Electronic Preprint for Journal of Information Processing Vol.26

Regular Paper

MAX-MIN Ant System with Memory Considering New Solution Importance

Takashi Isozaki[1,a)] Satoshi Hasegawa[2,b)] Hajime Anada[1,c)]

Received: June 23, 2017, Accepted: March 6, 2018

Abstract: Ant colony optimization (ACO) is a well-known swarm intelligence algorithm, with a population-based approach inspired by the foraging strategies of real ants. ACO has been applied to various combinatorial optimization problems belonging to non-deterministic polynomial-time hard (NP-hard) combinational problems. Of these, the traveling salesman problem (TSP) is one of the most important problems in the fields of technology and science. ACO algorithms provide promising results; however, they cannot compete with front-line algorithms when solving the TSP. In such situations, we consider that the Max-Min Ant System (MMAS) is fundamental and expansive in ACO algorithms. Therefore, we constructed a new algorithm by introducing three elements to the MMAS: individual ant memories initialized using the nearest neighbor method to memorize a good solution, New Solution Importance, and a local search procedure. Finally, we confirmed the effectiveness of the proposed algorithm by comparing it to other ACO algorithms using several benchmark problems.

Keywords: traveling salesman problem (TSP), ant colony optimization (ACO), Max-Min Ant System (MMAS)

1. Introduction

Ant colony optimization (ACO) [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] is a well-known metaheuristic swarm-intelligence-based algorithm, with a population-based approach inspired by the foraging strategies of real ants. ACO has been applied to various non-deterministic polynomial-time hard (NP-hard) combinational problems. Of these, the traveling salesman problem (TSP) is one of the most important problems in the fields of technology and science because it is widely applicable and academically interesting. Several researchers have studied the algorithms of this problem. ACO algorithms provide promising results; however, they cannot compete with front-line algorithms, because their convergence time is very long and the success rates of their optimal solutions are very low for practical purposes. Therefore, many groups of various fields have focused on improving ACO algorithms.

In such situations, we consider that the Max-Min Ant System (MMAS) [6] is fundamental and expansive in ACO algorithms. The MMAS's weakness is its long convergence time. There are two possible causes of this weakness: (1) although some ants find good paths, most of these ants cannot find good solutions owing to their subsequent inappropriate path selections and (2) although some ants find good solutions, no ant searches around these solutions owing to a lack of additional pheromones. These are common problems with ACO algorithms. On the other hand, Ant Colony Optimization with Memory (AS with Memory) [10] is an interesting algorithm and its application to the TSP should be considered. In this study, the memories of individual ants are introduced into the ant system (AS). These memories assume a role to include good paths among the paths newly found by an ant and the paths of the best solution found in the last iteration. These memories correct the first cause of the long convergence times mentioned above. As a result, including a memory in AS algorithm considerably improves AS.

Therefore, we constructed a new algorithm by introducing three elements to the MMAS: an individual ant memory initialized with the solution using the nearest neighbor method (NNM) to memorize a good solution, New Solution Importance, and a local search procedure. Finally, we confirmed the effectiveness of our algorithm via a comparison with two other ACO algorithms, MMAS and AS with Memory, using benchmark problems taken from the TSPLIB [11].

This study is organized as follows. In Section 2, we introduce three ACO algorithms. In Section 3, we explain the proposed algorithm. In Section 4, we show the experimental results obtained for the TSP. Finally, in Section 5, we briefly describe the discussion and conclusions of this study.

2. ACO

The basic procedure of ACO algorithms is as follows. They initialize the pheromone amounts on all the paths in Step (1) and iterate the three steps (Steps (2)–(4)) for a specified number of iterations:

(1) The pheromone amounts on all paths are initialized.

(2) All ants are placed in a randomly selected city.

[1] Systems Information Engineering, Graduate School of Engineering, Tokyo City University, Setagaya, Tokyo 158–8557, Japan
[2] Big Data Department Analyst Team, DMM.com Labo, Minato, Tokyo 106–6224, Japan
[a)] g1581802@gmail.com
[b)] android.acmos@gmail.com
[c)] h-anada@tcu.ac.jp

(3) Each ant creates its own solution according to the pheromone amounts and the heuristic information. The subsequent city j following city i at iteration t for ant k is determined by the probability $P_{ij}^k(t)$

$$P_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N'} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & \text{if } j \in N' \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\eta_{ij} = \frac{1}{d_{ij}}$$

where $\tau_{i,j}(t)$ is the pheromone amount of $path(i, j)$ at iteration t, α and β are the parameters that determine the relative importance of the pheromone amount and the heuristic information, respectively. N' is the set of cities that ant k has not yet visited and d_{ij} is the distance between city i and city j. By repeatedly using this probability, all ants find their own solution.

