] S. Armen, et al., "Bipartite Graphs and their Applications," Cambridge Tracts in Mathematics, No. 131, pp.1, 1998.

[3] R. M. Karp, Reducibility among combinatorial problems, in Complexity of Computer Computations, Plenum: New York, pp85-104, 1972.

[4] S. Even and Y. Shiloach, "NP-completeness of several arrangement problems," Technical Report 43, Department of Computer Science, Technion, Haifa Istrael, 1975.

[5] J. A. Bondy and S. C. Locke, "Largest bipartite subgraph in triangle-free graphs with maximum degree three," J. Graph Theory, Vol.10, pp.477-504, 1986.

[6] M. Grotschel and W. R Pulleyblank, "Weakly bipartite graphs and the max-cut problem," Oper. Res. Lett., Vol.1, No.1, pp.23-27, 1981.

[7] F. Barahona, "On some weakly bipartite graph," Oper. Res. Lett., Vol.2, No.5, pp.239-242, 1983.

[8] J. J. Hopfield and D. W. Tank, " 'neural' computation of decisions in optimization problems," Biol. Cybern., Vol.52, pp.141-152, 1985.

[9] K. C. Lee, N. Funabiki, and Y. Takefuji, "A parallel improvement algorithm for the bipartite subgraph problem," IEEE Trans. Neural Network, vol.3, No.1 pp.139-145, 1992.

[10] Y. Takenaka, N. Funabiki and T. Higashino, "A proposal neural filter: A constraint resolution scheme of neural networks for combinatorial optimization problems," IEICE Trans. Fundamentals, Vol.E83-A, No.9, pp.1815-1823, 2000.

[11] R. L. Wang, Z. Tang, and Q. P. Cao, "A near-optimum parallel algorithm for bipartite subgraph problem using the

Hopfield neural network learning," IEICE Trans. Fundamentals, Vol.E85-A, No.2, pp.497-504, 2002.
[12] D. H. Ackley, G. E, Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzman machines," Cogn. Sci., Vol.9, pp.147-169, 1985.
[13] J. Hertz, A. Krogh, and R. G. Palmer, Introduction to the Theory of Neural Computation, Addision Wesley, 1991.
[14] R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization---An overview," Swarm Intell., Vol.1, pp33-57, 2007.
[15] R.-L. Wang and K. Okazaki, "A two-state ant colony algorithm for solving the minimum graph bisection problem", International Journal of Computational Intelligence and Applications, Vol.8, No.4, pp487-498, 2009.
[16] M. Dorigo, Optimization, "learning and natural algorithms," PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
[17] M. Dorigo, L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., Vol.1, pp.53-66, 1997.
[18] T. Stutzle and M. Dorigo, "A short convergence proof for a class of ACO algorithms," IEEE Trans. Evol. Comput. Vol.6, pp.358-365, 2002.
[19] B. Bullnheimer, R. F. Hartl and C. Strauss, "A New Rank Based Version of the Ant System: A Computational Study," Central Eur. J. Oper. Res. Econ., Vol.7, pp.25-38, 1999.
[20] C. Blum, "Ant colony optimization: Introduction and recent trends," Physics of Life Reviews, Vol.2, No.4, pp.353-373, 2005.
[21] N. Frank, S. Dirk, W. Carsten, "Analysis of different MMAS ACO algorithms on unimodal functions and plateaus," Swarm Intelligence, Vol.3, No.1, pp.35-68, 2009.
[22] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part 1, graph partitioning," Oper. Res., Vol.37, pp.865-892, 1989.
[23] J. Marks, W. Ruml, S.Shieber, and J.Ngo, "A seed-growthheuristic for graph bisection," Proc.Algorithm and Experiments, Trento, Italy, pp.76-87, 1998.

Rong-Long Wang received a B.S. degree from Hangzhe teacher's college, Zhejiang, China and an M.S. degree from Liaoning University, Liaoning, China in 1987 and 1990, respectively. He received his D.E. degree from Toyama University, Toyama, Japan in 2003. From 1990 to 1998, he was an Instructor in Benxi University, Liaoning, China. In 2003, he joined University of Fukui, Fukui Japan, where he is currently an associate professor in Department of Electrical and Electronics Engineering. His current research interests include genetic algorithm, neural networks, and optimization problems.

Li-Qing Zhao received the B.S. and M.S. degrees in Micro-electronics from Nan-Kai University in 2006 and 2009, respectively. During 2006 to 2009, she was focus on LCD research. From 2009 she is working toward her Ph.D degree at Fukui University, Japan. Her main research interests include ant colony optimization and combinatorial optimization problems.