Artificial Bee Colony Algorithm Based on Adaptive Local Information Sharing Meets Multiple Dynamic Environments

Ryo Takano *, Hiroyuki Sato *, and Keiki Takadama *

Abstract: This paper focuses on the artificial bee colony (ABC) algorithm as one of swarm optimization methods and proposes ABC-alis (ABC algorithm based on adaptive local information sharing) by improving the ABC algorithm for dynamic optimization problems (DOPs). ABC-alis is applied to various types of dynamic changes embedded in DOPs to verify its tracking ability in such dynamic environments. Concretely, the following five types of dynamic changes and one of the high-dimensional problem are employed as the different environments: (A) a periodic change of evaluation values of local optima; (B) a random change of evaluation values of local optima; (C) a random change of local optima coordinates; (D) a combination of two kinds of random changes (B+C); (E) a random speed change of local optima in the environment (D); and (F) a high-dimensional problem in the environment (E). In these experiments, the following three methods are compared: ABC-alis as the proposed method, ABC-lis as our previous method of ABC-alis, and speciation-based particle swarm optimization (SPSO) as the conventional method. The experimental result revealed that the following implications: (1) ABC-alis and ABC-lis can capture the optimal solution more quickly and keep a better solution than SPSO in the various types of dynamic changes; (2) from environments C, D, and E, ABC-alis can adapt to the random change of local optima from the viewpoint of the evaluation value, coordinates, speed, and all of them; and (3) ABC-alis can maintain its performance even in the high-dimensional environment F.

Key Words: ABC algorithm, swarm optimization, dynamic optimization problems, search range, self-adaptation.

1. Introduction

Research handling dynamic optimization problems increases in the last decade because a lot of real-world problems are dealt with as dynamic optimization problems (DOPs) [1],[2]. Examples include dynamic vehicle routing [3], mobile ad hoc networks [4], and dynamic economic dispatch [5], and so on. As features of DOPs, the objective function changes over time and the optimal solution changes by the lapse of time, which makes the current good solutions including the best one change to the worse solutions. In such DOPs, it is generally difficult to solve them because we have to tackle not only predictable change such as linear one but also unpredictable change such as random.

To address DOPs, the artificial bee colony (ABC) algorithm (i.e., a bee corresponds to an individual) [6] as the swarm-based algorithm is one of candidates because the ABC algorithm was extended for DOPs by Nishida [7] and succeeded to track the dynamic change of solutions (i.e., local optima) by updating their evaluation values every iteration. When the number of local optima increases in the multimodal functions, however, Nishida’s modified ABC algorithm is hard to cope with the dynamic change, which results in decreasing the search performance [8]. To tackle this problem, our previous research proposed ABC-lis (ABC algorithm based on local information sharing) which was improved from the ABC algorithm to search globally while capturing local optima for tracking the dynamic change in the multimodal functions [9]. For this purpose, the search range for each individual in ABC-lis is limited in the interaction among individuals. Such the search range promotes individuals to capture their own local optima and to globally search even by change over time. What should be noted here is that it is necessary to appropriately determine the sharing range beforehand to track the change of solutions constantly and properly. However, it is difficult to determine the appropriate sharing range because prior knowledge on dynamic changes of optimal solutions cannot be obtained in most DOPs.

In order to solve this problem, this paper proposes ABC-alis (ABC algorithm based on adaptive local information sharing) which can change the local sharing range adaptively according to the location of individuals, and aims at investigating its effectiveness in the various dynamic changes. To verify the track ability of ABC-alis in dynamic changes, ABC-alis is compared with ABC-lis as our previous method and speciation-based particle swarm optimization (SPSO) [10] as the conventional method. As the various types of dynamic changes in DOPs, the following five types of dynamic changes and one high-dimensional problem are employed: (A) a periodic change of evaluation values of local optima; (B) a random change of evaluation values of local optima; (C) a random change of local optima coordinates; (D) a combination of two kinds of random changes (B+C); (E) a random speed change of local optima in the environment (D); and (F) a high-dimensional problem in the environment (E).

This paper is organized as follows. Section 2 explains the original ABC algorithm and the ABC algorithm modified for DOPs. Section 3 describes ABC-lis with discussing the limitation of information sharing among individuals to adapt to the
dynamic change, and Section 4 proposes ABC-alis which introduces the adaptive information sharing range mechanism. Section 5 explains the problem description and describes the five types of dynamic changes and one type of high-dimensional problem. Section 6 describes the experimental setting, and Section 7 gives the experimental results and discusses their results. Finally, Section 8 summaries the conclusion of this paper.

2. Artificial Bee Colony Algorithm

2.1 Original ABC Algorithm

In the ABC algorithm, bees change their search phases by playing one of the following roles of bees: employed bees, onlooker bees, and scout bees. The ABC algorithm regards the food sources of the bees as the solutions and discovers the most important food source as the optimal solution. The importance of the food sources is evaluated by the given objective function. As the brief algorithm, the employed bees start to search their food sources. The onlooker bees choose an important food source according to evaluations by the employed bees. If the employed bee cannot find the food source around the current location, the employed bee becomes the scout bee. The scout bees randomly choose their new food sources in search space. As the parameters of the ABC algorithm, the size of population is represented as \( N_s \). Each food source is represented as \( x_i \) \((i=1,2,\ldots,N_s)\), and the evaluation value of each food source is represented as \( f(\mathbf{x}_i) \). The detailed flow of the ABC algorithm is described as follows:

**Step0: Initialization**

At the beginning, the number of iterations \( C \) is set as zero, and all bees start to play their roles as the employed bees with setting their initial locations randomly in search space. The bees evaluate their own food sources (i.e., solutions) and the highest evaluation value among those solutions is memorized as \( x_{\text{best}} \) with setting the number of trials for each food source is 0 \((\text{trial}_i = 0)\).

