Electronics and Communications in Japan, Vol. 96, No. 11, 2013
Translated from Denki Gakkai Ronbunshi, Vol. 132-C, No. 4, April 2012, pp. 584–591

Modification of ABC Algorithm for Adaptation to Time-Varying Functions

TAKESHI NISHIDA
Kyushu Institute of Technology, Japan

SUMMARY

Although the ABC (artificial bee colony) algorithm is an effective tool for finding solutions to static optimization problems, application to dynamical problems that include a time-varying element has not been assumed. A modification of the ABC algorithm for adaptation to time-varying functions is proposed. To adjust to the change in the function, a procedure for reevaluating the bees at each time is introduced. It is shown that the proposed modification does not influence the search performance of the conventional algorithm. The efficiency of the modified ABC algorithm is demonstrated and verified through numerical simulations. © 2013 Wiley Periodicals, Inc. Electron Comm Jpn, 96(11): 44–53, 2013; Published online in Wiley Online Library (wileyonlinelibrary.com). DOI 10.1002/ecj.11479

Key words: artificial bee colony algorithm; time-varying function.

1. Introduction

The artificial bee colony (ABC) algorithm, a probabilistic optimal search method, is an efficient method of solving nonlinear programming problems by simulating the swarm intelligence of bees [1]. Its performance in finding optimal solutions robustly and quickly for nonlinear problems, nondifferentiable problems, and multimodal problems has been verified in various investigations, and in particular there are reports [2] that it performs better in global solution search when the objective function is of high dimension. Improvements have also been made [3–5] in order to improve its performance in various optimization problems, and it has been used [6, 7] in a wide variety of application problems. However, the ABC algorithms have been proposed for the purpose of global solution search of time-invariant functions, and temporal change of the cost function due to observational noise and disturbances generated in various real implementations have not been taken into consideration in past researches. Thus, in this paper we propose a modification of the ABC algorithm to add adaptive performance with respect to time-varying functions. This modification is shown by numerical simulations not to affect the performance of global solution search with a time-invariant cost function in the ABC algorithm, while conferring adaptive performance for unimodal and multimodal time-varying functions. We also verify the relationship between various parameters and search performance.

This paper is organized as follows. Section 2 gives the processing procedure of the ABC algorithm. In Section 3, the modification to adapt to time-varying functions is proposed. In Section 4, the performance of the proposed modification is verified by numerical simulations. Section 5 gives a summary of the paper.

2. The Artificial Bee Colony Algorithm

2.1 Overview

In the ABC algorithm, global solution search problems for a cost function are modeled using three types of artificial bees that engage in food-gathering activities, namely, employed bees, onlooker bees, scout bees, and food sources, as basic components. A colony is one group of bees, and the purpose of the colony is to search for food sources with the highest value. Half of the colony consists of employed bees and scout bees, and the remainder of onlooker bees. Employed bees send information about food to onlooker bees at the same time as they fly close to remembered food sources. Onlooker bees select good food sources from among the sources based on information from employed bees and search mainly in the vicinity of those sources. When information about food sources is not updated in a set period, the employed bees in question abandon that food source, change to scout bees, and move to a new food source.

Based on these models, the goal of the ABC algorithm is to solve the optimization problem

$$\min_{\boldsymbol{x}} g(\boldsymbol{x}), \quad \text{subj. to} \quad \boldsymbol{x} \in \mathbb{R}^D \tag{1}$$

for the function g: $\mathbb{R}^D \to \mathbb{R}$. Here $\boldsymbol{x} \in \mathbb{S}$, and the cost function g is defined in the search space $\mathbb{S} \subseteq \mathbb{R}^D$. In the present research, $\mathbb{S}$ is defined in a D-dimensional hypercube $\{\boldsymbol{x} \in \mathbb{R}^D | l \le x_j \le u, \forall_j\}$ in $\mathbb{R}^D$. Here $j \in J = \{1, 2, \ldots, D\}$, and l and u are the lower and upper bounds of x_j, respectively.

2.2 Processing procedure

The procedure for solving an optimization problem in the ABC algorithm is shown below.[†]

Given Let the number of employed bees be n_e and the number of onlooker bees be n_o. Let the colony size be $N = n_e + n_o$. The value *limit* that regulates the change from employed bee to scout bee is set.[‡] The total number of iterations C_{max} is also set. l and u, which limit the range of the search space, are set.

Step 0

(1) Let the number of iterations be $c = 1$. The locations of the food sources and the employed bees are represented by $\boldsymbol{x}_i \in \mathbb{R}^D$.[§] Here $i \in I = \{1, 2, \ldots, n_e\}$ is the number of the employed bee. The number of successive times that the employed bees are not altered is given by s_i, and $s_i = 0$ $(\forall i)$ is set initially. In addition, the employed bees are distributed in $\mathbb{S}$ by means of uniform random numbers.

(2) Calculate the fitness in the initial arrangement:

$$f_i := \begin{cases} \dfrac{1}{1 + g(\boldsymbol{x}_i)} & \text{if } g(\boldsymbol{x}_i) \ge 0 \\ 1 + |g(\boldsymbol{x}_i)| & \text{otherwise} \end{cases} \tag{2}$$

Here the sign ":=" denotes substitution. The best solution in the initial arrangement and its function value are stored as follows:

$$\boldsymbol{x}_{best} := \boldsymbol{x}_{i_b} \tag{3}$$

$$f_{best} := f_{i_b} \tag{4}$$

where $i_b = \arg\max_i f_i$.

