A New Distributed Modified Extremal Optimization using Tabu Search Mechanism for Reducing Crossovers in Reconciliation Graph and Its Performance Evaluation

Keiichi Tamura, Hajime Kitakami

***Abstract*—To determine the mechanism of molecular evolution, identifying the differences between two heterogeneous phylogenetic trees and the between a phylogenetic tree and a taxonomic tree is an important task for molecular biologists. Phylogenetic trees and taxonomic trees are referred to as ordered trees. In the process of comparing ordered trees, a graph, which is called a reconciliation graph, is created using the ordered trees. In the reconciliation graph, the leaf nodes of the two ordered trees face each other. Furthermore, leaf nodes with the same label name are connected to each other by an edge. It is difficult to compare two heterogeneous ordered trees, if there are many crossed edges between leaf nodes in the reconciliation graph. Therefore the number of crossovers in the reconciliation graph should be decreased; then reducing crossovers in a reconciliation graph is the combinatorial optimization problem that finds the state with the minimum number of crossovers. Several heuristics have been proposed for reducing crossovers in a reconciliation graph. One of the most successful heuristics is the modified extremal-optimization-based heuristics (the MEO-based heuristics). In this paper, we propose a novel MEO-based heuristic called distributed modified extremal optimization with tabu lists (DMEOTL). This heuristic is a hybrid of distributed modified extremal optimization (DMEO) and the tabu search mechanism. We have evaluated DMEOTL using actual data sets. DMEOTL shows better performance compared with DMEO.**

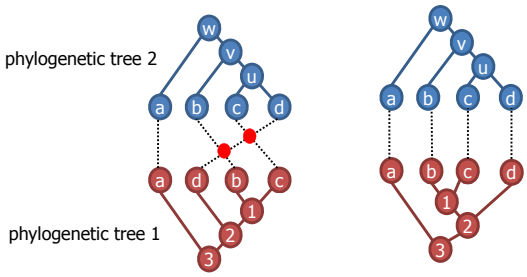
***Index Terms*—extremal optimization, distributed genetic algorithm, island model, tabu list, reconciliation graph**

I. Introduction

COMPARATIVE genomics is one of the most important research fields to reveal evolutionary relationship between organisms and the mechanism of evolutions. Especially, molecular biologists need to detect the main difference between two heterogeneous trees, which are phylogenetic trees and taxonomic trees, for determining the mechanism of molecular evolution in organisms [1], [2], [3], [4]. Phylogenetic trees and taxonomic trees are evolutionary trees showing the inferred evolutionary relationships among various biological organisms. A phylogenetic tree is a branching tree inferred from evolutionary relationships among species. A taxonomy tree is also a branching tree based on traditional biological classification. Identifying the difference between these heterogeneous trees is beneficial for many different application domains in comparative genomics.

K.Tamura is with Graduate School of Information Sciences, Hiroshima City University, 3-4-1, Ozuka-Higashi, Asa-Minami-Ku, Hiroshima 731-3194 Japan, corresponding e-mail: ktamura@hiroshima-cu.ac.jp

H.Kitakami is with Graduate School of Information Sciences, Hiroshima City University, 3-4-1, Ozuka-Higashi, Asa-Minami-Ku, Hiroshima 731-3194 Japan, corresponding e-mail: kitakami@hiroshima-cu.ac.jp



(a) (b)

Fig. 1. Examples of reconciliation graphs (In this figure, phylogenetic tree 1 and phylogenetic tree 2 are inferred from different molecular sequences with four identical species "a," "b," "c," and "d." Fig. 1(a) shows a reconciliation graph that has two crossovers, and Fig. 1 (b) shows a reconciliation graph that has no crossovers).

In this study, tasks that compare two heterogeneous trees and clarify differences between the two tree structures are called reconciliation work. In reconciliation work, a graph called a reconciliation graph that consists of two heterogeneous phylogenetic trees or a phylogenetic tree and a taxonomic tree is constructed. In a reconciliation graph, phylogenetic trees and taxonomic trees are referred to as ordered trees. Moreover, the leaf nodes of these ordered trees face each other and leaf nodes with the same label name are connected to each other by an edge. Fig. 1 shows an example of a reconciliation graph, which is composed phylogenetic tree 1 and phylogenetic tree 2.

It is difficult to compare two heterogeneous ordered trees, if there are many crossed edges between leaf nodes in the reconciliation graph. To accomplish reconciliation work efficiently, reducing crossed edges between leaf nodes in the reconciliation graph is required. The reconciliation graph shown in Fig. 1(a) has two crossovers. If order of node "1" and node "d" are replaced, we can obtain a optimal reconciliation graph shown in Fig. 1(b), which has no crossovers. Molecular biologists used to perform reducing crossovers in reconciliation graphs manually. Even though they can do this task using visualization tools, it is persnickety work for them to reduce large scale reconciliation graphs. Therefore, with increase in the number of nodes in a reconciliation graph, it is very difficult to make it manually, because the number of combinations increases exponentially as the number of leaf nodes increases.

To overcome this issue, there are several heuristics that can

be used for reducing crossovers in a reconciliation graph. The most simplest computational heuristic was proposed in [5]. This heuristic finds only a local optimal solution with a kind of local search. To improve the performance of search, a GA-based heuristic was proposed in [6]. There are two steps in the GA-based heuristic. First, the GA-based heuristic searches quasi-optimal solutions with simple GA. Second, the GA-based heuristic finds more better solutions from quasi-optimal solutions by using the local search. One of the most successful heuristics is the modified extremal-optimization-based heuristics (the MEO-based heuristics), which was proposed in [7].

In our previous study [8], we proposed a MEO-based heuristic called the distributed modified extremal optimization (DMEO), which is a hybrid of population-based modified extremal optimization (PMEO) [9] and the distributed genetic algorithm (DGA) [10], [11] using the island model [12], [13]. The MEO-based heuristics are based on extremal optimization (EO) [14], [15], [16]. EO is a general-purpose heuristic inspired by the Bak-Sneppen model [17] of self-organized criticality from the field of statistical physics. EO generates a neighbor individual, which is represented as a solution, randomly at an alternation of generations. On the other hand, modified extremal optimization (MEO) [7] generates multiple neighbor individuals and selects the best individual that has the best fitness value.