**Step1: Employed bee phase**

In the employed bee phase, the \( i^{th} \) employed bee selects the candidate of the new solution \( v_i \) which is located around the current solution \( x_i \). The candidate of new solution is calculated from the current solution according to Eq. (1):

\[
v_{ij} = x_{ij} + \phi \cdot (x_{ij} - x_{ik}),
\]

where \( k \in 1,2,\ldots,N_s \) indicates the indexes of the employed bee, \( j \in 1,2,\ldots,D \) indicates the dimension, and \( D \) indicates the maximum number of dimensions in search space (e.g., \( D = 2 \) in the case of a two-dimensional problem). Note that the vectors \( v_i \) and \( x_i \) are respectively represented as \( v_i = (v_{i1}, v_{i2}, \ldots, v_{iD}) \) and \( x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \), where the scalar values \( v_{ij} \) and \( x_{ij} \) mean the \( j^{th} \) component of the new solution \( v_i \) and the current solution \( x_i \), both of which belong to the \( i^{th} \) employed bee, and the function \( \phi \) provides a random number between \(-1\) and 1. When calculating the candidate of a new solution according to Eq. (1), one employed bee (except for the \( i^{th} \) bee) and one dimension are randomly selected, i.e., \( k \) and \( j \) are randomly determined.

If the evaluation of the candidate of the new solution \( f(\mathbf{v}_i) \) is better than the current one (i.e., \( f(\mathbf{v}_i) > f(\mathbf{x}_i) \)), the current solution is replaced with the new candidate (i.e., \( x_{ij} = v_{ij} \)) and the number of trials is initialized (i.e., \( \text{trial}_i = 0 \)). On the other hand, if the evaluation of the candidate of the new solution is worse than or equal to the current one (i.e., \( f(\mathbf{v}_i) \leq f(\mathbf{x}_i) \)), the bee stays in the current place and the number of trials is increased by 1 (i.e., \( \text{trial}_i = \text{trial}_i + 1 \)).

This procedure is summarized as follows:

\[
x_i = \begin{cases} v_i & \text{if } f(\mathbf{v}_i) > f(\mathbf{x}_i), \\ x_i & \text{if otherwise}, \end{cases}
\]

\[
\text{trial}_i = \begin{cases} 0 & \text{if } f(\mathbf{v}_i) > f(\mathbf{x}_i), \\ \text{trial}_i + 1 & \text{if otherwise}. \end{cases}
\]

**Step2: Onlooker bee phase**

After the employed bee phase, all employed bees become the onlooker bees. The onlooker bees search the potential solutions found in the employed bee phase. Concretely, the \( i^{th} \) onlooker bee selects a solution from all of solutions found by the employed bees according to the roulette wheel selection with its probability \( p_i \), which is calculated by Eq. (4):

\[
p_i = \frac{f(\mathbf{x}_i)}{\sum_{n=1}^{N_s} f(\mathbf{x}_n)}.
\]

As the mechanism of the roulette wheel selection, the solutions with the high evaluation values are frequently chosen by the onlooker bees. The candidate of the new solution is calculated by the onlooker bees with the same way as the employed bee phase.

**Step3: Memorizing the best food source**

After the onlooker bee phase, the best food source is memorized. Concretely, if the highest evaluation value in the current iteration is better than the best evaluation in the past iterations \((f(\mathbf{x}_{\text{best}}) < f(\mathbf{x}_{\text{best}}))\), the solution \( \mathbf{x}_{\text{best}} \) is memorized as \( \mathbf{x}_{\text{best}} \). The solution \( \mathbf{x}_{\text{best}} \) is the global optimal solution found by the ABC algorithm.

**Step4: Scout bee phase**

If the number of trials for finding a better solution in the \( i^{th} \) onlooker bee reaches the maximum limit value, the \( i^{th} \) onlooker bee becomes the scout bee. The scout bees are randomly relocated in search space, and the number of trials is reset (i.e., \( \text{trial}_i = 0 \)). Note that the step 4 is skipped if the onlooker bee does not become the scout bee.

**Step5: Repeating Step1 to Step4**

After the above steps are all executed, this algorithm updates the number of iterations as \( C = C + 1 \), and returns to the step 1. Note that the cycle from the steps 1 to 4 is repeated \( N_{\text{mc}} \) times.

2.2 Modified ABC Algorithm for DOPs

Since the original ABC algorithm does not assume dynamic changes, it is not possible to track the solution changes in DOPs. To tackle this problem, Nishida proposed the modified ABC algorithm that can adapt to solution changes in dynamic environments without decreasing its search performance [7]. In the modified ABC algorithm, the steps 1 and 3 of the original ABC algorithm are modified as follows (note that the other steps of the modified ABC algorithm are the same as those of the original ABC algorithm):

**Step1’: Modified employed bee phase**

In the original ABC algorithm, the evaluation value of the candidate of the new solution \( f(\mathbf{v}_i) \) is calculated and compared with the evaluation value of the current solution \( f(\mathbf{x}_i) \). In the static environment, in particular, \( f(\mathbf{x}_i) \) is not necessary to be recalculated because its value does not change during search
process. However, in the dynamic environment, $f_{\text{fit}}(x_i)$ changes during search process. For this reason, the modified ABC algorithm recalculates $f_{\text{fit}}(x_i)$ every iteration, which makes it possible to track the dynamic changes.

