IEICE TRANS. FUNDAMENTALS, VOL.E97–A, NO.8 AUGUST 2014

PAPER

Hybrid Consultant-Guided Search for the Traveling Salesperson Problem

Hiroyuki EBARA[†a)], ***Member***, **Yudai HIRANUMA**[†*], ***and*** **Koki NAKAYAMA**[†], ***Nonmembers***

SUMMARY Metaheuristic methods have been studied for combinational optimization problems for some time. Recently, a Consultant-Guided Search (CGS) has been proposed as a metaheuristic method for the Traveling Salesperson Problem (TSP). This approach is an algorithm in which a virtual person called a client creates a solution based on consultation with a virtual person called a consultant. In this research, we propose a parallel algorithm which uses the Ant Colony System (ACS) to create a solution with a consultant in a Consultant-Guided Search, and calculate an approximation solution for the TSP. Finally, we execute a computer experiment using the benchmark problems (TSPLIB). Our algorithm provides a solution with less than 2% error rate for problem instances using less than 2000 cities.

key words: *Consultant-Guided Search, Traveling Salesperson Problem, Combinatorial Optimization Problem, Parallel Algorithm, Ant Colony Optimization*

1. Introduction

In recent years, as the speed and ability of computers has continued to improve, the amount of calculation required for new problems has increased. When problems require huge calculations on a computer, researchers may apply known methods to solving the combinatorial optimization problem. The combinatorial optimization problem is a problem to determine the minimum or maximum valued combination based on the constraints given to evaluate the value of objective functions.

It is a rare case when the result one wants to find for an optimal value for must be strictly a combinatorial optimization problem. An approximate solution with a certain degree of accuracy is often acceptable instead, because in general, the solution can be obtained in a short period of time. It is generally known that the execution time for determining the exact solution increases exponentially with the size of the problem. Therefore, obtaining an approximate solution sufficiently close to the optimal solution is faster than finding the exact value. Methods for solving approximate solutions have been studied [1]. Metaheuristics, among the many approximate solution methods, for example, have been extensively studied in particular, because this method produces a general solution that can be adapted for many problems [2].

Swarm Intelligence is a metaheuristics that has been extensively studied as a method for solving optimization problems in recent years [3]. In addition, the Consultant-Guided Search (CGS) algorithm has recently been proposed using a Swarm Intelligence algorithm [4]–[6]. This algorithm is inspired by the way real people make decisions based on advice received from consultants. Human behavior is complex, but CGS uses virtual people that follow only simple rules. Also, there is no leadership role in which to organize people; all people act on their own. Each virtual person is responsible for being both a client and a consultant. The consultant builds a strategy (solution) to lead the client to create a solution, and the client creates a solution based on the strategy that the consultant builds.

In this research, we find a better approximate solution for the traveling salesperson problem (TSP), which is a typical example of a combinatorial optimization problem, than the existing CGS algorithm, by using a hybrid CGS method with an Ant Colony System (ACS). ACS [7] is one of the extended methods of Ant Colony Optimization (ACO) [8]. ACS is different from ACO in that it updates the pheromone every time virtual ants choose a city. Our proposed method in this research uses the ACS solution as a consultant in CGS to build a strategy, and the pheromone in ACS is carried over in CGS. In addition, the proposed method performs parallelization using MPI communication. Computer experiments were carried out on a PC cluster, where we verify the effectiveness of our method.

2. Traveling Salesperson Problem

The Traveling Salesperson Problem (TSP) is the problem of finding the shortest possible distance in a cyclic path from a starting city, to each n city once, and back to the starting city.

When C_{ij} is the distance between city i and city j, and $V = \{1, ..., n\}$ is the set of n cities, the formula to minimize the objective function is as follows:

$$f(x) = \sum_{k=1}^{n-1} C_{x(k)x(k+1)} + C_{x(n)x(0)} \tag{1}$$

$x(k) = i$ indicates that the kth city visited is i.

3. Consultant-Guided Search

Consultant-Guided Search (CGS) is one of the recently proposed methods based on metaheuristics which can directly exchange information between humans [4]–[6]. When a

Manuscript received October 16, 2013.
Manuscript revised March 5, 2014.

[†]The authors are with Kansai University, Suita-shi, 564-8680 Japan.
[*]Presently, with NTT Comware Corporation.
a) E-mail: ebara@kansai-u.ac.jp

DOI: 10.1587/transfun.E97.A.1728

client decides an action, that action is sometimes based on advice from a consultant. CGS obtains the solution based on the relationship between the consultant and the client receiving the advice. The virtual person in this method plays the role of both the consultant and the client. The advice the consultant gives to the client is a solution, called a strategy. Since the virtual person acts as both the client and the consultant, the method is actually divided into modes to build a strategy as a consultant and to create a solution as a client. These modes are called sabbatical modes and normal modes, respectively.

First, the consultant needs to build a strategy, which is a solution that will give advice to the client. Therefore, all of the virtual persons start the search from a sabbatical mode. In the sabbatical mode, the virtual person, as the consultant, builds a solution to guide the client. The solution is built on the basis of the distance between the cities. The sabbatical mode continues until it creates plural solutions to build as good a strategy as possible. When the sabbatical mode is completed, the algorithm moves to the normal mode. In the normal mode, the virtual person, as the client, selects a consultant and creates a new solution based on the solution the consultant produced. If the solution the client creates is better than the strategy the consultant created, the reputation of the consultant rises and the consultant is more likely to be selected by clients. The reputation of the consultant goes down gradually. If the consultant's reputation falls below a certain value, it migrates to the sabbatical mode and then starts to rebuild a strategy. CGS can find the solution using the following algorithm.

1. The algorithm creates a virtual person, and sends that person to the sabbatical mode.
2. In the sabbatical mode, each virtual person creates a solution according to a formula of strategy construction. In the normal mode, each virtual person creates a solution according to a formula of solution creation.
3. The algorithm updates strategies after each virtual person generates a solution. If the solution is better than before in the sabbatical mode, it replaces a previous strategy. If the solution is better than the strategy the consultants used to create the solution in normal mode, it replaces the previous strategy.
4. The algorithm updates the strategies.
5. If the consultant's reputation falls below a certain value, the consultant moves to the sabbatical mode. If the consultant builds a strategy the predetermined number of times, it changes to normal mode.
6. If the algorithm meets the criteria, it terminates the search.