Generating multiple neighbor solutions results in improving the ability of search, because MEO can avoid falling into local optimal in regard to reducing crossovers in a reconciliation graph; however MEO has a drawback in the mechanism of generating neighbor individuals. MEO often becomes stuck in poor-evaluated areas or areas where evaluate plateau at the end of alternation of generations because multiple neighbor solutions are generated at random and the best individual in them are selected as the next generation. MEO does not have the mechanism to avoid pitfalls and explore regions of the search space that would be left unexplored. Therefore, MEO needs to carefully explore the neighborhood of each individual as the search progresses.

To address this performance issue, we propose a novel MEO-based heuristic called MEO with a tabu list (MEOTL) that involves the mechanism of tabu search [18]. Moreover, we incorporate MEOTL into DMEO. This MEO-based heuristic called DEMO with tabu lists (DMEOTL) [19]. The main contributions of this study are as follows.

- To avoid pitfalls and explore the regions in the search space that would be left unexplored, MEOTL is proposed. MEOTL utilizes the mechanism of tabu search, which can avoid generating recently visited individuals as neighbor individuals at each alternation of generations using tabu lists. A tabu list in MEOTL contains two or more ancestors of an individual. If a generated neighbor individual is contained in the tabu list, MEOTL generates another neighbor individual iteratively until MEOTL generates different individual from the ancestor individuals in the tabu list.
- To improve the performance of DMEO for reducing crossovers in a reconciliation graph, we incorporate MEOTL in DMEO. In DMEOTL, a population is divided into two or more sub-populations called islands and each island evolves individually. Each individual has a tabu list, individuals located in islands evolve using MEOTL, in DMEOTL.
- To evaluate MEOTL and DMEOTL, we implemented MEOTL and DMEOTL for reducing crossovers in a reconciliation graph. We evaluated MEOTL and its performance outperforms MEO. Moreover, we evaluated DMEOTL using two actual data sets for experiments. and compared DMEOTL with DMEO and MGG [20]. Experimental results shows that DMEOTL outperforms DMEO and MGG.

The rest of the paper is organized as follows. In Section II, the data structure of a reconciliation graph and the problem definition of reducing crossovers in a reconciliation graph are briefly explained. In Section III, we describe the details of the MEO-based heuristic. In Section IV, a new MEO-based heuristic called MEOTL is proposed. In Section V, the details of DMEOTL for reducing crossovers in reconciliation graph are presented. In Section VI, the results of experimental results are reported, and Section VII is the conclusion of the paper.

II. Reducing Crossovers in Reconciliation Graph

In this section, the data structure of a reconciliation graph and the problem definition of reducing crossovers in a reconciliation graph are briefly explained.

A. Reconciliation Graph

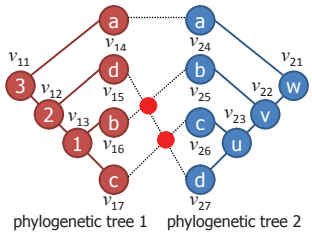


Fig. 2. Example of reconciliation graph (Order list OL_1 is given by $OL_1 = [v_{14}, v_{15}, v_{16}, v_{17}]$ and order list OL_2 is given by $OL_2 = [v_{24}, v_{25}, v_{26}, v_{27}]$).

Phylogenetic trees and taxonomic trees are referred to as ordered trees. A reconciliation graph (RG) consists of two ordered trees, $OT_1 = (V_1, E_1)$ and $OT_2 = (V_2, E_2)$, where V_1 and V_2 are finite sets of nodes and E_1 and E_2 are finite sets of edges. A node has a parent and multiple child nodes. Let $\mathcal{PT}(v)$ be the parent node of node v. If $\mathcal{PT}(v)$ is NULL, the node v is the root node. Let $\mathcal{CN}(v)$ be child nodes of node v. A node that has no child nodes is a leaf node. The leaf node sets of OT_1 and OT_2 are denoted by $L_1 \in V_1$ and $L_2 \in V_2$, respectively. If the number of species is n, the number of leaf nodes is n. A leaf node has a label name, which is a species' name. The label name set is denoted by L_{leaf}.

Let OL_1 and OL_2 be the order lists of leaf nodes:

$$OL_1 = [ol_{1,1}, ol_{1,2}, \cdots, ol_{1,n}](ol_{1,i} \in L_1, \mathcal{L}(ol_{1,i}) \in L_{leaf}),$$
$$OL_2 = [ol_{2,1}, ol_{2,2}, \cdots, ol_{2,n}](ol_{2,i} \in L_2, \mathcal{L}(ol_{2,i}) \in L_{leaf}),$$

where function $\mathcal{L}$ returns the label name of an input node.

The function $\mathcal{NC}(M)$ returns the number of crossovers:

$$\mathcal{NC}(CM) = \sum cm_{j,\beta} cm_{k,\alpha} [1 \le j < k \le n, \quad 1 \le \alpha < \beta \le n], \tag{1}$$

where $cm_{i,j}$ is (i,j)th-element of the connection matrix CM that is defined as

$$cm_{i,j} = \begin{cases} 1 & if\ \mathcal{L}(ol_{1,i}) = \mathcal{L}(ol_{2,j}), \\ 0 & otherwise. \end{cases} \tag{2}$$

In Fig. 2, OL_1 is given by $OL_1 = [v_{14}, v_{15}, v_{16}, v_{17}]$. Similarly, there are four leaf nodes in phylogenetic tree 2, $ol_{2,1} = v_{24}$, $ol_{2,2} = v_{25}$, $ol_{2,3} = v_{26}$, and $ol_{2,4} = v_{27}$. Therefore OL_2 is given by $OL_2 = [v_{24}, v_{25}, v_{26}, v_{27}]$. For example, the $(0,0)$th-element $cm_{0,0}$ is 1 because $\mathcal{L}(v_{14})$ equals $\mathcal{L}(v_{24})$. Similarly, the $(1,1)$th-element $cm_{1,1}$ is 0 because $\mathcal{L}(v_{15})$ does not equal $\mathcal{L}(v_{25})$. The following matrix is the connection matrix of the reconciliation graph shown in the Fig. 2.

$$CM = \begin{array}{c} \\ a \\ d \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

B. Problem Definition

The task of reducing crossovers in the reconciliation graph is defined as follows:

$$\begin{array}{rl} \textbf{min}: & \mathcal{NC}(CM), \\ \textbf{subject to}: & \text{(1) } CM \text{ is the connection matrix of the } RG, \\ & \text{(2) There are no crossovers on edges between non-leaf nodes in the } RG. \end{array}$$

There should be no crossovers on edges between non-leaf nodes in the reconciliation graph. For this constraint, we need to change order of leaf nodes by changing the order of child nodes in intermediate nodes. We cannot change the order between v_{15} and v_{17} (Fig. 2) because it will lead to the presence of crossovers on edges between non-leaf nodes. If we want to change the order between v_{15} and v_{17}, it is necessary to replace v_{15} and v_{13}, which are child nodes of v_{12}. If we replace v_{15} and v_{13}, the number of crossovers in the reconciliation graph becomes zero, and OL_1 is changed to $OL_1 = [v_{14}, v_{16}, v_{17}, v_{15}]$.