Step 1

This step is equivalent to search by the employed bees.

(1) A new food source candidate $\boldsymbol{v}_i \in \mathbb{R}^D (i \in I)$ is generated:

$$v_{ij} := \begin{cases} x_{ih} + \phi (x_{ih} - x_{mh}) & \text{if } j = h \\ x_{ij} & \text{otherwise} \end{cases} \tag{5}$$

Here v_{ij} and x_{ij} are the j-th elements in $\boldsymbol{v}_i$ and $\boldsymbol{x}_i$, respectively; $m \in I$ and $h \in J$ are values selected at random from I and J, and $\phi \in [-1, 1]$ is a uniform random number.

(2) The fitness $\boldsymbol{v}_i$ is found as follows:

$$\hat{f}_i := \begin{cases} \dfrac{1}{1 + g(\boldsymbol{v}_i)} & \text{if } g(\boldsymbol{v}_i) \ge 0 \\ 1 + |g(\boldsymbol{v}_i)| & \text{otherwise} \end{cases} \tag{6}$$

(3) Based on the fitness, the information held by the employed bees is updated as follows:

$$\boldsymbol{x}_i := \begin{cases} \boldsymbol{v}_i & \text{if } \hat{f}_i > f_i \\ \boldsymbol{x}_i & \text{otherwise} \end{cases} \tag{7}$$

$$f_i := \begin{cases} \hat{f}_i & \text{if } \hat{f}_i > f_i \\ f_i & \text{otherwise} \end{cases} \tag{8}$$

$$s_i := \begin{cases} 0 & \text{if } \hat{f}_i \ge f_i \\ s_i + 1 & \text{otherwise} \end{cases} \tag{9}$$

Step 2

This step is equivalent to search by the scout bees.

(1) Based on the fitness, the relative probability is calculated:

$$p_i = f_i \Big/ \sum_{n=1}^{n_e} f_n \tag{10}$$

(2) Using roulette selection based on the relative probability, the number of an employed bee is selected from J, and Step 1 is applied to the employed bee with that number. When this procedure has been repeated n_o times, the process moves to Step 3.

Step 3

If $f_{i_b} > f_{best}$, then the solution and the function value are updated:

$$\boldsymbol{x}_{best} := \boldsymbol{x}_{i_b} \tag{11}$$

$$f_{best} := f_{i_b} \tag{12}$$

where $i_b = \arg\max_i f_i$.

Step 4

This step is equivalent to search by the scout bees. The employed bees that have not been updated for a specified number of times, that is, the employed bees with number

$$m \in I_m = \{m | s_m \ge limit, m \in I\} \tag{13}$$

are redistributed within $\mathbb{S}$ by means of uniform random numbers, $s_m = 0$ is set, and the fitness f_m is calculated and stored.

[†]The procedure implemented here is based on C language source code made public on a web site [8] created by the person who proposed the ABC algorithm.

[‡]*limit* = $0.1 \cdot D \cdot N$ is set [2].

[§]The employed bees and the food sources are specified separately in conceptual terms, but in the processing procedure, they are completely identical.

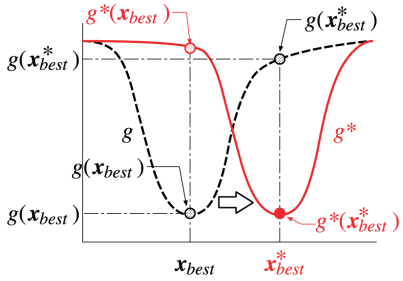


Fig. 1. Conceptual diagram of changes in the global minimum in a time-varying function. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Step 5

If $c = C_{max}$, the algorithm ends. If not, $c := c + 1$ is set, and the algorithm returns to Step 1.

2.3 Problems

Let us consider, for instance, a case in which the cost function is updated from $g(\cdot)$ to $g^*(\cdot)$ at a given time, as can be seen in Fig. 1. Here the global solutions for $g(\cdot)$ and $g^*(\cdot)$ are $\boldsymbol{x}_{best}$ and $\boldsymbol{x}^*_{best}$. The figure shows that if an evaluation of the solution using $g^*(\cdot)$ is possible, then evidently $g^*(\boldsymbol{x}^*_{best}) < g^*(\boldsymbol{x}_{best})$, and thus a search for a global solution $\boldsymbol{x}^*_{best}$ is possible. However, when continuing a search using $g(\cdot)$, $\boldsymbol{x}_{best}$ is not updated unless $g(\boldsymbol{x}^*_{best}) < g(\boldsymbol{x}_{best})$ or $f^*_{best} > f_{best}$ based on the procedure in Step 3, and thus the solution may converge to an unsuitable location.

As can be seen in this example, in the ABC algorithm, a time-varying cost function $g(\cdot)$ is not assumed, and thus a search adapted to environmental changes such as shifts in the optimal solution or variations of the cost function cannot be performed.

3. Modification of the ABC Algorithm

In order to create an ABC algorithm for global solution search of a time-varying function, part of the conventional ABC algorithm must be modified as shown below. First, the following procedure is inserted between items (2) and (3) in Step 1.