When a virtual person builds a strategy in sabbatical mode, the mode basically selects the city which is the shortest possible distance between the city where a virtual person is now and the next city. Also, in some cases, the virtual person decides the next city probabilistically. The formula the virtual person uses in the sabbatical mode is as follows.

$$j = \begin{cases} argmin_{l \in N_i^k}\{d_{il}\} & if\ a \leq a_0 \\ J & otherwise \end{cases} \tag{2}$$

$$p_{ij}^k = \frac{(1/d_{ij})^\beta}{\sum_{l \in N_i^k}(1/d_{ij})^\beta} \tag{3}$$

$l \in N_i^k$ represents that city l is included in the executable neighborhood around a virtual person k in city i. d_{il}, $a(0 \leq a \leq 1)$, $a_0(0 \leq a_0 \leq 1)$ represents the shortest possible distance between the city i and l, the random variable, and the parameter, respectively. J is the random variable selected in the probability distribution obtained by p_{ij}^k, and β is a parameter.

In normal mode, the client selects the next city based on the strategy of the consultant. More specifically, the client selects the consultant from the normal mode, including itself, based on the reputations of the consultants. Next, the client creates the new solution by the following formula, using the strategy that the selected consultant created in the sabbatical mode.

$$j = \begin{cases} v & if\ v \neq null \wedge q \leq q_0 \\ argmin_{l \in N_i^k}\{d_{il}\} & if (v = null \vee q > q_0) \\ & \wedge b \geq b_0 \\ J & otherwise \end{cases} \tag{4}$$

v represents the city that is connected to the city where the consultant's solution is now. q, $b(0 \leq q, b \leq 1)$ is a random variable, q_0, $b_0(0 \leq q_0, b_0 \leq 1)$ is a parameter.

The reputation is changed by the obtained solution based on the consultant's advice and the reputation value of the consultant. An update of the reputation is performed according to the following formula.

$$p_k = \frac{reputation_k^\alpha result_k^\gamma}{\sum_{c \in C} reputation_c^\alpha result_c^\gamma} \tag{5}$$

C is a set of consultants in normal mode, α and γ are parameters. *reputation* is the reputation value of the consultant; *result* represents the reciprocal of the consultant's solution. In addition, *reputation* is gradually reduced by the following formula;

$$reputation_k \leftarrow reputation_k(1 - r) \tag{6}$$

r represents the reduction rate.

4. Related Works

4.1 Consultant-Guided Search

CGS, used in this study, is a new metaheuristic approach that was developed recently through Lordache's study [4]. In addition to TSP, CGS has also been applied to the Quadratic Assignment Problem (QAP) [6] by the same author, and CGS has put out good results compared to the Max-Min Ant System (MMAS) [9]. In this study, the author positions CGS as a hybrid metaheuristic approach that combines this new approach with the concepts of other optimization techniques. The construction of the solution in CGS is similar to the method in which the set of ants in Ant Programming (AP) [10], [11] builds the solution. The solution is selected from the candidate set probabilistically, and desirability is given by the amount of pheromone in AP and the reputation and strategy of the consultant in CGS. The author also insists that the operation of the client selecting a consultant is similar to the waggle dance in Bee Colony Optimization (BCO).

4.2 Ant Colony Optimization

In the study [12] by Manfrin et al., a comparative experiment was performed for 4 parallel algorithms of MMAS. Both search processes for synchronous communication and asynchronous communication were examined. The authors also tested the communications independently in parallel, such that communication is not performed. The parallel algorithms were: Completely-Connected, Replace-Worst, Hypercube and Ring. Completely-Connected is the method in which the solution from the PC with the best solutions in the entire PC cluster are sent to all other PCs. Replace-Worst is the method in which the PC with the best solution sends its own solution to the PC with the worst solution in the entire PC cluster. Hypercube is the method in which each PC is located on the vertices of a hypercube, and only share the best solution with PCs on vertices that are directly connected to it. Ring is the method in which each PC is connected in a ring, and sends the best solution to the next PC. The method of running independently without sending the solution information gave the best solution based on the experimental result. The 4 parallel approaches reported that all the PCs quickly converged on the same solution. The 4 parallel approaches used the best solution when they did global pheromone updates every iteration. Therefore, each PC is estimated to respectively explore the nearest search space, the amount of pheromone increases that occur rapidly at the same place, and the processes converged at an early stage.

Hassan et al. [13] has done research on the number of virtual ants in ACS. In this study, he compares the quality of the solution at each iteration. If it is worse than the previous iteration, his process increases the number of ants. In addition, his process does an adaptive local search each time. The number of ants will not necessarily increase indefinitely; the maximum value is set to 1.5 times the initial value at first. However, if the number of ants is increased up to the maximum value, the maximum value is increased a little over time. An adaptive local search is used λ-opt(λ is between 2 and 5). This study performs better than ACS and MMAS based on the comparison experiment.

There has also been research on a hybrid metaheuristic approach with parallelization such as the combination of ACS and the Genetic Algorithm (GA). In the study of Chen et al. [14], the solution obtained by ACS is improved by GA. Studies that combine ACS and GA are frequent. In this study, each ant of ACS performs the search as a chromosomes of GA. From the result, this search speeds up the convergence by exchanging the information of the best solution with the other group, and updating the entire pheromone information. This study tried 7 different ways of exchanging the solution information. The first way was to share the best solution in all groups. In the second way, group numbers were represented by binary values, and the algorithm exchanged solutions among the groups that differed in the last 1 bit. The third way was to pass the solution to the next group. In the fourth way, group numbers were represented by binary values, and the algorithm exchanged solutions among groups which differed by 1 bit. The fifth, sixth, and seventh ways combined the first and second, the first and third, and the first and fourth, respectively.

Additional studies using Ant Colony Optimization other than those mentioned above include [15]–[19].

4.3 Other Metaheuristic Approaches

In the study [20] of Shi et al., the authors propose a hybrid algorithm combining Particle Swarm Optimization (PSO), which is a kind of swarm intelligence, and GA. PSO and GA created the solution using the individuals independently. This hybrid focused on this similarity. First, PSO and GA searched multiple times independently. After that, the method exchanged the individuals between PSO and GA, and repeated the search again. This algorithm has produced good results compared to the traditional PSO.

There are several other metaheuristic approaches that have also been recently proposed in addition to CGS, such as the Migrating Birds Optimization [21] based on the behavior of a swarm of migratory birds flying in a V shape, the Bat Algorithm [22], based on the behavior of echo position measurements in bats, and Ray Optimization [23] using refracted light based on Snell's law.

5. Proposed Method

5.1 Experimental System

The experimental environment used in this study has been constructed using SCore, which is a free software MPI environment distributed by a PC Cluster Consortium [24]. There are 10 PCs for calculation processing and 1 server in the laboratory, connected to the same LAN over a Gigabit Ethernet network switch, and the cluster can communicate with each other MPI environment (Fig. 1). Figure 1 shows the details of the experimental system used in this study. We used 1 server and the PC for calculation processing is 10 units. Tables 1, 2 shows the performance.