III. Modified Extremal Optimization

EO [14], [15], [16] follows the spirit of the Bak-Sneppen model, updating variables that have one of the worst values in a solution and replacing them by random values without ever explicitly improving them. There are many studies [14], [15], [16], [21], [22], [23], [24] that utilize EO to find optimal solutions for combinatorial optimization problems such as the traveling salesman problem, graph partitioning problem, and image rasterization. Some studies [25], [26] focus on improving EO mechanism and hybrid heuristics of EO and other heuristics.

In EO, an individual I consists of n components O_i $(1 \le i \le n)$. Let λ_i be the fitness value of O_i (Fig. 3). For each component, the fitness value of the component is evaluated. Then, we select the worst component. Finally, the state of the worst component is changed at random. The component with the worst fitness value has a high possibility that the fitness value of it will become better by changing state. Consequently, the fitness value of the individual also gets better because the fitness value of the component with worst fitness value gets better. While interactively changing the state of the worst component, the individual also improves overall, because the worst component might be getting better and better. Algorithm 1 shows the details of processing steps of EO. First, EO selects O_{worst}, which has the worst fitness value. Second, the state of component O_{worst} is changed at random. Henceforth, selection and change state

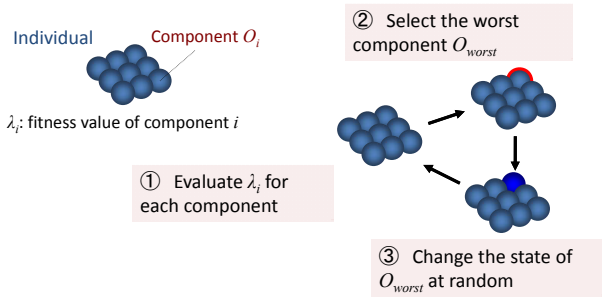


Fig. 3. Extremal optimization (There are multiple component in a individual. The state of the worst component is changed at random at alternation of generations).

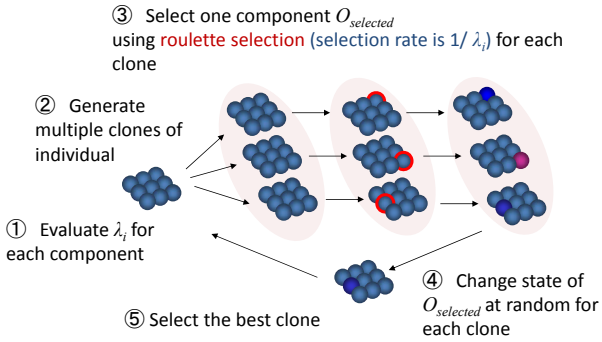


Fig. 4. Modified extremal optimization (In this heuristic, multiple neighbor individuals are generated. Then, the best of them is selected for the next generation).

Algorithm 1 EO

```
Generate initial individual I at random.
I_best ← I
m ← 0
while m < max_of_generations do
    Evaluate fitness value λ_i of each component O_i.
    Select O_worst with the worst fitness value.
    Change the state of O_worst at random.
    if F(I) > F(I_best) /* The function which returns the
    fitness value of an individual is denoted as F. */ then
        I_best ← I
    end if
    m ← m + 1
end while
```

Algorithm 2 MEO

1: Generate initial individual I at random.
2: $I_{best} \leftarrow I$
3: $m \leftarrow 0$
4: **while** $m < max\_of\_generations$ **do**
5: Evaluate fitness value λ_i of each component O_i.
6: $Candidates \leftarrow \phi$
7: $n \leftarrow 0$
8: **while** $n < num\_of\_candidates$ **do**
9: Select $O_{selected}$ with roulette selection according to fitness values of components.
10: Generate new individual I' from I by changing the state of $O_{selected}$.
11: $Candidates \leftarrow Candidates \cup I'$
12: $n \leftarrow n + 1$
13: **end while**
14: $I \leftarrow \mathbf{BEST}(Candidates)$
15: **if** $\mathbf{F}(I) > \mathbf{F}(I_{best})$ **then**
16: $I_{best} \leftarrow I$
17: **end if**
18: $m \leftarrow m + 1$
19: **end while**

of a component are repeated.

MEO improves the methodology of generating neighbor individuals in EO. EO uses the first admissible move strategy for the alternation of generations. Conversely, MEO utilizes the approximate best admissible move strategy. In EO, a neighbor individual is generated at random. In reducing crossover in reconciliation graph, global change occurs more easily than other combinatorial optimization problems. Therefore, only generating one neighbor individual results in generating worse neighbor individual. To overcome this issue, MEO [7] generates multiple neighbor individuals as candidates for the next generation individual (Fig. 4). The experimental results in [7] show that MEO outperforms EO. Moreover, MEO is good performance compared with the GA-based heuristic [5], [6].

Algorithm 2 shows the details of processing steps of MEO. MEO generates multiple neighbor individuals at each alternation of generations through the following steps (step 8 - step 13). First, MEO selects $O_{selected}$ with roulette selection (step 9). The selection rates of roulette selection are reciprocals of fitness values with components. Second, MEO generates new individual I' from I by changing the state of $O_{selected}$ (step 10). Third, the generated I' is stored into $Candidates$ (step 11). After generating multiple neighbor individuals, MEO selects the best individual from $Candidates$ (step 14).