Step 1 (2-2)

The fitness of $\boldsymbol{x}_i$ is calculated as follows:

$$f_i = \begin{cases} \dfrac{1}{1+g(\boldsymbol{x}_i)} & \text{if } g(\boldsymbol{x}_i) \geq 0 \\ 1+|g(\boldsymbol{x}_i)| & \text{otherwise} \end{cases}$$

By this procedure, the changes in the fitness due to variations in the cost function can be reflected in the updating of the employed bees in Step 1(3). When the cost function is time-invariant, there are no changes to the value of f_i, and thus the use of this procedure does not affect the results of search for an optimal solution of the time-invariant cost function.

Next, Step 3 is modified as follows:

Step 3*

The search solution and its fitness are updated:

$$\boldsymbol{x}_{best} := \boldsymbol{x}_{i_b} \tag{14}$$

$$f_{best} := f_{i_b} \tag{15}$$

where $i_b = \arg\max_i f_i$.

In this procedure, the condition $f_{i_b} > f_{best}$ is removed from Step 3. Thus, the peak fitness is found for each time increment and not for the entire time period, and changes in the fitness accompanying changes in the cost function can be handled. Even if only this procedure is used, without Step 1(2-2) described above, f_{best} and $\boldsymbol{x}_{best}$ may be able to adapt to movement of the global solution: (1) a large region of solutions is found within a set time after the start of the search and more employed bees gather around it; (2) the global solution moves due to changes in the cost function; (3) for the reasons described in Section 2.3, the employed bees located in the vicinity of the past global solution are updated slowly because a better solution cannot be found, and the value of s_i increases; (4) if the value of s_i rises to some extent, then the employed bees change to scout bees in accordance with the procedure in Step 4 and are redistributed at random in the solution space; (5) all of the employed bees are redistributed and all of the values of f_i are reinitialized; (6) the peak value f_{best} is selected from among the newly calculated f_i, and the search for the moved global solution proceeds again. However, in order to restart the search in the flow described above, it is necessary to wait until all of the employed bees have been reinitialized in Step 4, and thus the rate of adaptation is very slow. The adaptation performance when only Step 3* is used is verified through simulations to be described later.

Thus, the ABC algorithm with the two procedures described above added for the purpose of optimal solution search that is adaptive to a time-varying cost function has the following features compared to the conventional ABC algorithm. (1) The fitnesses f_i and f_{best} corresponding to the changes in the global solution for the cost function are reevaluated successively, and thus they do not increase monotonically. (2) There is no new increase in the number of design parameters. (3) The modification is simple and produces little increase in the computational cost.

4. Numerical Simulations

The performance of the conventional ABC algorithm and the proposed algorithm is now compared.

Table 1. Numerical benchmark functions

function	ranges	minimum value	name
$g_1(\boldsymbol{x}) = \sum_{n=1}^{D-1} \left[100(x_n^2 - x_{n+1})^2 + (x_n - 1)^2\right]$	$-100 \le x_n \le 100$	$g_1(\mathbf{1}, 0, 0) = 0$	Rosenbrock function
$g_2(\boldsymbol{x}) = \sum_{n=1}^{D} x_n^2$	$-100 \le x_n \le 100$	$g_2(\mathbf{0}) = 0$	Sphere function
$g_3(\boldsymbol{x}) = \sum_{n=1}^{D} \left(x_n^2 - 10\cos(2\pi x_n) + 10\right)$	$-5.12 \le x_n \le 5.12$	$g_3(\mathbf{0}) = 0$	Rastrigin function
$g_4(\boldsymbol{x}) = \sum_{n=1}^{D} \frac{x_n^2}{4000} - \Pi_{n=1}^{D} \cos\left(\frac{x_n}{\sqrt{n}}\right) + 1$	$-600 \le x_n \le 600$	$g_4(\mathbf{0}) = 0$	Griewank function

4.1 Search performance for a time-invariant function

The four typical benchmark functions used in Ref. 2 and elsewhere, namely, the Rosenbrock function, Sphere function, Rastrigin function, and Griewank function (see Table 1 and the Appendix) are used, and a simulation to find a global solution by the conventional and modified algorithms is performed. The colony size is set to 50, 100, and 200 and the number of dimensions in the cost function to 50, 100, and 150, and the simulation is performed on all of the combinations of these conditions. Movement of the function value $g_p(\boldsymbol{x}_{best})$ $(p = 1, \ldots, 4)$ for the sought solution is verified. A system with the same pseudorandom numbers is used in all of the simulations. The results are shown in Fig. 2. In all simulations using any of the functions, the trends of the search results for the conventional ABC algorithm and the modified algorithm coincided completely. These results confirm that the proposed modification of the algorithm has no effect on global solution search performance for a time-invariant function and no effect on the results.

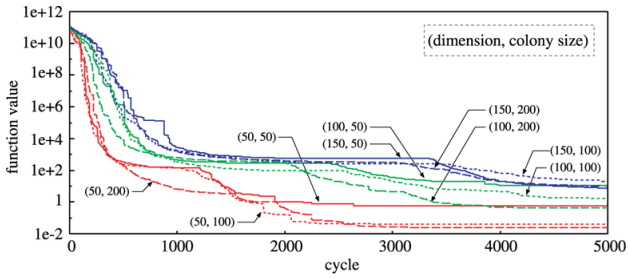
4.2 Adaptation performance for time-varying functions