5.2 Proposed Method

In this study, we aimed at obtaining a better approximate solution for TSP by using the solution in ACS for CGS. The main proposed method uses the strategy of the consultant and carries over the pheromone information. The method enables a more effective search by using the solution obtained in ACS for the strategy of the consultant in CGS. The

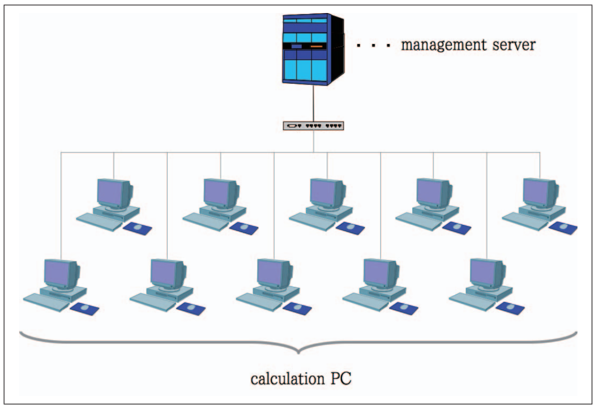

Fig. 1 Experiment system.

Table 1 Performance of server.

CPU	Intel Xeon X5470 3.33 GHz
Memory	4 GB
OS	CentOS 5.6
MPI	SCore version 7.0.1

Table 2 Performance of calculating PCs.

CPU	Intel Core2Duo E6850 3.00 GHz
Memory	4 GB
OS	CentOS 5.6
MPI	SCore version 7.0.1

carryover of the pheromone information makes it possible to explore a wide range by reusing the pheromone information in the next ACS.

The following is a description of the characteristics of the proposed method.

1. The strategy of the consultant
   The client creates the solution using the strategy of the consultant in CGS. The formula for construction of the strategy of the consultant is Eq. (2), which only uses the distance information between cities. Therefore, it is thought that similar solutions are created often, although there is some randomness. In this case, we avoid such a situation by using the solution obtained by other algorithms.
   In this proposed method, a virtual ant in ACS does a search as a virtual person in CGS. In the phases of ACS, each virtual ant always keeps the best solution of all the solutions obtained. After this phase, each virtual ant and its best solution is taken to CGS. CGS starts to search for the solution using the strategy of the consultant. Thus, the client can create a variety of solutions because the advice for the solution is obtained indirectly by pheromones and the probabilistic selection of the city using the distance between the cities.
2. The carryover of the pheromone information
   In this proposed method, CGS searches after converging on ACS. At the end of ACS, a large difference exists in the amount of pheromones between cities included in the best solution at that time, as opposed to the amount between other cities. In conventional CGS, the distance between the cities is used both in the construction of the strategy of the consultant and in the creation of the solution of the client. Because conventional CGS does not use the indirect information, such as pheromones, it does not affect the search directly or even take it over. However, CGS can search intensively in the close range of the previous search, although slightly differently in ACS, by taking over the pheromone information from ACS after this search and updating the information from time to time.

5.3 Parallelization

The proposed method uses the parallelization of MPI. This parallelization can be used to carry out a more extensive search than in the case of a single PC to share information between the plural number of PCs.

The calculation process of the proposed method is started by the server that manages the PC cluster by turning on the job for each PC, for all calculations. Each job is assigned to 1 unit of the PC for the calculation. The management server performs both the work to be assigned to the PC for the calculation of the job and the task of collecting the calculated result from the PC after all the calculations have been done. After the calculation is started, each PC for the calculation shares the information by communicating with the MPI for the PC. In this way, the calculation is performed in a multi-process, the solution space is not divided, and the calculation process is performed independently, except for the information to be shared. In addition, the communication is performed asynchronously; the calculation process is not interrupted by the communication process. The content of the information that the PC for the calculation communicates and shares in the entire PC cluster is the best solution each PC has calculated.

In the proposed method, the PC for the calculation holds the pheromone information of each calculation. When the PC updates the entire pheromone information, it uses the solutions that have been created. Normally, the PC updates with the best solution in the solutions made by the virtual ants of each PC. However, when the solution is not updated for a while, early convergence might possibly raise the pheromone on the path which constitutes the best solution. Therefore the PC can search a more diverse range by slowing convergence to use the solution of another PC occasionally.

A schematic representation of parallelism of the proposed method is shown in Fig. 2.

5.4 Proposed Algorithm

The proposed algorithm consists of Phase1 (ACS) and Phase2 (CGS) as shown below.

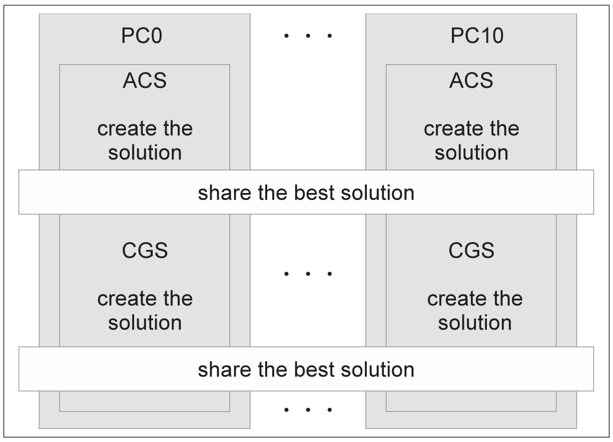

Fig. 2 Parallization of the proposed method.

1. The creation of virtual ants (persons)
   The number of virtual agents is decided randomly between 20 and 30.
2. Initialize the pheromone
   The initial value is

$$\frac{1}{(the\ number\ of\ cities\ N)} \times \frac{1}{(the\ solution\ created\ by\ the\ nearest\ neighborhood\ method)} \quad (7)$$

3. Phase1
   This phase is continued repeatedly through a search using the virtual ants until the solution can no longer improve within a specified number of iterations.
4. Set the virtual persons to normal mode
   Set the virtual persons to the normal mode to use the solution created by each of the virtual ants as the strategy of the consultant in the proposed method.
5. Phase2
   Start the search after assigning the solutions from the virtual ants (persons) found in Phase 1 to the strategies of each consultant.
   Update the pheromone in this phase.
   This phase is continually repeatedly through a search using the virtual persons until the solution can no longer improve within a specified number of iterations.
6. Iterate the search
   After the end of Phase 2, go back to Phase 1. In Phase 1 after Phase 2, if the solution is not improved, initialize the pheromone.
7. Termination condition
   The termination condition is determined by the search time. This condition will terminate the search after an interval is set in advance. It outputs the best solution at the end.

The flowchart of the proposed method is shown in Fig. 3.