IV. Modified Extremal Optimization with Tabu List

In this section, we propose a novel MEO-based framework called modified extremal optimization with tabu lists (MEOTL), which is a hybrid of MEO and the tabu search mechanism. The tabu search has a tabu list, which contains a set of ancestor solutions, in order to avoid visiting the search space that has been explored. MEOTL has a tabu list the same as the tabu search (Fig. 5). The original intent of the tabu list is not to prevent a previous move from being repeated, but rather to insure it is not reversed. MEOTL can

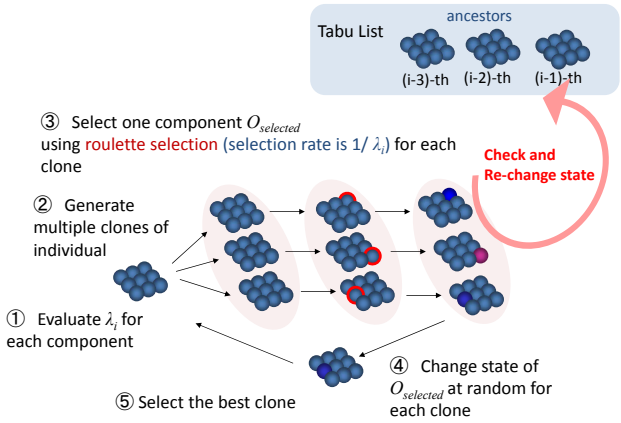


Fig. 5. Modified extremal optimization with a tabu list (MEOTL has a tabu list. At each changing state, each generated neighbor individual is checked using the tabu list).

improve the efficiency of search through the use of the tabu search mechanism.

Algorithm 3 shows the details of processing steps of MEOTL. MEOTL has a tabu list TL. The tabu list TL is a list of previous visited individuals. MEOTL generates multiple neighbor individuals at each alternation of generations through the following steps (step 9 - step 16) like MEO. First, MEOTL selects $O_{selected}$ with roulette selection (step 10). The selection rates of roulette selection are reciprocals of fitness values with components. Second, MEOTL generates new individual I' from I by changing the state of $O_{selected}$ (step 11). Third, if the generated I' is not in TL, the generated I' is stored into $Candidates$ (step 12 - step 15). Otherwise, MEOTL generates new individual I' from I by changing the state of $O_{selected}$ until I is not in TL. After generating multiple neighbor individuals, MEOTL selects the best individual from $Candidates$ (step 17). Finally, TL is updated (step 21).

V. Distributed Modified Extremal Optimization with Tabu Lists

In this section, the details of DMEOTL for reducing crossovers in a reconciliation graph are explained.

A. Alternation of Generations Model

DMEOTL is based on a MEO-based heuristic DMEO, which is a hybrid of PMEO and the island-model. In PMEO, there are two or more individuals in a population. Alternation of generations is repeatedly performed for every individual by using MEO. In the island model, a population is divided into two or more sub-populations called islands and each island evolves individually. The island model also has the migration mechanism that some individuals are transferred to another island. Each island can maintain different types of individuals at the end of alternation of generations. Therefore, DMEO can maintain diversity at the end of alternation of generations.

DMEOTL divides the entire population into two or more sub-populations, as islands like DMEO. In an island, each individual in the sub-population on the island evolves using MEOTL. Fig. 6 shows an example of the DMEOTL. There are two main steps in DMEOTL for reducing crossovers in

Algorithm 3 MEOTL

1: Generate initial individual I at random.
2: $I_{best} \leftarrow I$
3: $m \leftarrow 0$
4: $TL \leftarrow \phi$
5: **while** $m < max\_of\_generations$ **do**
6: Evaluate fitness value λ_i of each component O_i.
7: $Candidates \leftarrow \phi$
8: $n \leftarrow 0$
9: **while** $n < num\_of\_candidates$ **do**
10: Select $O_{selected}$ with roulette selection according to fitness values of components.
11: Generate new individual I' from I by changing the state of $O_{selected}$.
12: **if IS_INCLUDED**(I', TL) == false **then**
13: $Candidates \leftarrow Candidates \cup I'$
14: $n \leftarrow n + 1$
15: **end if**
16: **end while**
17: $I \leftarrow \mathbf{BEST}(Candidates)$
18: **if** $\mathbf{F}(I) > \mathbf{F}(I_{best})$ **then**
19: $I_{best} \leftarrow I$
20: **end if**
21: $TL \leftarrow \mathbf{UPDATE\_TABU\_LIST}(I, TL)$
22: $m \leftarrow m + 1$
23: **end while**

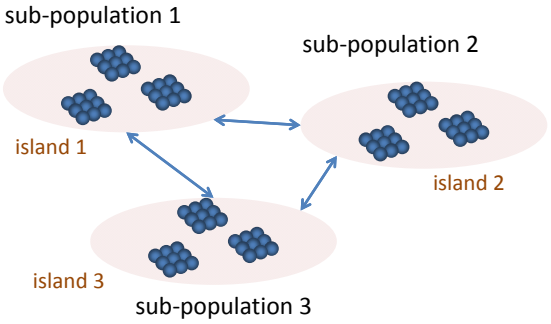


Fig. 6. Distributed modified extremal optimization with tabu lists (DMEOTL divides the entire population into two or more sub-populations, as many islands).

a reconciliation graph: (1) Evolution Step and (2) Migration Step.

(1) Each individual in the sub-population on a island evolves using MEOTL. Moreover, good sub-structures of a selected individual are copies to the other individuals in the sub-population at each alternation of generations.
(2) Some individuals in islands are migrated to other islands.

B. Definition of Individual and Component

In the MEO-based heuristics, we need to define the structure of individual representing a solution and the components. In this study, a reconciliation graph is referred as to as an individual. A pair of leaf nodes with the same label name is defined as a component:

$$O_i = \{ol_{1,i}, ol_{2,\delta(i)}\} \quad (\mathcal{L}(ol_{1,i}) = \mathcal{L}(ol_{2,\delta(i)})). \tag{3}$$

Algorithm 4 DMEOTL

1: Generate initial population P_{init} at random.
2: $I_{best} \leftarrow \mathbf{BEST}(P_{init})$
3: Divide P_{init} into p sub-populations.
4: For each sub-population $SubP_i$, create initial tabu list set STL_i.
5: Store sub-populations $SubP_i$ into island $ISLND_i$.
6: **for** $i = 1$ to $max\_generations/m$ **do**
7: **(Evolution Step)** For each $ISLND_i$, sub-population $SubP_i$ should be made to evolve through m generations by using the function $\mathbf{P\_MEOTL}(SubP_i, STL_i, m)$.
8: **(Migration Step)** For each $ISLND_i$, migrate some individuals of a sub-population in the island to another island.
9: **if** $\mathbf{F}(\mathbf{BEST}(SubP_1 \cap \cdots \cap SubP_p)) > \mathbf{F}(I_{best})$ **then**
10: $I_{best} \leftarrow \mathbf{BEST}(SubP_1 \cap \cdots \cap SubP_p)$
11: **end if**
12: **end for**

Let $ol_{1,i}$ be a leaf node of OL_1 and $ol_{2,\delta(i)}$ be a leaf node of OL_2. The function $\delta(i)$ returns the subscript number of an element of OL_2 whose label name is the same as the label name of $ol_{1,i}$. To change the state of O_i, it is necessary to change the order of child nodes of ancestor nodes of $ol_{1,i}$ or $ol_{2,\delta(i)}$. Here, $\mathcal{AS}(OT, lname)$ is a set of ancestor nodes of a leaf node in OT that has the label name $lname$. For example, $\mathcal{AS}(OT_1, O_2)$ returns $\{v_{12}, v_{11}\}$ in Fig. 2.