4.2.1 Unimodal time-varying function

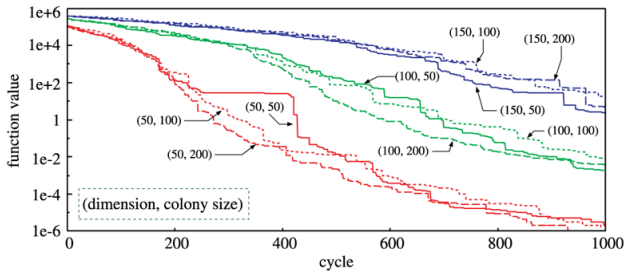
Let us now consider a global solution search for the following unimodal time-varying function:

$$g_5(x_1, x_2, k) = 1 - \exp\left[-\frac{(x_1 - 250 - 125\sin\alpha k)^2}{2 \cdot 40^2} - \frac{(x_2 - 250 + 125\cos\alpha k)^2}{2 \cdot 40^2}\right]$$

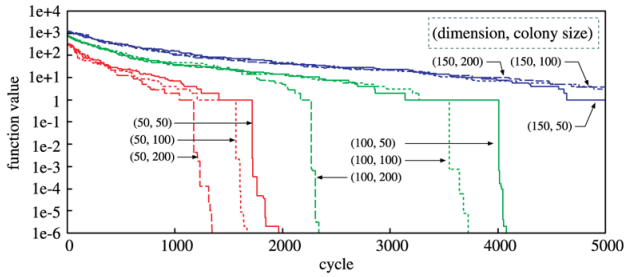
Figure 3 shows the shape of this function. This time-varying function simulates a visual tracking problem. Specifically, the distribution of the target features represented by a Gaussian function moves continuously in the figure, and the problem of tracking this movement is assumed. The function value for a global solution is 0, and its position shifts along a circle of radius 125 centered at $(x_1, x_2) = (250,$



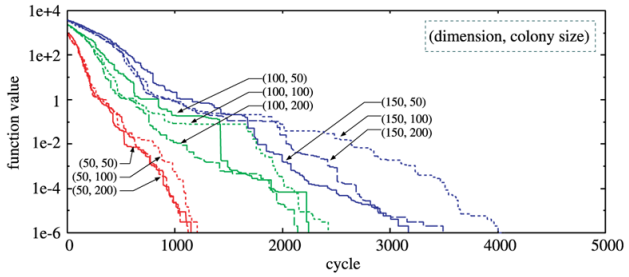
(a) Rosenbrock function.



(b) Sphere function.



(c) Rastrigin function.



(d) Griewank function.

Fig. 2. Evolution of the function value. The search results of the original and modified algorithm have completely coincided. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

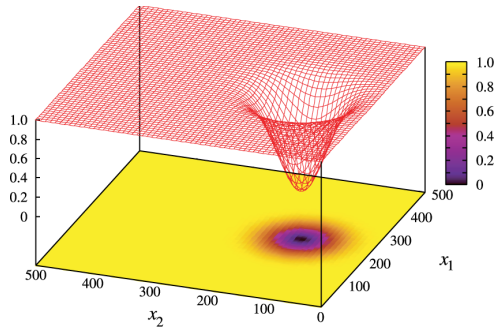
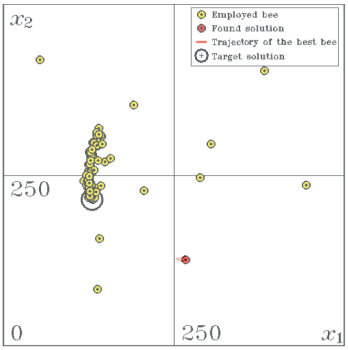


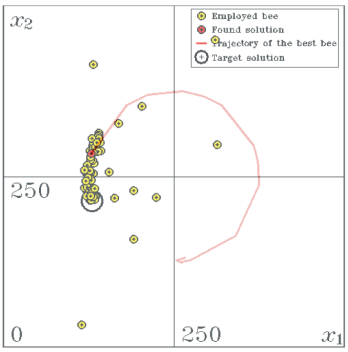
Fig. 3. Time-varying function $g_5(x_1, x_2, k)$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

250), then returns to its original position in $(2\pi/\alpha)$ steps. Here k represents the discrete time increment, and one step of the ABC algorithm is executed during the time in which this value is incremented by 1. The size of the change in each step increases with the value of α. The goal of each algorithm in this simulation is to continue to find $\boldsymbol{x}_{best}$, the position of the global solution for $g_5(x_1, x_2, x_3)$ that continues to move at a set rate. Here we use $n_e = n_o = 100$, with the same proportion of employed bees and onlooker bees as in Refs. 1 and 2. The parameter controlling the changes in the cost function is set to $\alpha = 0.01$.

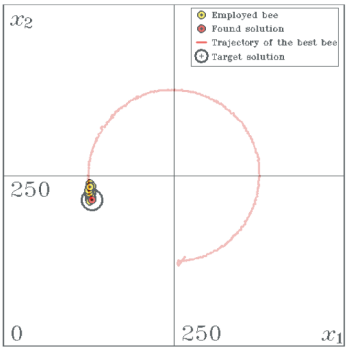
The three algorithms—the conventional ABC algorithm, the incompletely modified algorithm with only Step 3* utilized, and the proposed modified algorithm—are used for the time-varying cost function described above. Figures 4(a) to 4(c) show respectively the distribution of the employed bees at the instant at which $c = 500$, obtained by each algorithm, and the location of the solution that is sought as well as the trajectories of $\boldsymbol{x}_{best}$ toward it. Figure 4(d) shows the time evolution of the Euclidean distance between the global solution and $\boldsymbol{x}_{best}$. First, Figs. 4(a) and 4(d) show that updating of the sought solution $\boldsymbol{x}_{best}$ ceases because the value of f_{best} converges immediately after the start of the search in the conventional ABC algorithm. Next, Figs. 4(b) and 4(d) show that when using the incompletely modified algorithm, tracking of the solution is intermittent when the global solution moves, and in addition the trajectory of movement of the sought solution is distorted. This occurs because adaptive search is performed with reinitialization of the various parameters and creation of scout bees, but proceeds slowly. On the other hand, Figs. 4(c) and 4(d) show that there is little distortion of the trajectory of $\boldsymbol{x}_{best}$ when using the modified algorithm, and that tracking of the global solution when the cost function changes is performed very accurately. These results confirm that the modifications proposed in the present research allowed