**Step3**: Modified memorizing the best food source

In the dynamic environment, it is not necessary to hold the best solution through all iteration, because the best solution continuously change during search process. For this reason, the modified ABC algorithm chooses the best solution from only current iteration at the step 3. Concretely, the best solution at the current iteration $x_k$ is always stored as the best solution without comparing it with $x_{\text{best}}$.

3. ABC-lis

### 3.1 Local Information Sharing

As the problem of the modified ABC described in Section 2.2, all bees easily converge to some local optima as the number of local optima increases. This problem is caused because all bees tend to move to the local optima through their communication among them. This decreases the global search ability of the modified ABC in the dynamic environment. In order to solve this problem, our previous research proposed ABC-lis (ABC algorithm based on local information sharing) [9] that can track dynamic changes without convergence to a few local optima by limiting the range for communication among bees. Specifically, the communication range of the bees is predefined by the Euclidean distance, and the bees can communicate bees within this limited range (hereafter, this limited range is called as the sharing range). This local information sharing only allows the bees to share the information with only nearby bees and contribute to tracking dynamic solution changes.

### 3.2 Algorithm

For the limitation of the sharing range among the bees as mentioned above, the steps 1, 2, and 4 of the original ABC algorithm are modified as follows (note that the other step of ABC-lis is the same as those of the original ABC algorithm):

**Step1***: Employed bee phase with limited communication

In the employed bee phase in ABC-lis, each employed bee shares information with the bee within the sharing range, which is represented by the circle with the radius $d$ from the bee. Concretely, if some bees (represented as $X_i \not= X_j$) are located within the sharing range $d$ from bee $x_i$, the solution $v_i$ in the $j^{th}$ dimension which belongs to the $k^{th}$ employed bee (i.e., $x_{kj}$ in Eq. (1)) is randomly selected, and the candidate of the new solution $v_i$ in the $j^{th}$ dimension (i.e., $v_{ij}$) is calculated. On the other hand, if no bees are located within the sharing range $d$ from bee $x_i$, the candidate of the new solution is calculated by adding the current solution $x_i$ with the random value within the sharing range $d$ from the current solution. The candidate of the new solution in these two cases is calculated as shown in Eq. (5):

$$v_{ij} = \begin{cases} 
    x_{ij} + \varphi(x_{ij} - x_{kij}) & \text{if } \exists X_j, ||x_i - x_k|| < d, \\
    x_{ij} + \delta d & \text{otherwise,}
\end{cases}$$

(5)

where $X_i$ is the set of solutions within the distance $d$ around the bee $x_i$ and the inequality $||x_i - x_k|| < d$ means $\sqrt{(x_{1i} - x_{1k})^2 + (x_{2i} - x_{2k})^2 + \cdots + (x_{ni} - x_{nk})^2} < d$. From this equation, the candidate of the new solution $v_i$ is always generated within the sharing range $d$ from the bee $x_i$.

**Step2**: Onlooker bee phase with limited communication

In the original ABC algorithms, all solutions can be selected by all onlooker bees according to the roulette wheel selection with the probability $p_i$, calculated by Eq. (4). However, the onlooker bee in ABC-lis randomly searches around itself when no bees are located within the range of the distance $d$, while it searches around the solutions selected from those located within the range of the distance $d$ when some bees are located in the range. The solution selection probability $p_i$ of the onlooker bee $x_i$ is calculated by Eq. (6):

$$p_i = \begin{cases} 
    p_i & \text{if } \exists X_i, ||x_i - x_k|| < d, \\
    0 & \text{otherwise,}
\end{cases}$$

(6)

where $p_i$ is calculated by Eq. (4). In Eq. (6), the solution selection probability of the $i^{th}$ bee is set as $p_i$ when some bees are located within the range of the distance $d$, while it is set as 0 when no bees are located within the range. In other words, the onlooker bees can share information with only nearby bees which are located within the sharing range $d$, and select the solution from such nearby bees.

**Step4**: Scout bee phase

The scout bee phase is modified to control the number of bees in each local optimum by employing the local rank of the evaluation value and the new solution selection probability changed by the sigmoid function. From this improvement, each bee calculates its local rank of the evaluation value among the bees located within the sharing range $d$. The probability of changing to the scout bee (i.e., $p_{s\text{\_new}}$) for the bee $x_i$ is calculated by its local rank such as the following Eq. (7):

$$P_{s\text{\_new}} = \frac{1}{1 + e^{2(\text{localrank}_i - N_l)}},$$

(7)

where localrank$_i$ is the rank of $x_i$ among the bees located within the sharing range $d$ and $N_l$ is the parameter of the inflection point of the sigmoid function. For example, the shape of the sigmoid function ($N_l = 10$) is shown in Fig. 1, where the horizontal and vertical axes represent the local rank of the bee and the probability of changing to the scout bee, respectively. In particular, $N_l$ controls the number of bees in each local optimum. If localrank$_i$ is smaller than $N_l$, the bee $x_i$ mostly continues to search around the prior solution without changing to the scout bee. On the other hand, if localrank$_i$ is larger than $N_l$, the bee $x_i$ mostly searches other areas by changing to the scout bee. This mechanism keeps the number of bees around $N_l$ in each local optimum to enable the bees to always search in each local optimum.