C. Definition of Fitness

The number of crossovers between $ol_{1,i}$ and $ol_{2,\delta(i)}$ is denoted by $\mathcal{NC}(CM, i)$. The following are the definitions of $\mathcal{NC}(CM, i)$ and the fitness value λ_i of O_i:

$$\lambda_i = \frac{\mathcal{NC}(CM) - \mathcal{NC}(CM, i)}{\mathcal{NC}(CM)}, \tag{4}$$

$$\mathcal{NC}(CM, i) = \sum_{l=i+1}^{n} \sum_{m=1}^{\delta(i)-1} \frac{cm_{l,m}}{2} + \sum_{l=1}^{i-1} \sum_{m=\delta(i)+1}^{n} \frac{cm_{l,m}}{2}. \tag{5}$$

In Fig. 2, there are four components, $O_1 = \{ol_{1,1}, ol_{2,1}\} (= \{v_{14}, v_{24}\})$, $O_2 = \{ol_{1,2}, ol_{2,4}\} (= \{v_{15}, v_{27}\})$, $O_3 = \{ol_{1,3}, ol_{2,2}\} (= \{v_{16}, v_{25}\})$, and $O_4 = \{ol_{1,4}, ol_{2,3}\} (= \{v_{17}, v_{26}\})$, with $\delta(1) = 1$, $\delta(2) = 4$, $\delta(3) = 2$, and $\delta(4) = 3$. The fitness values of the components are $\lambda_1 = 1$, $\lambda_2 = 1/2$, $\lambda_3 = 3/4$, and $\lambda_4 = 3/4$.

D. Algorithm

Algorithm 4 shows the processing steps of DMEOTL. First of all, an initial population consisting of multiple individuals is generated at random. Then the initial population is divided into p sub-populations (p is the number of sub-populations) (step 3). Moreover, for each sub-population, the initial tabu list set is created (step 4). Sub-population $SubP_i$ is located in an island $ILND_i$ (step 5). In the Evolution Step (step 7), the sub-populations in all the islands are made to evolve through m generations by using the function $\mathbf{P\_MEOTL}(SubP_i, STL_i, m)$ (m is migration interval). In Migration Step (step 8), for each island, some individuals of the sub-population on the island are migrated to another

Algorithm 5 $\mathbf{P\_MEOTL}(\mathbf{P}, \mathbf{STL}, \mathbf{m})$

1: **for** $i = 1$ to m **do**
2: **for all** $I \in P$ **do**
3: Evaluate fitness value λ_i of each component O_i of I.
4: $C \leftarrow \phi$
5: $n \leftarrow 0$
6: **while** $n < num\_of\_candidates$ **do**
7: Select $O_{selected}$ by roulette selection (selection rates are the reciprocal of fitness values with components).
8: $I' \leftarrow \mathbf{GNI}(I, O_{selected})$
9: **if** $\mathbf{IS\_INCLUDED}(I', STL[I])$ == false **then**
10: $C \leftarrow C \cup \mathbf{GNI}(I, O_{selected})$
11: $n \leftarrow n + 1$
12: **end if**
13: **end while**
14: $I \leftarrow \mathbf{BEST}(C)$
15: **end for**
16: **for all** $I \in P$ **do**
17: Select $O_{selected}$ by roulette selection. Copy the good structure of $O_{selected}$ to I using the function $\mathbf{CGS}(I,P)$.
18: **end for**
19: $STL[I] \leftarrow \mathbf{UPDATE\_TABU\_LIST}(P, STL[I])$
20: **end for**

island. Finally, the best individual is selected from all the islands (step 9 and step 10).

E. Evolution Step

For each island, the sub-population on the island evolves through m generations by using the function $\mathbf{P\_MEOTL}$ (Algorithm 5). At each alternation of generations, first, for each individual in a sub-population, the state of the individuals in P is changed by using the MEOTL scheme. Second, for each individual, the individual copies a good sub-structure of another individual using the function $\mathbf{CGS}$.

At each alternation of generations, each individual in a sub-population evolves following steps in the function $\mathbf{P\_MEOTL}$. Initially, for each component, the fitness value λ_i (step 3) is evaluated. Next, the following three steps are repeated while n is less than $num\_of\_candidates$. First, component $O_{selected}$ in I is selected by using the roulette selection (step 7). Second, a neighbor individual from I is generated by using the function $\mathbf{GNI}$. The function $\mathbf{GNI}$ generates a neighbor individual I' by changing the state of component $O_{selected}$. Third, I' is stored in C (step 10), if I' is not in $STL[I]$, where $STL[I]$ is the tabu list of individual I. Finally, the best individual in C is selected and I is replaced by it (step 14). After evolving, for each individual, the individual copies a good sub-structure of another individual using the function $\mathbf{CGS}$ and STL is updated.