(4) The pheromone amounts are updated.

Multiple algorithms [7], [8], [9], [10] related to ACO have been proposed to improve previous solutions. We consider that the MMAS is fundamental and expansive in ACO algorithms. Moreover the AS with Memory is also an interesting ACO algorithm. While describing the following algorithms, we will explain some points relevant to the abovementioned ACO procedure.

2.1 AS

AS [1] was constructed in 1996 by Dorigo et al. as the first ACO algorithm. In Step (1) of the ACO procedure, pheromone amounts on all paths are initialized to a small constant value. In Step (4) of the ACO procedure, the pheromone amounts on each path are updated using the following equation:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \sum_{k=1}^{M} \Delta\tau_{ij}^k(t) \quad (2)$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L_k} & \text{if } path(i, j) \text{ belongs to } TOUR_k \\ 0 & \text{otherwise} \end{cases}$$

where ρ is the evaporation rate, M is the number of ants, $TOUR_k$ is the solution found by ant k, Q is a quantity of pheromone laid by an ant per tour, and L_k is the length of the $TOUR_k$. In this algorithm, all ants add pheromones according to their solutions.

2.2 MMAS

The MMAS [6] was constructed in 2000 by T. Stützle et al. This algorithm is different from AS only in Steps (1) and (4) of the ACO procedure. In Step (4), pheromone amounts on each path are updated using the following equation:

$$\tau_{ij}(t+1) = [\rho \tau_{ij}(t) + \Delta\tau_{ij}(t)]_{\tau_{\min}(t)}^{\tau_{\max}(t)} \quad (3)$$

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{1}{L_{IB}(t)} & \text{if } path(i, j) \text{ belongs to } TOUR_{IB} \\ 0 & \text{otherwise} \end{cases}$$

$$[x]_b^a = \begin{cases} a & \text{if } x > a \\ b & \text{if } x < b \\ x & \text{otherwise} \end{cases}$$

where $\tau_{\max}(t)$ is the upper bound, $\tau_{\min}(t)$ is the lower bound in iteration t, $TOUR_{IB}$ is the best solution in iteration t (the iteration-best solution) and $L_{IB}(t)$ is the length of $TOUR_{IB}$ in iteration t. $\tau_{\max}(t)$, $\tau_{\min}(t)$ are defined as follows:

$$\tau_{\max}(t) = \frac{1}{(1-\rho)L_{GB}(t)} \quad (4)$$

$$\tau_{\min}(t) = \frac{(1 - \sqrt[N]{0.05})\tau_{\max}(t)}{(N/2 - 1)\sqrt[N]{0.05}} \quad (5)$$

where N is the number of cities; and $L_{GB}(t)$ is the length of the best solution until iteration t (the global-best solution). In Step (1) of the ACO procedure, the initial pheromone amounts on all paths are $1/(\rho L_{NN})$, where L_{NN} is the tour length found via NNM. In this algorithm, only the ant that has the iteration-best solution in each iteration can add pheromones.

2.3 AS with Memory

AS with Memory [10] was constructed in 2012 by Wang et al. This algorithm is different from AS only in the use of memory, which all ants have in Step (3) of the ACO procedure in the following manner.

- Each ant's memory is initialized with the last iteration-best solution. The first city in the memory of each ant is the starting city for the ant. During the first iteration, each ant's memory is neither initialized nor used.
- Each ant selects its next city using Eq. (1). If this city is different from the next city in an ant's memory, then these cities in the memory are swapped. All ants create their own solution by iterating this procedure until the solution in their memory is shorter than the iteration best solution in the last iteration or they visit all the cities.

After all the ants create their solutions, the pheromone amounts on all the paths are updated in the same manner as updated in AS (cf. Eq. (2)) and all their memories are overwritten with the iteration-best solution. This algorithm assumes a role to include good paths among the paths newly found by an ant and the paths of the iteration-best solution. This algorithm delivers a superior performance compared to AS due to its shorter convergence time and better success rate of the optimal solution.