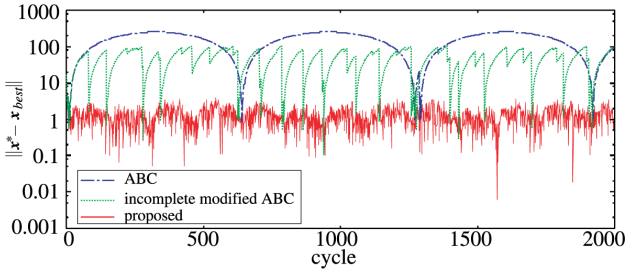
(a) Distribution of bees by original ABC at $c = 500$.



(b) Distribution of bees by incomplete modified ABC at $c = 500$.



(c) Distribution of bees by proposed ABC at $c = 500$.



(d) Time evolution of Euclidean distance between the global solution and the found solution.

Fig. 4. Results of tracking simulation of the unimodal time-varying function $g_5(x_1, x_2, k)$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

adaptive tracking of global solutions of a unimodal time-varying function that varies continuously.

4.2.2 Multimodal time-varying function

Let us next consider a global solution search for the multimodal time-varying function shown below:

$$g_6(x_1, x_2, k) = 1 - \left[\frac{\sin \beta k + 1}{2} \exp\left\{ -\frac{1}{2}\left(\frac{(x_1 - 125)^2}{40^2} + \frac{(x_2 - 375)^2}{40^2} \right) \right\} + \frac{\sin(-\beta k) + 1}{2} \exp\left\{ -\frac{1}{2}\left(\frac{(x_1 - 375)^2}{40^2} + \frac{(x_2 - 125)^2}{40^2} \right) \right\} \right]$$

Figure 5 shows a representation of this function. This time-varying function simulates a visual tracking problem, as the function described above did. Specifically, the distribution of the multiple target features represented by a Gaussian function repeatedly appears and disappears in the image, and moves discontinuously. The problem is to track them. The function value of the global solutions is assumed to be between 0 and 0.5, and the position is switched between (125, 375) and (375, 125) in a set period. The objective of the algorithm in this simulation is to continuously capture as $\boldsymbol{x}_{best}$ the position of the global solution for $g_6(x_1, x_2, k)$, which moves in a set period. Furthermore, the size of the change in the cost function in each step increases as the value of β is set higher. The numbers of employed bees and onlooker bees are the same as in the simulation described above, with $n_e = n_o = 100$. The parameter that controls the changes in the cost function is set to β = 0.05. In this simulation, movements of the global solution occurred 31 times during 2000 runs.

The three algorithms—the conventional ABC algorithm, the incompletely modified algorithm with only Step 3* added, and the proposed modified algorithm—are each applied to the time-varying function described above. Figures 6(a) to 6(c) show the distribution of the employed bees at the instant when $c = 500$ as obtained by each algorithm,

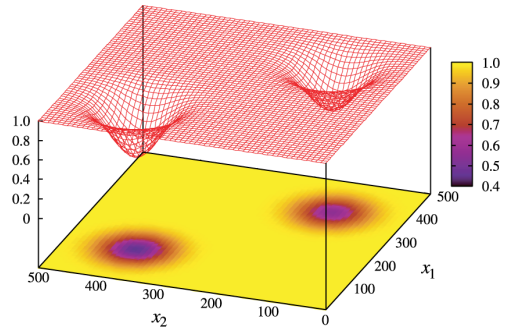
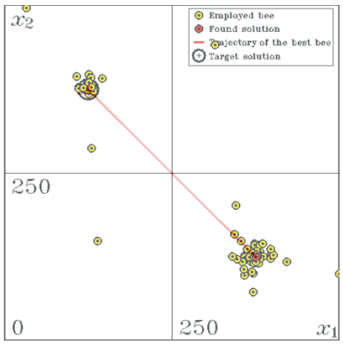
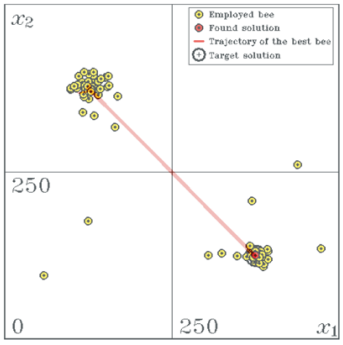


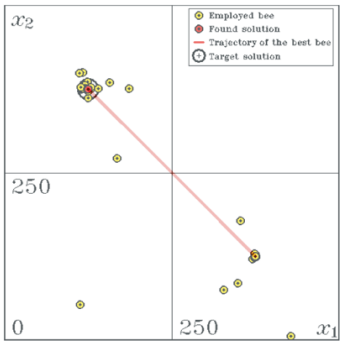
Fig. 5. Time-varying function $g_6(x_1, x_2, k)$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]



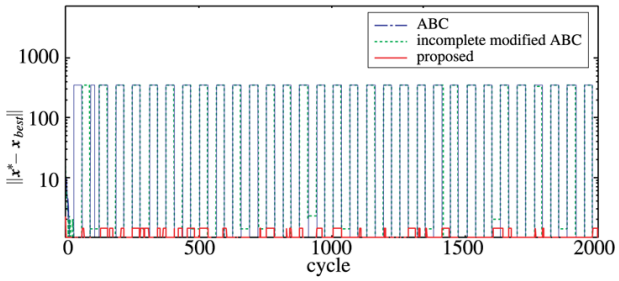
(a) Distribution of bees by original ABC at $c = 500$.