4. ABC-alis

To adjust the local information sharing range, the ABC algorithm based on adaptive local information sharing (ABC-alis) is proposed to improve ABC-lis.

![Fig. 1 Probability of changing to the scout bee by sigmoid function.](image-url)
4.1 Adaptation Local Information Sharing

The sharing range $d$ should be set to capture as many local optima as possible in search space. When several local optima are located in the sharing range $d$, most bees gather only in a few local optima, and the other local optima are hard to be captured. From this reason, ABC-lis needs the mechanism of adjusting the sharing range in accordance with the number of local optima within the range $d$ to cope with various DOPs. For this purpose, the sharing range $d_i$ is provided to the bee $x_i$, and it is individually changed by the rule of the binary tree shown in Fig. 2. Note that the rule of this binary tree is applied after the scout bee phase.

To adjust the sharing range, this paper starts to define the cloudingbees as one or more other bees which gather in the range of the radius $\lambda$ of the bee $x_i$. The range of the radius $\lambda$ is called as the closerrange. Note that the close range $\lambda$ should be much smaller than the sharing range $d_i$ of the bee $x_i$, since the close range $\lambda$ judges whether one bee is located in one local optimum. Figure 3 shows the clouding bees with the close range $\lambda$ and the sharing range $d$, where the two diamonds represent the two bees, and the two black dashed and gray dotted circles represent the sharing range and the close range, respectively.

4.2 Four Rules of Adaptation Mechanism

The binary tree is composed of the four rules (i.e., Rules 1 to 4), which are designed for the initial, middle, and late stages of search by bees. In Fig. 2, the four rules are represented by the four areas (i.e., the black dashed line area, the gray dotted line area, the black dotted line area, and the gray dotted line area). In detail, Rule 1 is designed for the initial stage of search, which is applied when the situation satisfies the condition $\text{localrank}_i = 1$ and the other cloudingbees are not formed. Rule 2 is designed for the middle stage of search, which is applied when the situation satisfies the condition $\text{localrank}_i = 1$ and cloudingbees are not formed. Rules 3 and 4 are designed for the late stage of search. In particular, Rule 3 is applied when the situation satisfies the condition $\text{localrank}_i = 1$ and the other cloudingbees is located in the range $d_i$, and Rule 4 is applied when the situation satisfies the condition $\text{localrank}_i = 1$ and the other cloudingbees is not located in the range $d_i$. The situations of these four rules are illustrated in Fig. 4 (a)-(d). In these figures, the small diamond represents the bee $x_i$, the dashed line circle around the bee $x_i$ represents the sharing range $d_i$, and the dotted gray line circle represents the local minimum range $\lambda$. For the other bee, the gray diamond and the dashed and dot lines indicate bee $x_j$, the sharing range $d_j$ and the close range $\lambda$, respectively. The four rules are described in detail as follows.

First, Rule 1 shown in the black dashed line area of Fig. 2 is applied in the initial stage of search such as the case when the bees start their search or just after the bee becomes the scout bee. In this initial stage of search, the sharing range $d_i$ of bees is narrow, and the bees can hardly share information with other bees. In this case, each bee expands its sharing range $d_i$ by adding $\Delta d$ in order to communicate with other bees as shown in Fig. 4 (a).

Next, Rule 2 shown in the gray dashed line area in Fig. 2 is applied in the middle stage of search such as the case when the bees search for the good solutions by sharing information with other nearby bees. If the good local optimum can be found, other bees gather around it and the cloudingbees are formed. Since it is necessary to adjust the sharing range in order not to overlap two or more local optima, the cloudingbees change their own sharing range $d_i$ to the sharing range $d_j$ of the good solution $x_j$ which has $\text{localrank}_j = 1$ as shown in Fig. 4 (b).

Finally, Rule 3 and Rule 4 shown in the black and the gray dotted line areas in Fig. 2 are applied in the late stage of search such as the case after the cloudingbees are formed. The bee continues to search for the captured one local optimum. Although the sharing range should not be wide in order not to overlap two or more local optima, it needs to be wide in order to increase the global search ability. To tackle this problem, the sharing range $d_i$ is reduced as shown in Fig. 4 (c) if other cloudingbees is perceived within the sharing range $d_i$, otherwise the sharing range $d_i$ is expanded until the bees recognize other cloudingbees within the sharing range as shown in Fig. 4 (d).

5. Problem Description

5.1 Function

In order to verify the tracking ability of ABC-lis in the dynamic environments, the multimodal function represented by
Eq. (8) is designed:

\[ g(x, t) = 1 - \sum_{n=0}^{N_{peaks}} \left( \sin(\omega t + \alpha_n) + \frac{1}{2} \right) \exp\left( -\frac{1}{2} \sum_{m=0}^{M} \left( x_{mn} - \Phi_m \cos(nc_m + m\pi/2) \right)^2 \right), \]  

where \( N_{peaks} \) and \( M \) indicate the number of the local optima and the dimension of the vector \( x \), respectively. When \( M = 2 \), in particular, the \( N_{peaks} \) number of local optima are represented by Gaussian functions with a radius \( \Phi \) from the origin of the coordinate axes \((0, 0)\).