This algorithm is the AS introducing the memory. All solutions on the memories are not initialized at 0th iteration and but are overwritten with the iteration-best solution after each iteration.

3. MMAS with Memory Considering New Solution Importance

We constructed a new algorithm to solve the TSP by introducing the following three elements to the MMAS:

- Individual ant memories initialized with the solution found via NNM
- New Solution Importance
- A local search procedure

3.1 Introduction of Memory to the MMAS

For all the past ACO algorithms, although some ants find new good paths, they are unable to use them because of inappropriate selections of the subsequent paths. The MMAS suffers a similar defect.

On the other hand, the AS with Memory is an interesting algo-

Table 1 Concordance rates between the solution with NNM and optimal solutions.

Problem	kroA100	eil51	att48	kroC100	lin105	tsp225	lin318
Concordance Rate	79.00%	76.47%	68.75%	72.00%	75.24%	78.67%	71.70%

rithm. The memory in this algorithm plays a key role exploiting the good paths that some ants discover. The memory assumes the role to include good paths among the paths newly found by an ant and the paths of the iteration-best solution. This memory helps AS to deliver a superior performance. However, the AS with Memory algorithm does not exploit its memory advantage well because the solution in the memory in early steps is not good. Therefore, we introduced a memory that is initialized with a solution found via NNM because NNM solutions for most TSP instances have paths that are the same as those in optimal solutions as shown in **Table 1**. The problems in Table 1 have been taken from the TSPLIB and have been used to compare algorithms later in this study.

In addition, in the case of the MMAS, only the ant that has the iteration-best solution can add pheromones. Then, in this algorithm, all ants research solutions around the iteration-best solution. Therefore, we introduce the memory which is overwritten with a global-best solution in each iteration instead of the iteration-best solution to include the good paths of these solutions.

3.2 New Solution Importance

In previous ACO algorithms, although some ants find new good solutions, no ant searches around these solutions owing to a lack of additional pheromones compared to the pheromone amounts of the past global-best solutions. Therefore, we introduced New Solution Importance. We modified Eq. (3) as follows:

$$\tau_{ij}(t+1) = [\rho\tau_{ij}(t) + \Delta\tau_{ij}(t)]_{\tau_{\min}(t)}^{\tau_{\max}(t)} \tag{6}$$

$$\Delta\tau_{ij}(t) = \begin{cases} Q \times \frac{I_{ij}(t)}{L_{IB}(t)} & \text{if } path(i,j) \text{ belongs to } TOUR_{IB} \\ 0 & \text{otherwise} \end{cases}$$

$$I_{ij}(t) = \begin{cases} \frac{\tau_{\max}^{GB}(t)}{\tau_{ij}(t)} & \text{if } L_{IB}(t) < L_{GB}(t-1) \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

where $\tau_{\max}^{GB}(t)$ is the maximum pheromone amount of the path of the global-best solution in iteration t. Updating the pheromone amounts using this equation leads to sufficient pheromone amounts on each path of the new solution compared to the those on the paths of past global-best solutions.

3.3 Local Search

At each step, the proposed algorithm checks all the neighborhood solutions that are obtained by swapping the adjacent cities of the solution in the memory (the global-best solution) and memorizes the best solution of them. The number of all neighborhood solutions is equal to the number of cities.

3.4 Calculation Cost

Taking into account the procedures of the Memory, NSI, and the local search, the calculation cost of each iteration of the proposed algorithm is slightly more than that of the other algorithms such as the MMAS and the AS with Memory.

3.5 Procedure for the Proposed Algorithm

The procedure for the proposed algorithm is as follows. First, it initializes the pheromone amounts on all paths in the same manner as initialized in the MMAS. Second, each ant memory is initialized with the solution found via NNM. The first city in the memory is the starting city of the ant. Then, it iterates the next six steps for a specified number of iterations.

(1) All ants are placed in random cities and set the first city in the memory of each ant as this city.
(2) All the neighborhood solutions that are obtained by swapping the adjacent cities of the solution in the memory and the best solution is retained only if the memory is updated using the new global-best solution.
(3) Each ant creates its own solution in the same manner as that used in the AS with Memory algorithm.
(4) $\tau_{\min}$ and $\tau_{\max}$ are updated using Eqs. (4) and (5), respectively.
(5) The pheromone amount on each path is updated using Eq. (6).
(6) All solutions on the memories are overwritten with either the global-best solution of this iteration or the solution found via NNM, whichever of the two is a shorter solution.