(b) Distribution of bees by incomplete modified ABC at $c = 500$.



(c) Distribution of bees by proposed ABC at $c = 500$.



(d) Time evolution of Euclidean distance between the global solution and the found solution.

Fig. 6. Results of tracking simulation of the multimodal time-varying function $g_6(x_1, x_2, k)$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

the position of each solution found, and the trajectory of $\boldsymbol{x}_{best}$ up to that point. Figure 6(d) shows the time evolution of the Euclidean distance between the global solution and $\boldsymbol{x}_{best}$. First, Figs. 6(a) and 6(d) show that in the conventional algorithm, $\boldsymbol{x}_{best}$ moved only once, from (125, 375) to (375, 125), after which updating ceased, and it was unable to adapt to the changes in the position of the global solution. Thus, the solution discovered by the conventional ABC algorithm stops at (375, 125), and the Euclidean distance to the global solution is $250\sqrt{2}$ when the global solution shifts to (375, 125) or when it shifts to (125, 375). The algorithm moves back and forth between these two values. Figures 6(b) and 6(d) show that the partially modified algorithm was able to perform an adaptive search. This is a result of the slow speed of the adaptive search due to the procedure in Step 3*. On the other hand, Fig. 6(c) shows that under the fully modified algorithm, $\boldsymbol{x}_{best}$ could adapt to changes in the position of the global solution. These results confirm that the modifications of the algorithm proposed in the present research allowed adaptive tracking and search for a global solution to a multimodal time-varying function that varies discontinuously.

4.3 Colony size and adaptation performance

We next verify the effects of colony size on the search performance of the modified algorithm. Using the time-varying functions $g_5(x_1, x_2, k)$ and $g_6(x_1, x_2, k)$ described above, Fig. 7 shows the results of simulations with the colony size set to 10, 50, and 100. We used $\alpha = \beta = 0.01$ and the value of n_e was half the colony size. These results show that when the colony size N was only 10, the moving optimal solution could not be tracked well. It can also be seen that the adaptation performance can be improved by setting the colony size larger.

4.4 Movement speed and adaptation performance of the cost function

Simulations with the multimodal time-varying function used in Section 4.2.2 and the unimodal time-varying function used in Section 4.2.1 were performed to verify the relationship between the speed of movement of the cost function and the adaptation performance of the modified algorithm. The values of α and β, which adjust the rate of variation of the time-varying functions $g_5(x_1, x_2, k)$ and $g_6(x_1, x_2, k)$ described above, were set to 0.01 and 0.05 and varied by 0.1, and another simulation was performed. The colony size was set to 200, and $n_e = 100$ was used. Figure 8 shows the time evolution of the Euclidean distance of the global solution and solution found in the simulation. Figure 9 also shows the trajectory of the solution found when $\alpha = 0.05, 0.1$ for $g_5(x_1, x_2, k)$ and the arrangement of the employed bees at times $k = 100$ and $k = 50$. These results show

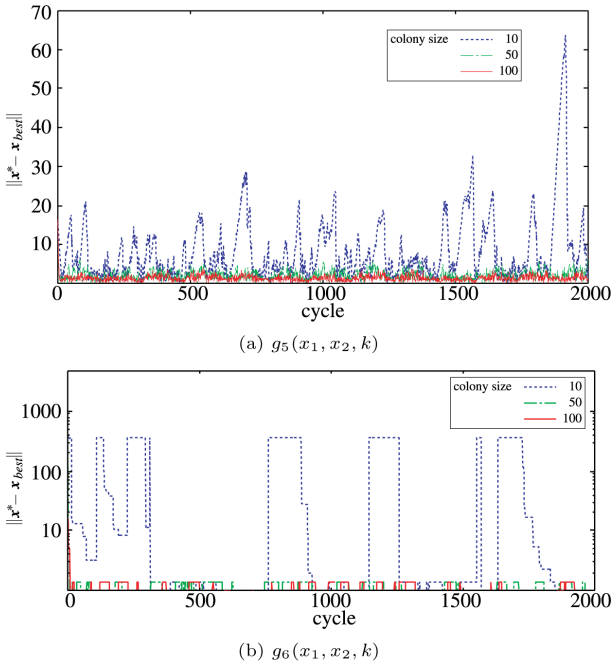


Fig. 7. Time evolution of the mean squared error versus the colony size. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