This function is designed as the minimization problem in which the evaluation values of these local optima change by \( \sin(\omega t + \alpha_n) \) as the discrete time \( t \) increases. Here, the time \( t \) denotes an elapsed time in the problem, and the evaluation value of the function is changed when \( t = t + 1 \). In this problem, \( t = t + 1 \) is calculated only once for each iteration of the algorithm. In other words, when the step 1 to the step 4 of the ABC algorithm is executed, \( t = t + 1 \). The following random factors are employed to implement dynamic environments from the viewpoint of the evaluation value of local and global optima, coordinates of local and global optima, and speed of change: (i) the evaluation values of local optima randomly change according to a set of the uniform random numbers \( \alpha = \{\alpha_n = \text{rand}(0, 2\pi) \} | n = 1, 2, \ldots, N_{peaks}\) (note that \( \text{rand}(0, 2\pi) \) means the uniform random number from 0 or more to less than \( 2\pi \)); (ii) the locations (coordinates) of the local optima change according to a set of the uniform random numbers \( \Phi = \{\Phi_n = \text{rand}(150, 500) \} | n = 1, 2, \ldots, N_{peaks}\) and \( c = \{c_n = \text{rand}(0, 2\pi/(N_{peaks} + 5)) \} | n = 1, 2, \ldots, N_{peaks}\) (note that the minimum solution transits randomly according to \( \Phi \) and \( c \)); and (iii) the speed of changing the local optima is determined by \( \omega = 2\pi / \text{oneCycle} \) in our experiments. The variable \( \text{oneCycle} \) determines the interval of one cycle in which three random numbers \( \Phi, c, \) and \( \alpha \) are updated. The variable \( \text{oneCycle} \) denotes the changing interval during the same speed in time \( t \). For example, when the time elapses from \( t = 0 \) to \( t = t + \text{oneCycle} \), \( \omega \) and three random numbers \( \Phi, c, \) and \( \alpha \) are updated, and then the next time to update is \( t = \text{oneCycle} \times 2 \). When \( \text{oneCycle} \) is short or long, the degree of changing local optima is large or small, respectively.

5.2 Dynamic Environment Design

For the various types of dynamic changes calculated by Eq. (8), the following five types of dynamic changes and one type of the high-dimensional problem are employed:

- **Environment A**: periodic change of evaluation values of local optima (the values of \( \Phi, c, \) and \( \alpha \) are the same value, and the minimum solution periodically transits at time \( t \));

- **Environment B**: random change of evaluation values of local optima (\( \alpha \) is updated every \( \text{oneCycle} \));

- **Environment C**: random change of local optima coordinates (\( \Phi \) and \( c \) are updated every \( \text{oneCycle} \));

- **Environment D**: combination of two kinds of random changes (B+C) (\( \Phi, c, \) and \( \alpha \) are updated every \( \text{oneCycle} \));

- **Environment E**: random speed change of local optima in the environment (D) (\( \Phi, c, \) and \( \alpha \) are updated every \( \text{oneCycle} \), while \( \text{oneCycle} \) is also randomly determined);

- **Environment F**: high-dimensional problem in the environment (E) (all parameters except for \( M \) are the same as the environment E).

Figure 5 shows the schematic diagram of the change of the local optima in the environments from A to D. This figure is composed of four graphs, where the vertical and horizontal axes indicate \( x_0 \) and \( x_1 \), respectively. The diamonds indicate the local optima and the gray line indicates their transitions. As shown in Fig. 5, the minimum (optimal) solution rotates from the local optima on the circle in the environment A, the minimum (optimal) solution changes randomly from the local optima on the circle in the environment B. On the other hand, in the environment C, the coordinates of the local optima randomly change, and the minimum (optimal) solution rotates from the local optima located randomly. Note that it may seem not to rotate in a strict sense, since the coordinates of the local optima change randomly. Finally, in the environment D, the minimum (optimal) solution changes randomly from the local optima located randomly. In the environments A to D, in particular, the timing of the local optima transition is constant as \( \text{oneCycle} \), while it determined at random in the environment E.

6. Experimental Setting

6.1 Comparison

As described in Section 1, the following three methods are compared in the experiments: ABC-alis as the proposed method; ABC-lis as our previous method of ABC-alis; and speciation-based particle swarm optimization (SPSO) [10] as the conventional method. We employ SPSO because it also limits the information sharing among individuals within the local range like ABC-lis. In SPSO, all individuals select their highest \( g\text{-best} \) among their personal best \((p\text{-best})\) within the Euclidean distance \( r \). Here, the group of individuals within the Euclidean distance \( r \) is called specie. By setting the upper limit \( p_{max} \) of the number of individuals composed one specie, the individuals in SPSO belong to their own species. Such features are very similar to ABC-lis and ABC-alis.

6.2 Evaluation Criteria

The tracking performance of the algorithm is evaluated from the viewpoint of the maximum, minimum, and median values.
of the distance to the optimal solution. Concretely, it is calculated by the distance between the minimum (optimal) solution coordinates of the function $g(x, r)$ and the solution coordinates found by the algorithm. In this paper, we assume that the distance which is less than 1 means that the algorithm can capture the optimal solution.

### 6.3 Parameters Setting

As the parameter setting, we have to set them from the viewpoint of the experiment setting, the problem (i.e., the multimodal function), and the methods (i.e., ABC-alis, ABC-lis, and SPSO) as follows

- **Experimental setting:** In all environments, the variable $t$ representing the time is calculated as $t = t + 1$ in one iteration, and the termination condition of one experiment is $t = 2000$. The problems are carried out 30 times in each of the environments from A to F.

- **Problem:** For the multimodal function represented by Eq. (8), $N_{\text{peaks}} = 10$ in all environments, while oneCycle $= 400$ in the environments from A to D and oneCycle is a random number from 1 to 400 in the environment E. The dimension is $M = 2$ in the environments A to E while $M = 6$ in the environment F.