4. Experimental Results

In this study, we tested our algorithm using symmetric the TSP. All the TSPs used in this study are taken from the benchmark TSPLIB.

4.1 Setting Parameter Values

For implementing the MMAS [6] and the AS with Memory [10] algorithms, we used the same parameter values as those used in each study for each algorithm in the following computational experiments. For the proposed algorithm, we used the same parameter values (M = *number of cities of each problem*, $\alpha = 1, \beta = 2$, and $\rho = 0.98$) as used for the MMAS algorithm, excluding the parameter Q. According to a preliminary experiment using kroA100 taken from the TSPLIB, we used $Q = 0.1$ in the following computational experiments.

4.2 Experimental Results

In this section, we present the computational results of three algorithms, i.e., the MMAS, the AS with Memory, and the proposed algorithm, using seven problems, namely kroA100, eil51, att48, kroC100, lin105, tsp225, and lin318 taken from the TSPLIB. The tour lengths of the optimal solutions of these problems are 21282, 426, 33522, 20749, 14379, 3916, and 42029, respectively.

First, we present a graph illustrating the global-best solution of each iteration in the three algorithms for the kroA100 problem (**Fig. 1**). This graph shows the average of 100 experiments. Graphs representing the other problems showed a similar behavior.

Second, **Table 2** illustrates the performance at the 1,000th it-

Table 2 Results at the 1,000th iteration for the kroA100(21282) problem. "+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, "LS" means Local Search, and "NSI" means New Solution Importance. "AS+Memory(u)" means the AS with memory. These results are the average of 200 experiments.

	MMAS	AS+Memory(u)	MMAS+Memory(u)	MMAS+Memory(i)
Success Rate of Optimal Solution	13.0%	5.50%	54.0%	56.5%
Error Rate	0.543%	1.20%	0.186%	0.0528%
Standard Deviation of Solutions	120	191	81.9	19.1
	MMAS+Memory(i)+LS	MMAS +Memory(i)+NSI	Proposed Method	
	66.5%	89.0%	91.0%	
	0.0512%	0.0419%	0.0277%	
	16.6	12.1	8.73	

Table 3 Results at the 1,000th iteration for the eil51(426) problem. "+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, and "NSI" means New Solution Importance. These results are the average of 200 experiments.

	MMAS	MMAS+Memory(u)	MMAS+Memory(i)	MMAS +Memory(i)+NSI	Proposed Method
Success Rate of Optimal Solution	55.0%	83.0%	92.5%	94.0%	94.5%
Error Rate	1.20%	0.982%	0.892%	0.909%	0.879%
Standard Deviation of Solutions	1.07	0.916	0.705	0.716	0.646

Table 4 Results at the 1,000th iteration for the att48(33522) problem. "+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, and "NSI" means New Solution Importance. These results are the average of 200 experiments.

	MMAS	MMAS+Memory(u)	MMAS+Memory(i)	MMAS +Memory(i)+NSI	Proposed Method
Success Rate of Optimal Solution	52.5%	68.0%	78.0%	80.0%	82.5%
Error Rate	0.212%	0.145%	0.0887%	0.0854%	0.0493%
Standard Deviation of Solutions	114	75.1	67.0	67.9	44.6

Table 5 Results at the 1,000th iteration for the kroC100(20749) problem. "+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, and "NSI" means New Solution Importance. These results are the average of 200 experiments.

	MMAS	MMAS+Memory(u)	MMAS+Memory(i)	MMAS +Memory(i)+NSI	Proposed Method
Success Rate of Optimal Solution	9.50%	27.5%	35.5%	63.5%	65.0%
Error Rate	0.782%	0.667%	0.665%	0.192%	0.176%
Standard Deviation of Solutions	109	127	156	75.8	67.3

Table 6 Results at the 1,000th iteration for the lin105(14379) problem. "+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, and "NSI" means New Solution Importance. These results are the average of 200 experiments.