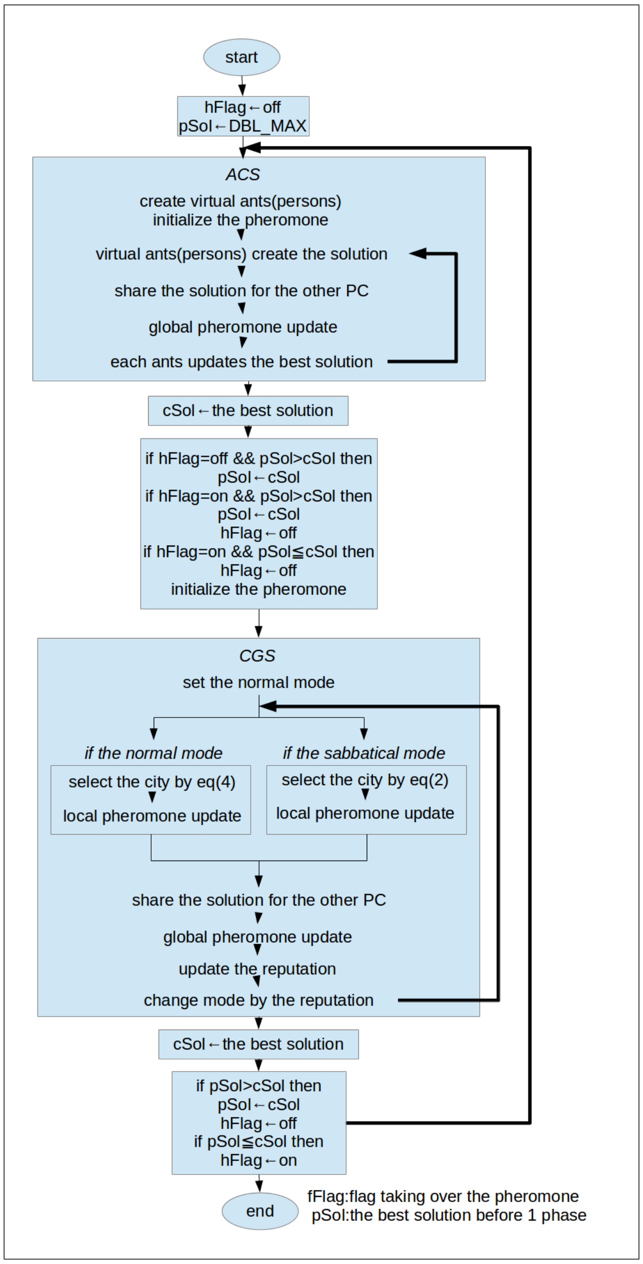

Fig. 3 Flowchart of the proposed method.

6. Experimental Result

6.1 Experimental Environment

TSP instances treated in this study are obtained from the TSPLIB [25], by distributing the TSP benchmark. We created an experiment for the following 7 problem instances, *rat*575, *rat*783, *pr*1002, *u*1060, *u*2152, *pr*2392, and *pcb*3038.

We do an experiment for more than one item in order to measure the accuracy of the proposed method. First, we compare the proposed method (ACCGS), ACS (only Phase

Table 3 Parameters (common).

Parameter	Value
nthread	2
npc	10
nagent_low	20
nagent_high	30
lph	0.10
gph	0.10

Table 4 Parameters (ACS).

Parameter	Value
ACS_α	1.0
ACS_β	3.0
ACS_q_0	0.99

Table 5 Parameters (CGS).

Parameter	Value
a_0	0.9
CGS_α	7.0
CGS_β	12.0
γ	7.0
CGS_q_0	0.98
b_0	0.98
leave	100
repini	6
repmax	40
repmin	1
bonus	8
fadingranks	3
r_0	3.0×10^{-7}
w	1000
k_w	3.0

1), and CGS (only Phase 2) in order to verify the effectiveness of the proposed method (Experiment 1). Second, we carried out a experiment for the case of changing the number of agents in order to verify the number of virtual agents (ants or persons) in the proposed algorithm (Experiment 2). Finally, we compare *ACCGS*, as the proposed method, and *ACCGS_PHINIT*, which initializes the pheromone information after every Phase 2, in order to verify the influence on the results when the ACCGS carries over the pheromone information from Phase 1 to Phase 2 and created the update of the pheromone information in Phase 2.

We show the various parameters of the experiment in Tables 3, 4, and 5.

In Table 3, the parameter values in common, such as the number of virtual agents, and the search time are shown. *nthread* is the number of threads running on a single PC. *npc* is the number of PCs. *nagent_low* is the lower limit of the number of virtual agents to perform the search, and *nagent_high* is the upper limit of the number of virtual agents to perform the search. The number of virtual agents is determined randomly by each process between the lower and upper limits. *lph* is the rate of increase of the pheromone in the local pheromone update. *gph* is the rate of increase of the pheromone in the global pheromone update.

Table 4 shows the parameter values concerned in Phase 1 of ACS. ACS_α, ACS_β are the weight of the distance between the cities based on a pseudo-random proportional rule, and the weight of the pheromone amount, respectively. ACS_q_0 is the parameter to judge how to determine the next city based on a pseudo-random proportional rule.

Table 5 shows the parameter value concerned in Phase 2 of CGS. a_0 is the same as ACS_q_0 of Table 4. CGS_α is the weight of the reputation of the consultant for use when calculating the probability to select the consultant. CGS_β is the weight of the distance based on the pseudo-random proportional rule. γ is the weight of the personal preference for use in calculating the probability to select the consultant. b_0, CGS_q_0 is the parameter to judge whether to determine the next city based on the city the consultant recommends. *leave* is the count to construct the strategy as the consultant in sabbatical mode. *repini*, *repmax*, *repmin* are the initial, maximum, and minimum values of reputation, respectively. The reputation increases or decreases between *repmin* and *repmax*, and does not exceed this range. *bonus* is a parameter to add specially to the reputation of the consultant, who advised the client to seek the best solution. r_0, *fadingranks* are the parameters needed to calculate the percentage for reducing the reputation, and the number of consultants needed, respectively. w is the period needed to allow the count to succeed. k_w is the parameter for the difference between the rate of decrease when the algorithm does not succeed, and the rate of decrease when it does succeed.

In each experiment, the same parameters are used for all problem instances except the search time. The search time is determined by the number of cities: 2 hours if under 2000 cities, 3 hours if between 2000 and 3000 cities, and 5 hours if over 3000 cities. In addition, these parameter values are determined with reference to preliminary experiments. The average value and the minimum value are calculated from the search results performed 10 times. We performed simulations 15 times for some problem instances as a trial. The simulation result hardly changed. Therefore, we consider 10 simulations adequate.

6.2 Experiment1 (Performance Evaluation of the Proposed Method)

We compared ACCGS (the proposed method), ACS (only Phase 1), and CGS (only Phase 2) in Experiment 1.