Changing state of a selected component $O_{selected}$ is performed by changing the order of child nodes in an intermediate node, which is an ancestor node of $O_{selected}$. The processing steps of $\mathbf{GNI}$ are as follows. First, OT_1 or OT_2 is selected at random. If OT_1 is selected,

Algorithm 6 $\mathbf{CGS}(\mathbf{I}, \mathbf{P})$

1: Select individual $SI \in P$ by roulette selection (selection rates are the fitness values of components).
2: **for** $i = 1$ to n **do**
3: Calculate the difference $diff_i$ between the fitness value of O_i in SI and the fitness value of O_j in I, where O_i and O_j have the same label name.
4: **end for**
5: Select $O_{selected}$ by roulette selection (selection rates are $diff_i$).
6: $A \leftarrow \mathcal{AS}(OT_1, \mathcal{L}(O_{selected}))$ or $\mathcal{AS}(OT_2, \mathcal{L}(O_{selected}))$
7: $C \leftarrow \phi$
8: **for all** $a \in A$ **do**
9: Generate a new individual I' from I by changing the order of child nodes in a.
10: $C \leftarrow C \cup I'$
11: **end for**
12: $I \leftarrow \mathbf{BEST}(C)$

$\mathcal{AS}(OT_1, \mathcal{L}(O_{selected}))$ is put into the set $Ancestor$. If OT_2 is selected, $\mathcal{AS}(OT_2, \mathcal{L}(O_{selected}))$ is put into the set $Ancestor$. Then, node a is selected at random from A. Finally, the order of the child nodes in a is changed. Suppose that the selected component is O_2 in Fig. 2. The function $\mathcal{AS}(OT_1, \mathcal{L}(O_2))$ returns $\{v_{12}, v_{11}\}$ and $\mathcal{AS}(OT_2, \mathcal{L}(L_2))$ returns $\{v_{22}, v_{21}\}$. If $Ancestors = \{v_{12}, v_{11}\}$ and v_{12} is selected as a, the order of child nodes in v_{12} is changed. In this case, order of node v_{15} and v_{13} are changed. As a result, a new individual I' is obtained by the change state.

Algorithm 6 shows the function $\mathbf{CGS}$. At the beginning, an individual SI in P is selected by roulette selection (step 1). The individual I copies a sub-structure of SI by the following steps. First, the function calculates the difference $diff_i$ between the fitness value of O_i of SI and the fitness value of O_j of I, where O_i and O_j have the same label name (steps 2, 3, and 4). Second, $O_{selected}$ is selected by roulette selection (step 5). Next, $\mathcal{AS}(OT_1, \mathcal{L}(O_{selected}))$ or $\mathcal{AS}(OT_2, \mathcal{L}(O_{selected}))$ is stored in A (step 6). Then, for all $a \in A$, a new individual I' is generated from I by changing the order of child nodes in a, and I' is stored in C (steps 8, 9, 10, and 11). Finally, the function selects the best individual from C (step 12).

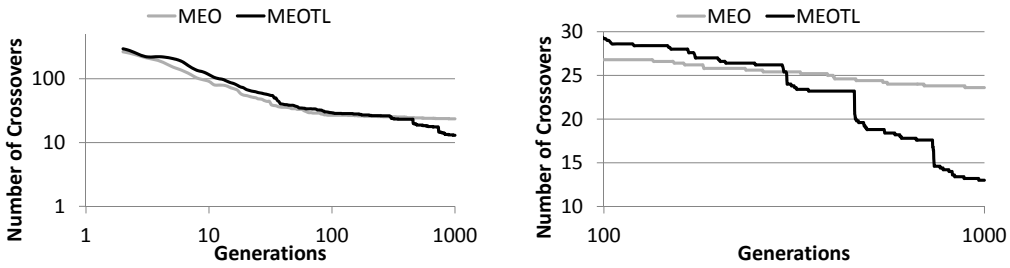
F. Migration Step

In the population-based evolutionary computation, it is difficult to maintain the diversity of population. Evolving sub-population independently has efficacy for conservation of diversity. However, this method increases risk of falling local optimal solutions in the island because the size of population usually becomes smaller than the population-based model. To avoid falling local optimal solutions in each island, the island has the mechanism of migration. In the Migration Step, some individuals in each island are migrated to another island. Therefore, different types of individual can make sub-populations avoiding falling local optimal solutions.

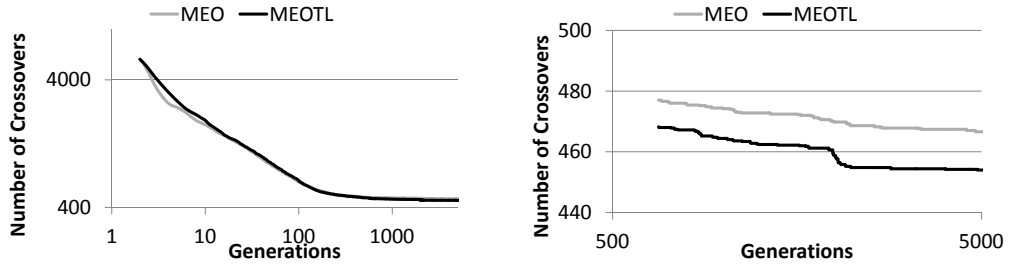
The island model requires $number\ of\ sub-populations$, $migration\ rate$, $migration\ interval$, and $migration\ model$. The first three items are user-

TABLE I
DATA SETS

	Taxonomic tree		Phylogenetic tree	
	Number of nodes	Number of leaf nodes	Number of nodes	Number of leaf nodes
$Housekeeping$	241	40	79	40
$Moss$	290	207	394	207



(a) $Housekeeping$ data set



(b) $Moss$ data set

Fig. 7. Experiment 1 (This experiment compares MEOTL with MEO. Figure (a) and (b) shows the number of crossovers of the best individual using $Housekeeping$ data set and $Moss$ data set respectively) .

given parameters. The last item consists of two things: $selection\ method$ and $topology$. The method used for the selection of individuals for migration is referred as $selection\ method$. The structure of the migration of individuals between sub-populations is referred as $topology$. In this study, we use uniform random selection as the $selection\ method$. In the Migration Step, some individual are selected from a sub-population in each island according to $migration\ rate$. Moreover, the proposed algorithm uses the random ring migration topology. In this topology, the ring includes all islands, and the order of the islands is determined randomly every Migration Step. Each island transfers some individuals to the next inland based on the direction of the ring.

VI. Experimental Results

To evaluate the performance of DMEOTL, five experiments were conducted in the performance evaluation. In this section, we report the result of experimental results.