	MMAS	MMAS+Memory(u)	MMAS+Memory(i)	MMAS +Memory(i)+NSI	Proposed Method
Success Rate of Optimal Solution	69.5%	55.0%	59.5%	84.5%	85.5%
Error Rate	0.157%	0.289%	0.305	0.109%	0.104%
Standard Deviation of Solutions	41.6	67.5	54.4	37.7	37.1

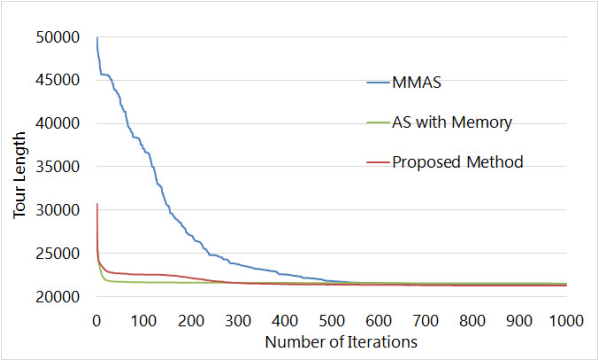

Fig. 1 Computational results for kroA100 using three different algorithms.

eration for the kroA100 problem that compares the seven algorithms. **Tables 3–6** illustrate the performance at the 1,000th iteration for the eil51, att48, kroC100, and lin105 problems, respectively. **Tables 7** and **8** illustrate the performance at the 2,000th iteration for the tsp225 problem and at the 3,000th iteration for the lin318 problem, respectively, to compare the five algorithms.

Error Rate in these tables is defined as the average of ER:

$$ER = \frac{L_{GB} - L_{OPT}}{L_{OPT}} \tag{8}$$

where L_{GB} is a global-best solution at the last iteration and L_{OPT} is an optimal solution.

"+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, "LS" means Local Search, and "NSI" means New Solution Importance, e.g., "MMAS+Memory(i)+NSI" means an algorithm that introduces the Memory initialized with NNM and the NSI; and, "AS+Memory(u)" in Table 2 means the AS with Memory.

Memory(u) means that all solutions on the memories are not initialized at the 0th iteration and are overwritten with the global-best solution after each iteration. Memory(i) means that all solutions on the memories are initialized with NNM at the 0th iteration and are overwritten with either the global-best solution of

Table 7 Results at the 2,000th iteration for the tsp225(3916) problem. "+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, and "NSI" means New Solution Importance. These results are the average of 50 experiments.

	MMAS	MMAS+Memory(u)	MMAS+Memory(i)	MMAS +Memory(i)+NSI	Proposed Method
Success Rate of Optimal Solution	0.0%	0.0%	6.0%	2.0%	6.0%
Error Rate	1.03%	1.71%	0.499%	0.504%	0.250%
Standard Deviation of Solutions	17.4	39.1	13.4	7.00	6.25

Table 8 Results at the 3,000th iteration for the lin318(42029) problem. "+" means introduction, "(u)" means uninitialized, "(i)" means initialized with NNM, and "NSI" means New Solution Importance. These results are the average of 25 experiments.

	MMAS	MMAS+Memory(u)	MMAS+Memory(i)	MMAS +Memory(i)+NSI	Proposed Method
Success Rate of Optimal Solution	0.0%	0.0%	0.0%	0.0%	0.0%
Error Rate	1.63%	2.45%	1.82%	1.02%	0.919%
Standard Deviation of Solutions	203	332	131	232	239

the iteration or the solution found via NNM, whichever of the two solutions is shorter.

Tables 2–6 show the average values of 200 experiments. Tables 7 and 8 show the average of 50 and 25 experiments, respectively.

Statistically significant differences ($p < 0.05$) were observed in pairwise comparisons of the error rate among the algorithms without pairwise comparison between the MMAS+Memory(i) and the MMAS+Memory(i)+NSI in the case of tsp225 problem as well as between the MMAS+Memory(i)+NSI and the proposed algorithm in the case of lin318 problem.

Considering the calculation costs of each algorithm, it is evident from Fig. 1 that the convergence time of the proposed algorithm is as short as that of the AS with Memory and much shorter than that of the MMAS.

The reason for the slower convergence time of the proposed method than that of the AS with Memory is the element NSI's dispersal of pheromone distribution. According to Fig. 1, the performance of the AS with Memory seems to be better than that of the proposed method. However it is evident from Table 2 that the performance of the proposed method was much better than that of the AS with Memory.

We can see that all of the introduced elements contributed toward improving the two types of results, i.e., the success rate of the optimal solutions and the error rate. These results indicate that all the improvements in the proposed algorithm are effective and that our algorithm delivered a superior performance.