The algorithm of *ACS* initializes the pheromone information every time if the termination condition is met. All virtual agents are started in the sabbatical mode in *CGS* as is the case in the conventional Consultant-Guided Search. The calculation cost decides each problem's instances: rat575, rat783, pr1002 and u1060 were set to 2 hours search time and u2152 and pr2392 were set to 3 hours; pcb3038 was set to 5 hours. We show these experimental results in the

Table 6 Result of experiment 1 (compare *ACS*, *CGS*, *ACCGS*).

Problem	Algorithm	Average	Error rate[%]	Minimum	Error rate[%]
rat575	*ACS*	6862.87	1.33	6835.71	0.93
	CGS	6861.01	1.30	6850.02	1.14
	ACCGS	**6853.19**	**1.18**	**6828.37**	**0.82**
rat783	*ACS*	8945.97	1.59	8924.31	1.34
	CGS	8994.58	2.14	8948.63	1.62
	ACCGS	**8920.45**	**1.30**	**8882.73**	**0.87**
pr1002	*ACS*	262337.06	1.27	261647.84	1.00
	CGS	264797.42	2.22	264132.46	1.96
	ACCGS	**261909.04**	**1.11**	**260981.47**	**0.75**
u1060	*ACS*	229791.05	2.54	228346.92	1.90
	CGS	229125.39	2.25	227798.83	1.65
	ACCGS	**228114.18**	**1.79**	**227333.66**	**1.45**
u2152	*ACS*	65809.63	2.42	65628.97	2.14
	CGS	67446.90	4.97	66761.69	3.90
	ACCGS	**65621.88**	**2.22**	**65380.55**	**1.88**
pr2392	*ACS*	388383.71	2.74	384362.46	1.71
	CGS	405769.24	7.34	402506.64	6.47
	ACCGS	**386072.60**	**2.13**	**384499.13**	**1.67**
pcb3038	*ACS*	159241.46	15.65	158343.06	15.00
	CGS	158333.16	14.99	157919.31	14.69
	ACCGS	**156661.70**	**13.78**	**155358.11**	**12.83**

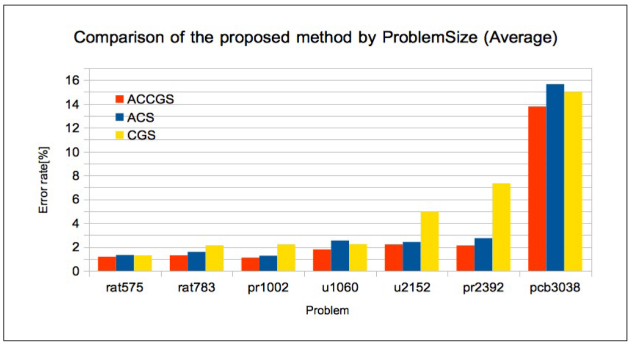

Fig. 4 Result of experiment 1 (compare *ACS*,*CGS*,*ACCGS*)(Average).

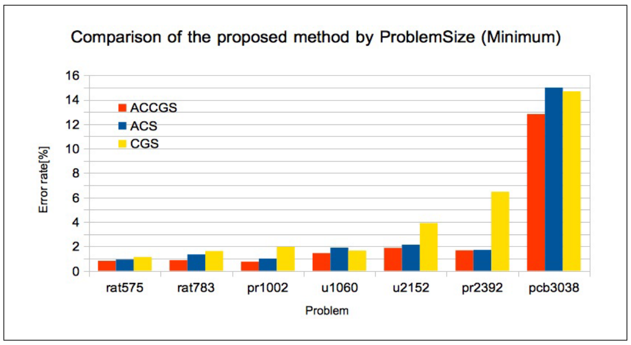

Fig. 5 Result of experiment 1 (compare *ACS*,*CGS*,*ACCGS*) (Minimum).

following: Table 6 and Figs. 4 and 5.

Various settings are shown in Tables 3, 4, and 5. In Table 6, the results obtained by *ACCGS*, *ACS*, and *CGS* for each problem instance are shown. The bold values represent the best. These are derived average values, and minimum values using 10 search results. The error rate is the value obtained by the following equation from the results derived from this experiment and the known optimum value published in the TSPLIB [25].

Table 7 Optimum values of TSPLIB problem instances.

Problem	Best known solution
rat575	6773
rat783	8806
pr1002	259045
u1060	223094
u2152	64253
pr2392	378032
pcb3038	137694

$$p = \frac{x - x_{knownbest}}{x_{knownbest}} \times 100 \qquad (8)$$

p is the error rate, $x_{knownbest}$ is optimum, x is the result. In Experiment 1, the optimum solution shown in Table 7 could not be found in all problem instances.

Figures 4 and 5 is a graphs of Table 6. In Figs. 4, 5, the horizontal axes represents problem instances and the vertical axes is the error rate of a known optimal value.

From Table 6 and Figs. 4 and 5, it can be seen that ACCGS is better than both the CGS and the ACS algorithms that make up the proposed method in all instances. Since CGS uses only distance information between cities, numerous similar strategies are built. Using ACS with both the pheromone information and the distance information diversifies the strategy of the consultant, and therefore the search range of Phase 2 spreads, and a good solution is obtained. In addition, since the algorithm carries over the pheromone information, the algorithm was able to intensively explore the close space from the range in the previous search in Phase 1.

As shown in Table 6, ACS is good at solving rat783, pr1002, u2152, and pr2392, and CGS is good at sloving rat575, u1060, and pcb3038. From this, we believe that there are strong and weak points in each problem. Therefore, ACCGS can achieve a variety of searches by combining both CGS and ACS in ACCGS.

6.3 Experiment2 (Measurement of the Effectiveness of Changing the Number of Virtual Agents)

We created an experiment to verify whether the number of virtual agents really uses the optimal range. Specifically, the experiment looks at how much change appears by changing the range on the number of virtual agents. We conducted experiments on the range of the number of virtual agents on a total of 4 patterns of 3 to 10, 10 to 20, 20 to 30, and 30 to 40. The calculation cost decides each problem's instances: rat575, rat783, pr1002, and u1060 were set to 2 hours search time and u2152, and pr2392 were set to 3 hours; pcb3038 was set to 5 hours as well as Experiment 1. We show the experimental results in Table 8 to Table 14 and Fig. 6 to Fig. 12. In Experiment 2, the optimum solution shown in Table 7 was not able to be found in all problem instances.

From the results of Table 8 to Table 14 and Fig. 6 to Fig. 12, we see that the error rate is minimum using the range of 20 to 30 virtual agents, although there is a dif-

Table 8 Result of experiment 2 (comparison of the number of virtual agents in *rat*575).

Number of virtual agents	Average	Error rate[%]	Minimum	Error rate[%]
3to10	6864.96	1.36	6843.24	1.04
10to20	6857.37	1.25	6839.94	0.99
20to30	**6853.19**	**1.18**	**6828.37**	**0.82**
30to40	6862.99	1.33	6842.96	1.03

Table 9 Result of experiment 2 (comparison of the number of virtual agents in *rat*783).