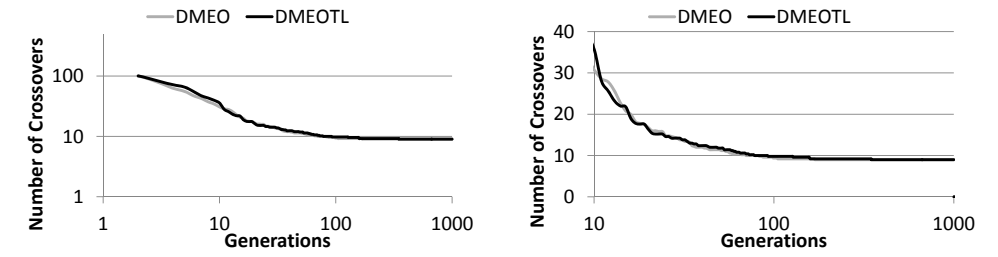
A. Setup

In the performance evaluation, we conducted five experiments and the two actual data sets listed in Table I were used. The $Housekeeping$ data set consists of a phylogenetic tree of the housekeeping gene and its taxonomic tree. The $Moss$ data set consists of a phylogenetic tree of the rps4 gene and its taxonomic tree. The number of species in the $Housekeeping$ data set is 40 and that in the $Moss$ data set is 207. In Experiment 1, we measured the number of crossovers of the best individual at each generation to compare MEOTL with MEO. In Experiment 2, we also measured the number of crossovers of the best individual at each generation to compare DMEOTL with DMEO. In Experiment 3, we measured the number of crossovers of the best individual at different time instants. In Experiment 4, we measured frequency of the number of crossovers of best individuals in fixed generations. In Experiment 5, DMEOTL is compared with MGG.

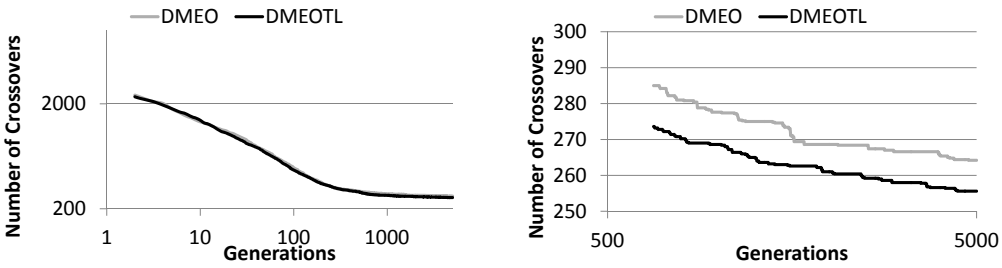
B. Experiment 1

In Experiment 1, we compared MEOTL with MEO. We measured the number of crossovers of the best individual in each generation. In MEO and MEOTL, $num\_of\_candidates$ was set to 50. The maximum number of generations of the $Housekeeping$ data set and the $Moss$ data set were set to 1000 and 5000 respectively. In MEOTL, the length of a tabu list was set to 10. Fig. 7(a) and Fig. 7(b) show the number of crossovers (vertical axis: the number of crossovers, horizontal axis: generations) at each generation. The number of crossovers is the average of five trials in these graphs.

Fig. 7(a) shows that the number of crossovers of MEOTL is less than that of MEO at the end of alternation of generations. MEO fell into a cycle of local optimal solutions;



(a) $Housekeeping$ data set



(b) $Moss$ data set

Fig. 8. Experiment 2 (This experiment compares MEOTL with MEO. Figure (a) and (b) shows the number of crossovers of the best individual using $Housekeeping$ data set and $Moss$ data set respectively) .

however MEOTL could get away from local optimal solutions. Fig. 7(b) shows the results of the $Moss$ data set. The number of crossovers of MEOTL is less than that of MEO after 100-th generations. Experiment 1 shows that MEOTL outperforms MEO.

C. Experiment 2

In Experiment 2, we compared DMEOTL with DMEO. We also measured the number of crossovers of the best individual in each generation. In DMEO and DMEOTL, the number of individuals in the population was set to 50. The user parameters $num\_of\_candidates$, $migration\_interval(m)$, $number\ of\ sub\text{-}populations(p)$, and $migration\ rate$ were set to be 50, 10, 5 and 0.1, respectively. In DMEOTL, the length of a tabu list was set to 10. The number of individuals in a sub-population is 10. The number of crossovers was the average of five trials. Since the size of Housekeeping data set is small, the results of both heuristics are not different. Fig. 8(b) shows that the number of crossovers of DMEOTL is less than that of DMEO when we used the $Moss$ data set. The number of crossovers of DMEOTL is less than that of DMEO after 100-th generations. Experiment 2 shows that DMEOTL outperforms DMEO.

D. Experiment 3

The computation time of DMEOTL is longer than that in the case of DMEO because the former includes the matching generated individuals to individuals in tabu lists. Therefore, it is necessary to compare the number of crossovers for the same computation time. In Experiment 3, we measured the number of crossovers of the best individual at each elapsed processing time. Fig. 9 shows the number of crossovers at at wash elapsed processing time (vertical axis: the number of crossovers, horizontal axis: elapsed processing time), when

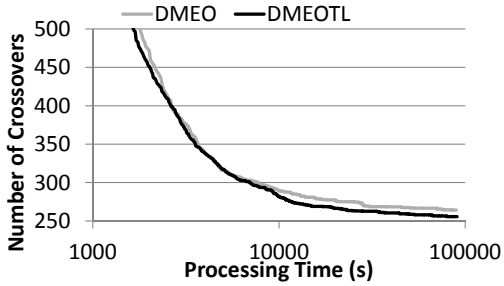
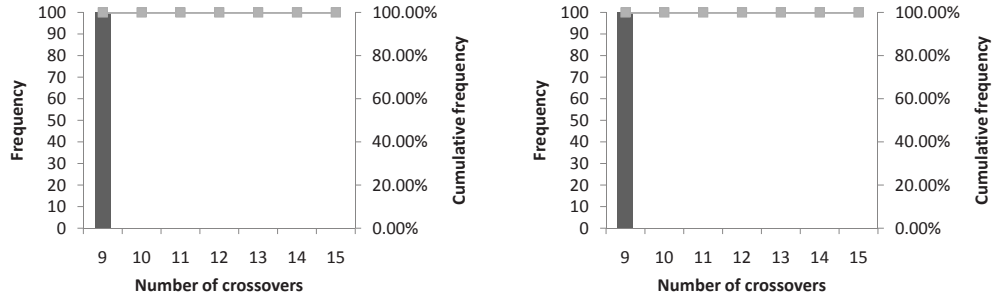


Fig. 9. Experiment 3 ($Moss$ data set).

the $Moss$ data set is used. At the end of the processing, DMEOTL have fewer crossovers than DMEO. This result indicates DMEOTL performs better with fewer crossovers than DMEO.