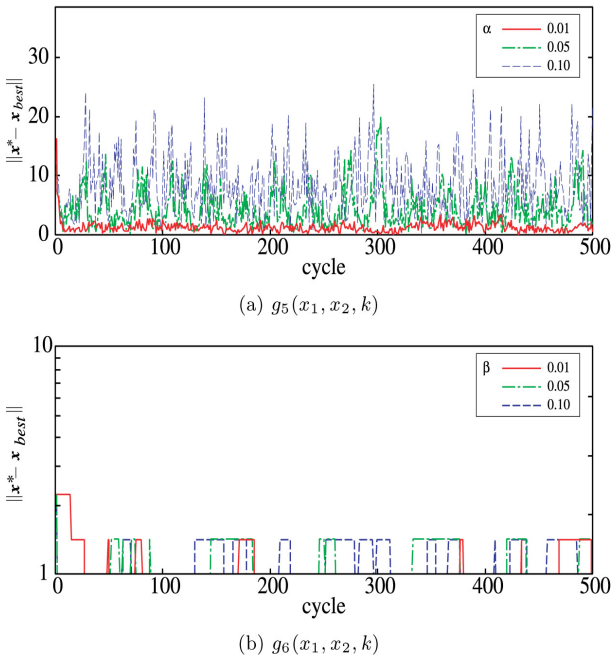
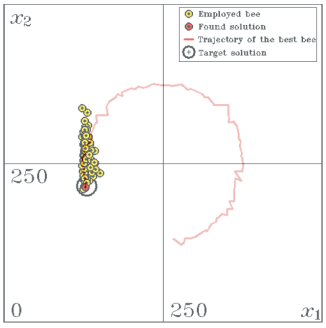
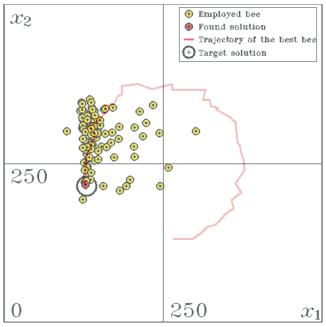


Fig. 8. Time evolution of the mean squared error versus the speed of change of the function. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]



(a) $\alpha = 0.05$



(b) $\alpha = 0.1$

Fig. 9. Trajectory of bees at (a) $k = 100$ and (b) $k = 50$ for tracking of $g_5(x_1, x_2, k)$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

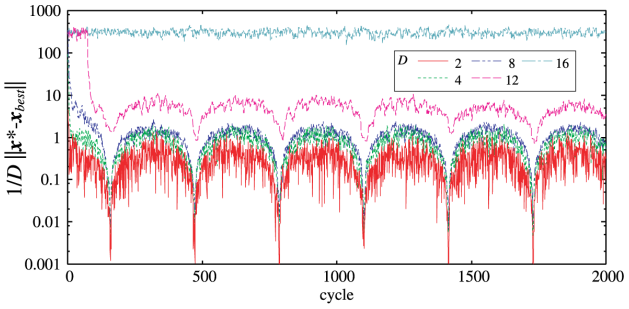
that the movement of the global solution as a result of changes in the cost function can be adapted to at any speed, and that as the rate of change of the cost function increases, the Euclidean distance between the global solution and the found solution tends to increase. Figure 9 also shows that there is more distortion of the trajectory of the found solution relative to the shifting trajectory of the global solution as the rate of change in the cost function increases.

4.5 Number of dimensions and adaptation performance

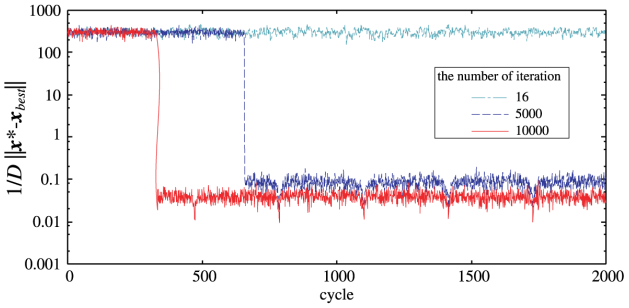
Let us next consider a global solution search of a unimodal time-varying cost function in D dimensions, as follows:

$$g_7(\boldsymbol{x}, k) = 1 - \exp\left\{-\frac{1}{2}\sum_{n=1}^{D}\left(\frac{x_n - 125\sin\alpha k}{40}\right)^2\right\} \tag{16}$$

Here x_n represents the n-th element in $\boldsymbol{x} \in \mathbb{R}^D$, and the search region is inside the hypercube extended so that $-500 \le x_n \le 500$. This function has a minimum value of 0 at $x_n = \sin \alpha k$ $(\forall n)$. Here we set $\alpha = 0.01$ and $n_e = n_o = 100$ and a minimal solution search using the proposed algorithm was performed. As can be seen from the results in Section 4.1, as the number of dimensions of the function in question increased, more iterations were needed for convergence of the solution search. Thus, as the discrete time k of the cost function advanced by one step, the iteration steps c in the algorithm were increased by only the same number as the number of dimensions, that is, the search was repeated D times. The tracking accuracy was also evaluated using the value $1/D\|\boldsymbol{x}^* - \boldsymbol{x}_{best}\|$ found by dividing the Euclidean distance between the found solution and the optimal solution by the number of dimensions. Figure 10(a) shows the results obtained when the number of dimensions was varied over $D = 2$, 4, 8, and 16. These results show a drop in the search accuracy and the convergence speed of the solution with an increase in the number of dimensions. In particular, when the number of dimensions is $D = 16$, tracking of the global solution fails. Figure 10(b) shows the search results for a solution when the number of iterations in the proposed algorithm was varied over 16, 5000, and 10,000 as the discrete time k in the function advanced by one step, for a number of dimensions $D = 16$. These results show that the global solution can be found and tracked by significantly increasing the number of times the algorithm is used as the number of dimensions increases, and that tracking accuracy is further improved.