Number of virtual agents	Average	Error rate[%]	Minimum	Error rate[%]
3to10	8997.28	2.17	8967.23	1.83
10to20	8929.31	1.40	8904.30	1.12
20to30	**8920.45**	**1.30**	**8882.73**	**0.87**
30to40	8945.81	1.59	8913.62	1.22

Table 10 Result of experiment 2 (comparison of the number of virtual agents in *pr*1002).

Number of virtual agents	Average	Error rate[%]	Minimum	Error rate[%]
3to10	268527.30	3.66	266944.66	3.05
10to20	267237.61	3.16	262022.70	1.15
20to30	**261909.04**	**1.11**	**260981.47**	**0.75**
30to40	264722.44	2.19	263820.92	1.84

Table 11 Result of experiment 2 (comparison of the number of virtual agents in *u*1060).

Number of virtual agents	Average	Error rate[%]	Minimum	Error rate[%]
3to10	233159.44	4.05	232084.95	3.57
10to20	234290.12	4.55	231018.83	3.09
20to30	**228114.18**	**1.79**	**227333.66**	**1.45**
30to40	229636.09	2.47	228032.18	1.76

Table 12 Result of experiment 2 (comparison of the number of virtual agents in *u*2152).

Number of virtual agents	Average	Error rate[%]	Minimum	Error rate[%]
3to10	67615.59	5.23	67362.93	4.84
10to20	67104.60	4.44	65572.57	2.05
20to30	**65621.88**	**2.22**	**65380.55**	**1.88**
30to40	65913.38	2.58	65671.55	2.21

Table 13 Result of experiment 2 (comparison of the number of virtual agents in *pr*2392).

Number of virtual agents	Average	Error rate[%]	Minimum	Error rate[%]
3to10	408516.71	8.07	406290.32	7.48
10to20	404778.15	7.07	387133.62	2.41
20to30	**386072.60**	**2.13**	**384499.13**	**1.67**
30to40	389824.76	3.12	385014.89	1.85

ference for each instance, as the number of times to select the short distance branch increases. If the number of virtual agents increases, it would lead to a local optima immediately, as the case to select short edges increases.

6.4 Experiment3 (Evaluation If the Algorithm Does not Carry over Pheromone Information)

The results of Experiment 2 found the number of virtual agents is appropriate in the proposed method. In this section, we make an experiment to determine the influence that carryover of the pheromone information has on the search.

In the proposed method, the pheromone information is carried over to Phase 2. In Phase 2, the pheromone information is updated. In this experiment, *ACCGS_PHINIT*, which performs the initialization of pheromone information after the end of each Phase 2 is compared with the proposed method, and the effectiveness of the carryover of the pheromone information was confirmed. The calcula-

Table 14 Result of experiment 2 (comparison of the number of virtual agents in *pcb*3038).

Number of virtual agents	Average	Error rate[%]	Minimum	Error rate[%]
3to10	156930.45	13.97	156335.89	13.54
10to20	**156052.67**	**13.33**	155485.25	12.92
20to30	156661.70	13.78	**155358.11**	**12.83**
30to40	157284.31	14.23	156070.22	13.35

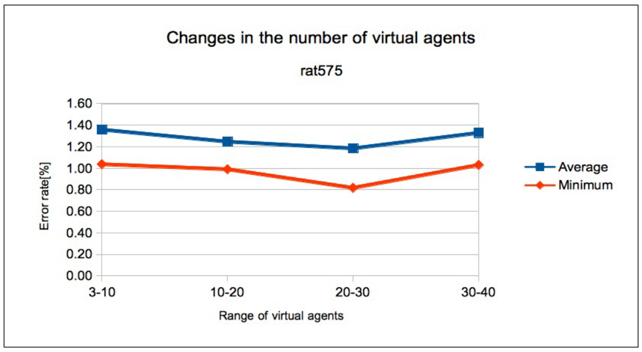

Fig. 6 Result of experiment 2 (change the number of virtual agents in *rat*575).

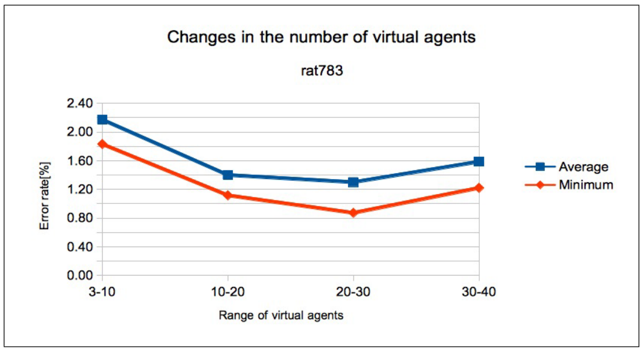

Fig. 7 Result of experiment 2 (change the number of virtual agents in *rat*783).

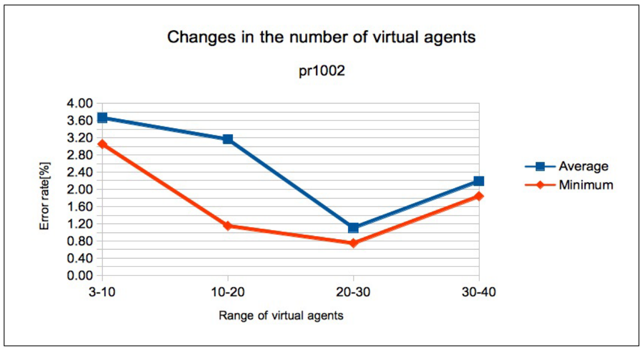

Fig. 8 Result of experiment 2 (change the number of virtual agents in *pr*1002).

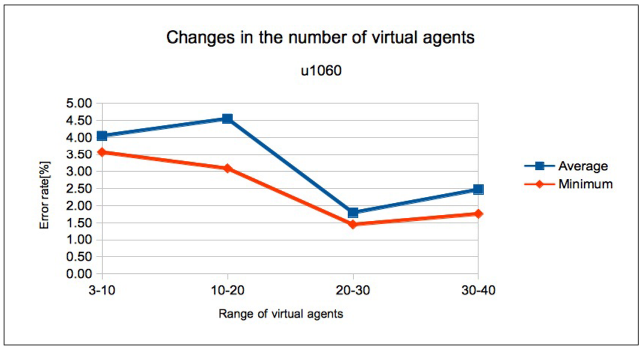

Fig. 9 Result of experiment 2 (change the number of virtual agents in *u*1060).