- **Method:** For ABC-alis, $N_i = 100$, $N_i = 10$, $\Delta d = 10$, and $\lambda = 1$; for ABC-lis, $N_i = 100$, $N_i = 10$, and $d = 100$; and for SPSO, the number of individuals $= 200$, $p_{\text{max}} = 20$, $c_1 = c_2 = 1.4$, $w = 0.8$, and $r = 100$. In particular, the number of individuals in SPSO is twice of that in ABC-alis for the same number of evaluations in one iteration.

### 6.4 Sensitive Analysis of Important Parameters

In ABC-alis, $\Delta d$ and $\lambda$ are the important parameters to adaptively determine the parameter $d$. For this issue, we discuss how to determine $\Delta d$ and $\lambda$ appropriately. First, $\Delta d$ should be generally smaller than the search space, e.g., $\Delta d$ is around 1% of the search space from the pre-experiment. Since the parameter $\Delta d$ depends on the size of the search space, the following equation is proposed to determine $\Delta d$:

$$\Delta d = \frac{\sum_{m=1}^{M}(\text{bound}_{\text{upper}}^{m} - \text{bound}_{\text{lower}}^{m})}{100M} \times \text{element}, \ (9)$$

where $M$ is the number of dimension of problems, $\text{bound}_{\text{upper}}^{m}$ and $\text{bound}_{\text{lower}}^{m}$ are the upper and lower bounds of the search space in the $m$th dimension in a given problem. According to Eq. (9), $\Delta d$ is set as 10 because the difference between the upper and lower bounds of the search space is 1000, which is calculated by $\text{bound}_{\text{upper}}^{m} = 500$ and $\text{bound}_{\text{lower}}^{m} = -500$ in the problems. Second, $\lambda$ should be very smaller than $\Delta d$, e.g., $\lambda = 0.1 \times \Delta d$ from the pre-experiment. If $\lambda$ is larger than or equal to $\Delta d$, the tracking ability of the dynamic change of solutions decreases because the individuals form the clouding bees before expanding the sharing range $d$ enough by adding $\Delta d$.

To investigate the sensitive analysis of $\Delta d$ and $\lambda$, we conducted the several experiments ($5 \times 5$ cases) in the two-dimensional problem. Concretely, $\Delta d$ and $\lambda$ of ABC-alis are set as a combination of $\Delta d = 1, 5, 10, 50,$ and $100$ and $\lambda = 0.1, 0.5, 1, 5,$ and 10. Note that the other parameters (such as $N_s,$ and $N_i$) are set as the same as the environment E. Table 1 shows the results, which are the average value of the distance to the minimum solution of all times ($t = 0$ to 2000) in 30 runs. In Table 1, the distance to the minimum solution in $\Delta d = 10$ is smaller (better) than the others regardless of the value of $\lambda$, which suggests that 10 is appropriate for $\Delta d$. For this issue, it is generally difficult to find such an appropriate value of $\Delta d$ before conducting sensitive analysis, but what should be noted here is that $\Delta d = 10$ is somewhat appropriate if we want to check whether the method can track the dynamic change of solutions within 100 time. This is because $\Delta d = 10$ as 1% of the search space means that the whole search space can be covered by 100 time of expanding the sharing range $d$ with $\Delta d$. Even if $\Delta d$ is set as a larger or smaller value than 10, the distance to the minimum solution is mostly kept around 12 to 13 by setting $\lambda = 0.1 \Delta d$, which follows the condition where $\lambda$ should be very smaller than $\Delta d$.

### 7. Experiment and Discussion

#### 7.1 Experimental Results on Dynamic Change

| $\Delta d$ | 1  | 5  | 10 | 50 | 100 |
|------------|----|----|----|----|-----|
| $\lambda$  | 0.1| 0.5| 1  | 5  | 10  |
| 1          | 20.71 | 13.08 | 11.08 | 30.94 | 49.58 |
| 0.5        | 22.43 | 13.71 | 11.29 | 25.00 | 44.16 |
| 5          | 20.05 | 13.95 | 12.66 | 21.30 | 35.21 |
| 10         | 22.50 | 15.05 | 11.51 | 12.20 | 18.67 |
| 20         | 23.15 | 13.23 | 11.91 | 11.18 | 13.04 |

#### 7.2 Environment A: Periodic change of evaluation values of local optima

The results of the environment A are shown in Fig. 6 (a)–(c), where the vertical and horizontal axes indicate the distance to the minimum solution and the time $t$, respectively. The maximum, minimum, and median values of the distance to the minimum solution are represented by the dotted, dashed and solid lines, respectively. In detail, Fig. 6 (a), (b), and (c) show the result of SPSO, ABC-lis, and ABC-alis, respectively. From these results, the minimum and median values of SPSO, ABC-lis, and ABC-alis are smaller than 1 at most times, which suggests that all methods can track the dynamic periodical change of solutions. When focusing on the maximum value, the value of SPSO is larger than the others, while that of ABC-lis and ABC-alis is almost equal to the minimum value and median value. These results suggest that ABC-lis and ABC-alis have a high tracking ability in the dynamic periodical change.

#### 7.3 Environment B: Random change of evaluation values of local optima

The results of the environment B are shown in Fig. 7 (a)–(c). The configuration of this figure is the same as in Fig. 6. From these results, the median value of SPSO fluctuates between $10^{-2}$ and $10^{-4}$, while those of ABC-lis and ABC-alis are almost kept around $10^{-4}$. When focusing on the maximum value, the value of SPSO is around 1 or larger, while those of ABC-lis and ABC-alis are around $10^{-4}$ except for only the short time every 400 times, which is the transition time by oneCycle $= 400$. Since the maximum value of ABC-lis is larger than ABC-lis, the tracking ability of ABC-alis is slightly worse due to the in-
fluence of the change of the sharing range \( d_i \) by the mechanism described in Section 4.