5. Discussion and Conclusions

In this study, we constructed a new algorithm by introducing three elements to the MMAS, which include a memory initialized with NNM, New Solution Importance, and a local search procedure.

With the introduction of memory to the proposed algorithm, if an ant finds a good path, the ant can make good use of it, leading to shorter convergence time.

The success rate of the optimal solution of the MMAS with Memory(i) is better than that of the MMAS with Memory(u) without not only in the case of the lin318 problem but in the case of all other problems used in this study. In addition, the error rate of the MMAS with Memory(i) is better than that of the MMAS with Memory(u) in all the cases.

Therefore, on average, the MMAS introducing Memory(i) obtains a better solution than the MMAS introducing Memory(u).

In the later stages of the MMAS experimental calculations, several paths reached the pheromone trail upper limit and nearly all the other paths reached the pheromone trail lower limit. In these situations, if one ant finds a new solution, all ants are scarcely able to search around the solution in following iterations because there are numerous pheromones on the paths of past global-best solutions. Therefore, we introduced New Solution Importance, which enabled us to obtain higher success rates of the optimal solutions.

Tables 2–8 show that the proposed algorithm has the following three favorable features:

- It has the maximum success rate of the optimal solution without the lin318 problem
- It has the smallest error rate of solution among all algorithms in this paper.
- Taking into account the ratio of the MMAS's error rate to that of the proposed algorithm, the standard deviation of the solution of proposed algorithm is larger than that of the MMAS. This fact shows the potential of the proposed algorithm for solving the TSP.

Therefore, we conclude that all the three elements as well as our proposed algorithm are effective.

Our algorithm has many parameters. In terms of future prospects, we will study the method to determine these parameters.

Acknowledgments The authors would like to thank Enago (www.enago.jp) for the English language review.

References

[1] Dorigo, M., Maniezzo, V. and Colorni, A.: Ant system: Optimization by a colony of cooperating agents, *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol.26, No.1, pp.29–41 (1996).
[2] Dorigo, M. and Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation, Vol.1, No.1, pp.53–66 (1997).
[3] Dorigo, M. and Stützle, T.: Comparison of Ant System with Its Extensions, *Ant Colony Optimization*, pp.91–92, The MIT Press, Massachusetts (2004).
[4] Stützle, T. and Hoos, H.H.: MAX-MIN ant system and local search for the traveling salesman problem, *Evolutionary Computation, IEEE International Conference*, pp.309–314 (1997).
[5] Dorigo, M. and Stützle, T.: ACO plus Local Search, *Ant Colony Optimization*, pp.92–93, The MIT Press, Massachusetts (2004).
[6] Stützle, T. and Hoos, H.H.: Max-Min ant system, Future Generation

Computer Systems, Vol.16, No.8, pp.889–914 (2000).
[7] Bullnheimer, B., Hartl, R.F. and Strauß, C.: A new rank based version of the ant system - A computational study, *Central European Journal of Operations Research and Economics*, Vol.7, pp.25–38 (1997).
[8] Hlaing, Z.C. and Khine, M.A.: Solving traveling salesman problem by using improved ant colony optimization algorithm, *International Journal of Information and Education Technology*, Vol.1, No.5, pp.404–409 (2011).
[9] Liu, G. and Xiong, J.: Ant colony algorithm based on dynamic adaptive pheromone updating and its simulation, *2013 6th International Symposium on Computational Intelligence and Design*, pp.221–223 (2013).
[10] Wang, R-L., Zhao, L-Q. and Zhou, X-F.: Ant colony optimization with memory and its application to traveling salesman problem, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E95-A, No.3, pp.639–645 (2012).
[11] TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Takashi Isozaki received his B.S. degree in Engineering from Tokyo City University, Tokyo, Japan, in 2015. Since 2015, he is currently enrolled in Graduate School of Engineering, Tokyo City University. His major research interests include approximation algorithms of combinatorial optimization.

Satoshi Hasegawa received his B.E degree from Musashi Institute of Technology in March 2009. He worked on complex systems and swarm intelligence. In 2016, he joined DMM.com Labo, where he worked as a data scientist. His current work interests include complex networks, machine learning algorithms, and swarm intelligence.

Hajime Anada received his doctorate in physics from Kobe University, Japan in 1993. He is an associate professor at Graduate School of Engineering, Tokyo City University. His research interests include complex systems.