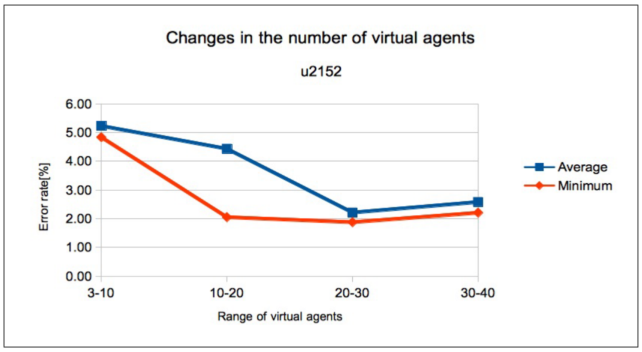

Fig. 10 Result of experiment 2 (change the number of virtual agents in *u*2152).

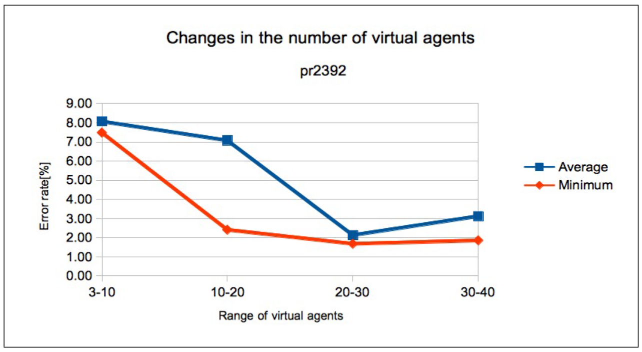

Fig. 11 Result of experiment 2 (change the number of virtual agents in *pr*2392).

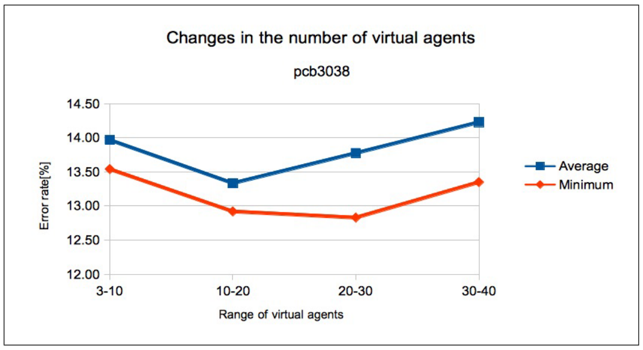

Fig. 12 Result of experiment 2 (change the number of virtual agents in *pcb*3038).

Table 15 Result of experiment 3 (comparison of *ACCGS*, *ACCGS_PHINIT*).

Problem	Algorithm	Average	Error rate[%]	Minimum	Error rate[%]
rat575	*ACCGS*	**6853.19**	**1.18**	**6828.37**	**0.82**
	ACCGS_PHINIT	6854.00	1.20	6839.38	0.98
rat783	*ACCGS*	**8920.45**	**1.30**	**8882.73**	**0.87**
	ACCGS_PHINIT	8936.33	1.48	8898.46	1.05
pr1002	*ACCGS*	**261909.04**	**1.11**	**260981.47**	**0.75**
	ACCGS_PHINIT	262231.25	1.23	261480.02	0.94
u1060	*ACCGS*	**228114.18**	**1.79**	**227333.66**	**1.45**
	ACCGS_PHINIT	229434.49	2.38	227993.24	1.74
u2152	*ACCGS*	**65621.88**	**2.22**	**65380.55**	**1.88**
	ACCGS_PHINIT	65852.90	2.49	65576.61	2.06
pr2392	*ACCGS*	**386072.60**	**2.13**	**384499.13**	**1.67**
	ACCGS_PHINIT	388325.22	2.72	384571.95	1.73
pcb3038	*ACCGS*	**156661.70**	**13.78**	**155358.11**	**12.83**
	ACCGS_PHINIT	157644.27	14.49	156540.29	13.69

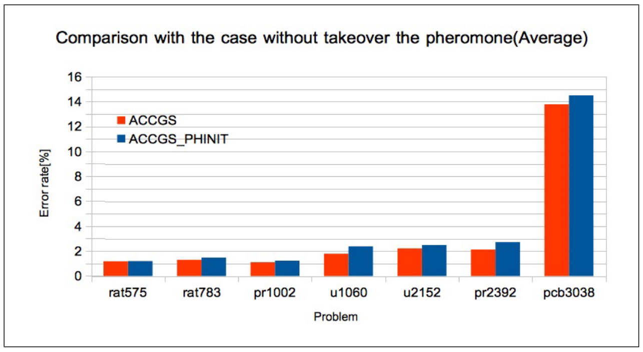

Fig. 13 Result of experiment 3 (comparison of *ACCGS* with *ACCGS_PHINIT*) (Average).

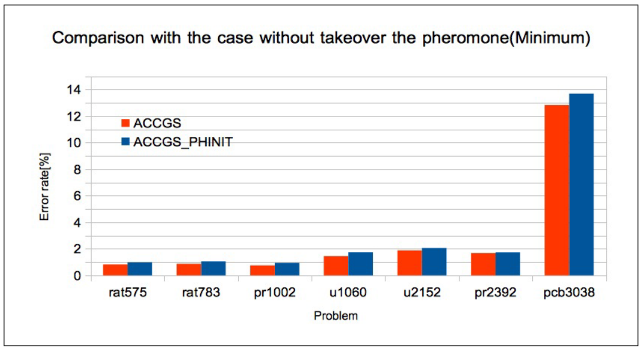

Fig. 14 Result of experiment 3 (comparison of *ACCGS* with *ACCGS_PHINIT*) (Minimum).

tion cost decides each problem's instances: rat575, rat783, pr1002, and u1060 were set to 2 hours search time and u2152, and pr2392 were set to 3 hours; pcb3038 was set to 5 hours as well as Experiment 1. We show the experimental results in Table 15 and Figs. 13, 14.

From the results in Table 15 and Figs. 13 and 14, the proposed method of carrying over the pheromone information was superior in all cases. In Experiment 3, the optimum solution shown in Table 7 could not be found in any problem instances.

The difference between the amount of pheromone on the path to configure the best solution and the amount of pheromone of the other edges is large. However, the difference becomes small by continuing to update in Phase 2. This indicates that the algorithm intensively explored the solution near the last search in Phase 1. In CGS, creating a search without being dependent on the carryover of the pheromone information is possible. As a result, the algorithm can reduce the difference in the amount of the pheromone between cities. In addition, examining the nearby areas searched in the previous iteration to increase the possibility of moving to cities that have not been previously selected is possible. MMAS has a smoothing function called PTS, and this adds to the pheromone for edges with a small amount of pheromone. The carryover of the pheromone information in the proposed method resembles PTS.

It is important that a heuristic keep a balance between the exploration of new areas in the search space and the exploitation of promising areas already searched. In ACS, the heuristic enhances the convergence of the search regions by the pheromone information. On the other hand, the convergence in the search for the solution is discarded in $ACCGS\_PHINIT$. However, it is held in $ACCGS$. In CGS, the heuristic enhances the diversity by sabbatical leave, which allows regions of the search space to abandoned that are no longer promising and allows new regions to be explored. Thereby, the proposed method (ACCGS) not only improves convergence, but also maintains diversity.