- **Environment C: Random change of local optima coordinates**

The results of the environment C are shown in Fig. 8 (a)–(c). The configuration of this figure is the same as in Figs. 6 and 7. From these results, the median and the minimum values of SPSO, ABC-lis and ABC-alis are less than 1 except for only the short time every 400 times when the coordinates of local optima change. However, ABC-lis and ABC-alis can track changes more quickly than SPSO because the decreasing speed of the minimum value of ABC-lis and ABC-alis is faster than that of SPSO. When focusing on the maximum value, the values of SPSO, ABC-lis and ABC-alis are kept around 1 or larger, which suggests that the optimal solution cannot be found for a certain time in the worst case. Since the results of ABC-lis and ABC-alis are very similar, ABC-lis and ABC-alis have a high tracking ability in the random change of local optima coordinates.

- **Environment D: Combination of two kinds of random changes (B+C)**

The results of the environment D are shown in Fig. 9 (a)–(c). The configuration of this figure is the same as in Figs. 6–8. From these results, the median value of SPSO is shown in the range of \( 10^{-1} \) to \( 10^{-2} \), while those of ABC-lis and ABC-alis are shown in the range of \( 10^{-2} \) to \( 10^{-4} \). ABC-lis and ABC-alis can also track the dynamic change quickly because the decreasing speeds of the median value are faster than the that of SPSO.
These results suggest that the tracking ability of ABC-lis and ABC-alis are better than SPSO.

**Environment E: Random speed change of local optima in the environment (D)**

The results of the environment E are shown in Fig. 10 (a)–(c). The configuration of this figure is the same as in Figs. 6–9. From these results, the median value of SPSO is shown in the range from $10^{-1}$ to $10^{-6}$, while those of ABC-lis and ABC-alis are shown in the range from $10^{-4}$ to $10^{-7}$. When focusing on the minimum values, the values of ABC-lis and ABC-alis are shown in the range from $10^{-6}$ to $10^{-10}$. More importantly, the results are almost similar to those of Fig. 9 (a)–(c), because the time when the coordinate changes is randomly determined in the environment E.

### 7.2 Experimental Results on High-Dimensional Problem

**Environment F: High-dimensional problem in the environment (E)**

The results of the environment F are shown in Fig. 11 (a)–(c). The configuration of this figure is the same as in Figs. 6–10. From the results, the minimum value of SPSO cannot reach even 0.1, which suggests that SPSO cannot track the dynamic change in the high-dimensional space. On the other hand, the minimum value of ABC-lis can reach $10^{-6}$. However, ABC-lis can only track the dynamic change temporarily because its median and maximum values show the same trend as SPSO. Finally, the minimum value of ABC-alis is larger than the value of ABC-lis, but the median value of ABC-alis is smaller than...
the median value of ABC-lis, which suggests that ABC-lis cannot generally keep the tracking ability of the dynamic change of solutions in high dimensions, while ABC-alis can track it continuously more than ABC-lis and SPSO.

7.3 Discussion

From the results shown in the environments B, ABC-lis and ABC-alis have the higher tracking abilities than SPSO because ABC-lis and ABC-alis can keep the distance to the minimum solution around $10^{-2}$ which is smaller than that of SPSO. Since the evaluation values of the optimal solution change randomly while keeping local optimum coordinate in the environment B, ABC-lis and ABC-alis have the strong robustness against the random change of evaluation values of all local optima.

From the results shown in the environments C, D, and E, it is difficult to show the clear difference between the performance of ABC-lis and ABC-alis. To clarify the difference, this section calculates the numerical values of the tracking ability of three methods, ABC-alis, ABC-lis and SPSO, as the average value of the distance to the minimum solution of all times ($t = 0$ to 2000) in 30 trials in each experiment. Table 2 summarizes the results, where the average values of SPSO, ABC-lis, and ABC-alis are shown in the top, middle, and bottom lines, respectively.

From these results, the average values of ABC-alis are lower than those of ABC-lis and SPSO in the environments C, D, E, and F, while the values of ABC-lis in some environments are not much different from those of SPSO. This result suggests that ABC-alis can adapt to the dynamic changes of solutions in comparison with ABC-lis and SPSO. In other words, the results of the environments C, D, E, and F reveal that ABC-alis has the high tracking ability to the change of the coordinates of all local optima because all of these experiments except for the environments A and B have such a change.

Finally, the results of environment F suggest that ABC-alis can keep the tracking ability even in the dynamic change of the high-dimensional problem, which reveals that the adaptive mechanism of ABC-alis enables individuals (bees) to track the various changes such as the change of the coordinates of local optima and the change of the height of the local peaks even in high-dimensional problems.