(a) The number of dimensions vs. search performance.



(b) The number of iteration vs. search performance ($D = 16$).

Fig. 10. Time evolution of the distance between $\boldsymbol{x}_{best}$ and the optimal solution $\boldsymbol{x}^*$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

5. Conclusions

This paper shows for the first time that the ABC algorithm cannot adapt to the time evolution of a cost function because it has an algorithmic structure in which the fitnesses f_i ($\forall i$) of the employed bees and the fitnesses f_{best} of the found solution increase monotonically. A modified method was proposed to resolve these problems, and its features were verified by numerical simulations. The proposed modified algorithm has a structure that actively takes into consideration the time evolution of the cost function by reevaluating at each time increment the values of f_i and f_{best} based on changes in the cost function. There is also no need to introduce new parameters, and no significant increase in the computational cost or in the complexity of the algorithm structure results.

Among the many evolutionary algorithms (EAs) proposed up to the present time, the features of the ABC algorithm are superior for dealing with the finding of optimal solutions in higher dimensions [2]. In the present research, it was first shown by simulations that the proposed algorithm has a performance similar to that of the conventional ABC algorithm in search problems with a static cost function in higher dimensions, and the superiority of the proposed method was then confirmed. Next, using unimodal and multimodal cost functions of higher dimension, it was shown that the proposed method provides search performance adapted to these issues. The practicability of its use for time-varying functions of higher dimension was also verified.

Most EAs cannot be used for optimal solution searches with a dynamic cost function because they do not assume changes in the search environment. In recent years, improvements to various EAs to allow adaptation to a dynamic environment have been proposed [10, 11]. However, the dynamic problems that these revised EAs assume are of various kinds, and remain to be systematized. Therefore, a performance comparison between the proposed method and other EAs that assumes application to dynamic problems is a topic for the future. The use of the present modifications in a revised ABC algorithm, as has been frequently proposed, together with performance verification should be considered. Application of the present method to a specific or real time-varying system should also be considered.

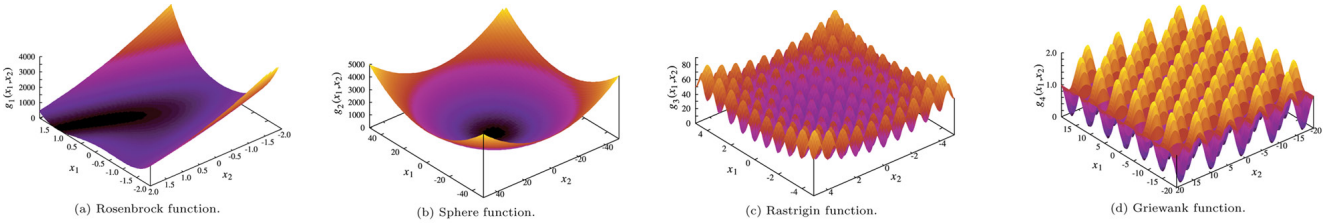
REFERENCES

1. Karaboga D. An idea based on honeybee swarm for numerical optimization. Tech Rep TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
2. Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 2007;8:687–697.
3. Kang F, Li J, Ma Z. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. Inf Sci 2011;181:3508–3531.
4. Gao W, Liu S. Improved artificial bee colony algorithm for global optimization. Inf Process Lett 2011;111:871–882.
5. Gao W, Liu S. A modified artificial bee colony algorithm. Comput Oper Res Lett 2012;39:687–697.
6. Horng M. Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation. Comput Oper Res Expert Syst Appl 2011;38:13785–13791.
7. Szeto WY, Wu Y, Ho SC. An artificial bee colony algorithm for the capacitated vehicle routing problem. Eur J Oper Res 2011;215:126–135.
8. Artificial bee colony (ABC) algorithm home page: http://mf.erciyes.edu.tr/abc/.
9. Doi M, Kamiya Y, Mori Y. Evolutionary algorithms: Genetic programming, particle swarm optimization, memetic algorithm, Cma-Es, Gaussian adaptation, harmony search, evolutionary algorithm mutation testing, neuroevolution, differential evolution, learning classifier system, evolution strategy. Books LLS, Memphis; 2010.
10. Kominami M, Hamagami T. A new genetic algorithm with diploid chromosomes by using probability decoding for adaptation to various environments. IEEJ Trans EIS 2008;128:381–387. (in Japanese)
11. Nishida T, Sakamoto T. Adaptive PSO for online identification of time-varying systems. IEEJ Trans EIS 2011;131:1642–1649. (in Japanese)

APPENDIX

Time-Invariant Test Function

Representations in two dimensions of the test functions used in the evaluation of the global solution search performance of the proposed method for time-invariant functions are shown in Fig. A.1.



(a) Rosenbrock function. (b) Sphere function. (c) Rastrigin function. (d) Griewank function.

Fig. A.1. Test functions in two dimensions. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

AUTHOR

Takeshi Nishida (member) received a bachelor's degree from the Department of Design and Manufacturing of Kyushu Institute of Technology in 1998, completed the latter half of the doctoral program at the Graduate School of Engineering in 2002, and became a lecturer in the School of Machine Intelligence. He was appointed an assistant professor in 2007 and received his D.Eng. degree. He is engaged in research on outdoor mobile robots. He is a member of the Robotics Society of Japan, SICE, the Japanese Neural Network Society, and IEICE.