7. Conclusion

In this study we proposed the Hybrid Consultant-Guided Search for the Traveling Salesperson Problem in a parallel environment. The features of the proposed method include the following 3 points. First, we used the best solution shared in parallel when updating the global pheromone of ACS periodically in order to increase the diversity of the solution. Second, in CGS, we used the best solution each virtual ant obtained in ACS. Third, we updated the pheromone in CGS in addition to ACS.

The results of Experiment 1 indicate that the search performance of ACCGS outperforms ACS and CGS. ACCGS got better solutions by setting the solutions obtained using the pheromone information as the strategy of consultant.

The results of Experiment 3 indicate the effectiveness of carrying over the pheromone information in ACCGS. By carrying over the pheromone information, the search method intensively searches the solution space where the pheromone information has accumulated in Phase 1. In Phase 2, ACCGS sets the solution obtained in Phase 1 as the strategy of consultants. In addition, CGS has sabbatical leave that allows to regions of the search space to be abandoned that are not promising and allows new regions to be explored. The proposed method, therefore can search more diversely and intensively, because it obtained better approximate solutions using the pheromone information as the strategy of consultants more than the existing CGS and ACS.

In future research, we will compare the performance of the proposed method with conventional methods, for example, the one with ACS and GA [14].

References

[1] S.M. Sait and H. Youssef, Iterative Computer Algorithms with Applications in Engineering, IEEE Computer Society Press, 1999.

[2] C.R.R.B. MPhil, Modern Heuristic Techniques for Combinatorial Problems, Halsted Press, 1993.

[3] A. Abraham, C. Grosan, and V. Ramos, (eds.), Swarm Intelligence in Data Mining, Springer-Verlag New York, 2006.

[4] S. Iordache, "Consultant-guided search: A new metaheuristic for combinatorial optimization problems," Proc. 12th Annual Conference on Genetic and Evolutionary Computation, pp.225–232, 2010.

[5] P. Pop and S. Iordache, "A hybrid heuristic approach for solving the generalized traveling salesman problem," LANZI, Pier L.(HRSG.): GECCO, pp.481–488, 2011.

[6] S. Iordache, "Consultant-guided search algorithms for the quadratic assignment problem," Hybrid Metaheuristics, pp.148–159, 2010.

[7] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., vol.1, no.1, pp.53–66, 1997.

[8] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," Proc. 1999 Congress on Evolutionary Computation, vol.2, pp.1470–1477, 1999.

[9] T. Stützle and H. Hoos, "Max-min ant system," Future Generation Computer Systems, vol.16, no.8, pp.889–914, 2000.

[10] O. Roux and C. Fonlupt, "Ant programming: Or how to use ants for automatic programming," Proc. ANTS, pp.121–129, 2000.

[11] M. Birattari, G.D. Caro, and M. Dorigo, "Toward the formal foundation of ant programming," Ant Algorithms, Lecture Notes in Computer Science, vol.2463, pp.188–201, 2002.

[12] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo, "Parallel ant colony optimization for the traveling salesman problem," Ant Colony Optimization and Swarm Intelligence, pp.224–234, 2006.

[13] M. Hassan, M. Islam, and K. Murase, "A new local search based ant colony optimization algorithm for solving combinatorial optimization problems," IEICE Trans. Inf. & Syst., vol.E93-D, no.5, pp.1127–1136, May 2010.

[14] S. Chen and C. Chien, "Parallelized genetic ant colony systems for solving the traveling salesman problem," Expert Systems with Applications, vol.38, no.4, pp.3873–3883, 2011.

[15] S.M. Chen and C.Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," Expert Syst. Appl., vol.38, no.12, pp.14439–14450, Nov. 2011.

[16] L. Wang, R. Cai, L. Jing, and H. Zhang, "Object-guided ant colony optimization algorithm with enhanced memory for traveling salesman problem," Research Journal of Applied Sciences, vol.4, pp.3999–4006, 2012.

[17] G. Dong, W.W. Guo, and K. Tickle, "Solving the traveling salesman problem using cooperative genetic ant systems," Expert Syst. Appl., vol.39, no.5, pp.5006–5011, April 2012.

[18] R. Takahashi, "A hybrid method of genetic algorithms and ant colony optimization to solve the traveling salesman problem," International Conference on Machine Learning and Applications, 2009. ICMLA'09, pp.81–88, 2009.

[19] H. Duan and X. Yu, "Hybrid ant colony optimization using memetic algorithm for traveling salesman problem," IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL 2007, pp.92–95, 2007.

[20] X. Shi, Y. Lu, C. Zhou, H. Lee, W. Lin, and Y. Liang, "Hybrid evolutionary algorithms based on PSO and GA," The 2003 Congress on Evolutionary Computation, vol.4, pp.2393–2399, 2003.

[21] E. Duman, M. Uysal, and A.F. Alkaya, "Migrating birds optimization: A new meta-heuristic approach and its application to the quadratic assignment problem," Proc. 2011 International Conference on Applications of Evolutionary Computation, vol.1, pp.254–

263, 2011.
[22] X. Yang, "A new metaheuristic bat-inspired algorithm," Nature Inspired Cooperative Strategies for Optimization, pp.65–74, 2010.
[23] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: Ray optimization," Computers & Structures, vol.112, pp.283–294, 2012.
[24] "PC Cluster Consortium," http://www.pccluster.org/
[25] "TSPLIB," http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Hiroyuki Ebara received the B.S., M.S., and Ph.D. degrees in Communications Engineering from Osaka University, Osaka, Japan, in 1982, 1984, and 1987, respectively. In 1987 he became Assistant Professor at Osaka University. Since 1994 he has been with Kansai University, where he is currently an Associate Professor. His main research interests are in computational geometry, combinatorial optimization, and parallel computing. He is a member of IEEE (The Institute of Electrical and Electronics Engineers, Inc.), ACM (Association for Computing Machinery), SIAM (Society for Industrial and Applied Mathematics), IPSJ (Information Processing Society of Japan), and OR Society of Japan.

Yudai Hiranuma received the B.S., and M.S. degrees in the Faculty of Engineering Science from Kansai University, Osaka, Japan, in 2011 and 2013, respectively. Currently, he is working to NTT COMWARE CORPORATION. He is engaged in system development for corporate customers.

Koki Nakayama received the B.S. degree in the Faculty of Engineering Science from Kansai University, Osaka, Japan, in 2013. Currently, he is a M.S. student in the Algorithm Engineering Lab. at kansai University. His research interests are in finding an algorithm for solving combinatorial optimization problems.