8. Conclusions

To tackle dynamic optimization problems (DOPs), this paper proposed ABC-alis (artificial bee colony (ABC) algorithm based on adaptive local information sharing) by improving the ABC-lis [9] algorithm which is extended from the ABC algorithm. Concretely, ABC-alis changes the local sharing range $d$ adaptively according to the location of the individuals to cope with various dynamic changes. In order to verify the tracking ability of ABC-alis, ABC-alis is applied to the following types of dynamic changes embedded in DOPs: (A) periodic change of evaluation values of local optima; (B) random change of evaluation values of local optima; (C) random change of local optima coordinates; (D) combination of two kinds of random changes (B+C); (E) random speed change of local optima in the environment (D); and (F) high-dimensional problem in the environment (E). In these experiments, the following three methods are compared: ABC-alis as the proposed method, ABC-lis as our previous method of ABC-alis; and speciation-based particle swarm optimization (SPSO) as the conventional method. The experimental results revealed that the following implications: (1) ABC-alis and ABC-lis can capture the optimal solution more quickly and keep the better solutions than SPSO in the various types of dynamic changes, which can be found that the distances to the optimum of ABC-alis and ABC-lis become immediately smaller than that of SPSO in Figs. 6-11; (2) from environments C, D, and E, ABC-alis can adapt to the random change of local optima from the viewpoint of the evaluation value, coordinates, speed, and all of them, which can be found that the average values of ABC-alis are smaller than the values of ABC-lis and SPSO in Table 2; and (3) ABC-alis can maintain its performance even in the high dimensional environment F, which can be found that the average value of ABC-alis is smaller than the average values of ABC-lis and SPSO in Table 2 and the median value of ABC-alis is smaller than the values of ABC-lis and SPSO in Fig. 11.

The following work must be addressed in the near future: (1) a comparison of ABC-alis with other methods except for SPSO; (2) an improvement of ABC-alis to adaptively change the population size $N_p$ and/or the number of bees $N_l$ for an unknown number of peaks in multimodal problems.

References

[1] C. Cruz, J.R. Gonzalez, and D.A. Pelta: Optimization in dynamic environments: A survey on problems, methods and measures, Soft Computing, Vol. 15, No. 7, pp. 1427–1448, 2011.

[2] M. Mavrovouniotis, C. Li, and S. Yang: A survey of swarm intelligence for dynamic optimization: Algorithms and applications, Swarm and Evolutionary Computation, Vol. 33, pp. 1–17, 2017.

[3] H.F. Hanshar and B.M. Ombuki-Berman: Dynamic vehicle routing using genetic algorithms, Applied Intelligence, Vol. 27, No. 1, pp. 89–99, 2007.

[4] S. Yang, H. Cheng, and F. Wang: Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks, IEEE Transactions on Systems, Man, and Cybernetics, Part C, Vol. 40, No. 1, pp. 52–63, 2010.

[5] B.K. Panigrahi, V.R. Pandi, and S. Das: Adaptive particle swarm optimization approach for static and dynamic economic load dispatch, Energy Conversion and Management, Vol. 49, No. 6, pp. 1407–1415, 2008.

[6] D. Karaboga and B. Basturk: A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, Journal of Global Optimization, Vol. 39, No. 3, pp. 459–471, 2007.

[7] T. Nishida: Modification of ABC algorithm for adaptation to time-varying functions, The Transactions of the Institute of Electrical Engineers of Japan, Vol. 132-C, No. 4, pp. 584–591, 2012.

[8] R. Takano, T. Harada, H. Sato, and K. Takadama: Artificial bee colony algorithm based on local information sharing in dynamic environment, The Asia Pacific Symposium of Intelligent and Evolutionary Systems (IES), pp. 627–641, 2014.

[9] R. Takano, T. Harada, H. Sato, and K. Takadama: Toward robustness against environmental change speed by artificial bee colony algorithm based on local information sharing, 2015 IEEE Congress on Evolutionary Computation (CEC2015).

| Algorithm | Env. A | Env. B | Env. C | Env. D | Env. E | Env. F |
|-----------|--------|--------|--------|--------|--------|--------|
| SPSO      | 0.52   | 7.67   | 8.59   | 14.69  | 0.599  | 589.48 |
| ABC-lis   | 0.32   | 1.15   | 8.83   | 12.98  | 26.39  | 596.65 |
| ABC-alis  | 0.35   | 1.43   | 5.03   | 5.65   | 12.66  | 444.12 |
[10] D. Parrott and X. Li: A particle swarm model for tracking multiple peaks in a dynamic environment using speciation, Proceedings of the IEEE Congress on evolutionary computation, Vol. 1, pp. 98–103, 2004.

---

Ryo Takano

He received BS and MS degrees from the University of Electro-Communications in 2013 and 2015, respectively. He entered the University of Electro-Communications as a doctorate course student in 2017. His research interests include swarm intelligence and dynamic optimization problems.

Hiroyuki Sato

He received BS and MS degrees from Shinshu University in 2003 and 2005, respectively. In 2009, he received Ph.D. degree from Shinshu University. He has been working at the University of Electro-Communications since 2009. He is currently an associate professor. He received Best Paper Awards from GECCO (Genetic and Evolutionary Computation Conference) 2011, GECCO2014, Transaction of the Japanese Society for Evolutionary Computation in 2012 and 2015. His research interests include multi- and many-objective optimization, evolutionary computation, and their applications. He is a member of the Japanese Society for Evolutionary Computation, IPSJ, JSAI, IEEE, and ACM/SIGEVO.

Keiki Takadama (Member)

He received a Doctor of Engineering degree from the University of Tokyo, Japan, in 1998. He worked as a visiting researcher at Advanced Telecommunications Research Institute International (ATR) between 1998 and 2002 and then worked at Tokyo Institute of Technology as a lecturer between 2002 and 2006. He became an associate professor at the University of Electro-Communications in 2006 and has been a professor there since 2011. His research interests include multi-agent systems, reinforcement learning, and evolutionary computing.