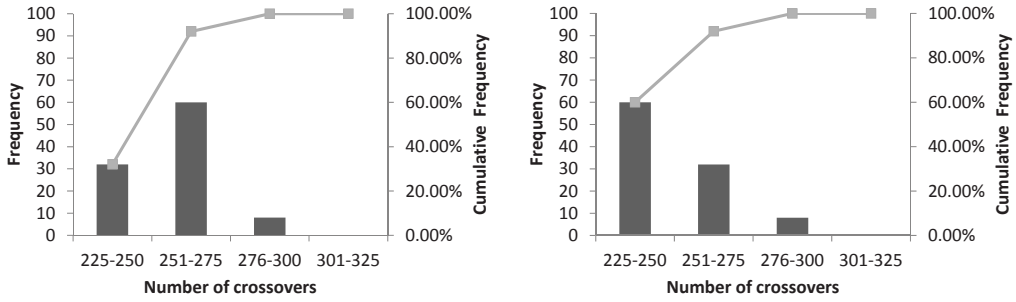
E. Experiment 4

The number of crossovers of the best individual was measured 100 times for the 5,000-th alternation generation. In DMEO and DMEOTL, the number of individuals in the population was set to 50. The user parameters $num\_of\_candidates$, $migration\_interval(m)$, $number\ of\ sub\text{-}populations(p)$, and $migration\ rate$ were set to be 50, 10, 5 and 0.1, respectively. In DMEOTL, the length of a tabu list was set to 10. The number of individuals in a sub-population is 10. Fig. 10(a) and Fig. 10(b) show frequency of the number of crossovers when $Housekeeping$ data set is used. Fig. 11(a) and Fig. 11(b) show frequency of the number of crossovers when $Moss$ data set is used. In DMEO, 32% of optimal solutions were between 225 and 250. On the other hand, in DMEO, 60% of optimal solutions were between 225



(a) DMEO (b) DMEOTL

Fig. 10. Experiment 4 (This experiment compares DMEOTL with DMEO. Figure (a) and (b) shows frequency of the number of crossovers of the best individual when we used $Housekeeping$ data set) .



(a) DMEO (b) DMEOTL

Fig. 11. Experiment 4 (This experiment compares DMEOTL with DMEO. Figure (a) and (b) shows frequency of the number of crossovers of the best individual when we used $Moss$ data set) .

and 250. This result indicates the search ability of DMEOTL is better than that of DMEO.

F. Experiment 5

In Experiment 5, we compared DMEOTL with MGG, , which is the one of the best generation alternation models. In DMEO and DMEOTL, the number of individuals in the population was set to 50. The user parameters $num\_of\_candidates$, $migration\_interval(m)$, $number\ of\ sub\text{-}populations(p)$, and $migration\ rate$ were set to be 50, 10, 5 and 0.1, respectively. In DMEOTL, the length of a tabu list was set to 10. The number of individuals in a sub-population is 10. In MGG, the number of individuals is 50, the number of children is 100, and the rate of mutation is 5%.The number of crossovers was the average of three trials. Fig. 12 show the number of crossovers (vertical axis: the number of crossovers, horizontal axis: generations). The number of crossovers using DMEOTL at the 5000-th generation is 255. In MGG, at more than 30000-th generation, the number of crossovers became less than 255. The performance of DMEOTL is better than that of MGG. Especially, the speed of convergence in DMEOTL is faster than that in MGG.

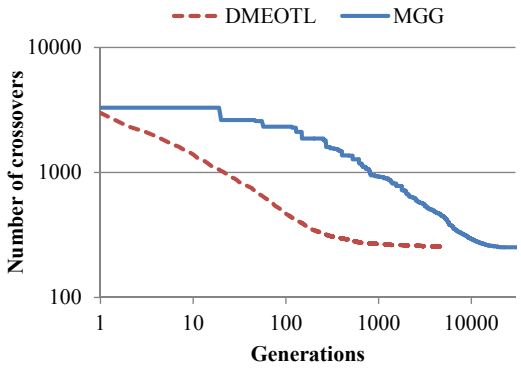


Fig. 12. Experiment 5 ($Moss$ data set).

VII. Conclusion

In this paper, we have proposed a novel MEO-based heuristic called MEOTL, which is a hybrid of MEO and the tabu search mechanism. MEOTL has a tabu list to avoid pitfalls and explore regions of the search space that would be left unexplored. Moreover, MEOTL is integrated in DMEO, which was proposed in our previous work. This new MEO-based heuristic is called DMEOTL. DMEOTL is a hybrid of distributed modified extremal optimization (DMEO) inspired by the island model and the tabu search mechanism. In the island model, a population is divided into two or more sub-populations called islands and each island evolves individually. Each island can maintain different types

of individuals at the end of alternation of generations. Therefore, DMEOTL can maintain diversity at the end of alternation of generations. Moreover, each individual has a tabu list, individuals located in islands evolve using MEOTL, in DMEOTL. Therefore, DMEOTL also can avoid pitfalls and explore regions of the search space that would be left unexplored. To evaluate MEOTL and DMEOTL, we used two actual data sets composed of a phylogenetic tree and its taxonomic tree. Experimental results show that MEOTL and DMEOTL is better performance compared with MEO and DMEO respectively. In the future work, we are going to develop improving the Migration Step. The Migration Step uses the simple island model. The migration topology and migration rate is required to improve. We are also going to develop extended DMEOTL for making it applicable to other combination optimization problems.

ACKNOWLEDGMENT

This work was supported in part by Hiroshima City University Grant for Special Academic Research (General Studies) and JSPS KAKENHI Grant Number 26330139.

REFERENCES

[1] M. Goodman, J. Czelusniak, G. Moore, A. Romero-Herrera, and G. Matsuda, "Fitting the gene lineage into its species lineage, a parsimony strategy illustrated bycladograms constructed from globin sequences," *Systematic Zoology*, vol. 28, pp. 132–163, 1979.

[2] R. Page, "Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas," *Systematic Biology*, vol. 43, pp. 58–77, 1994.

[3] R. D. M. Page and M. A. Charleston, "Reconciled trees and incongruent gene and species trees," *Discrete Mathametics and Theoretical Computer Science*, vol. 37, pp. 57–70, 1997.

[4] R. D. M. Page, "Genetree: comparing gene and species phylogenies using reconciled trees," *Bioinformatics*, vol. 14, no. 9, pp. 819–820, 1998.

[5] H. Kitakami and M. Nishimoto, "Constraint satisfaction for reconciling heterogeneous tree databases," in *Proceedings of DEXA 2000*, 2000, pp. 624–633.

[6] H. Kitakami and Y. Mori, "Reducing crossovers in reconciliation graphs using the coupling cluster exchange method with a genetic algorithm," *Active Mining, IOS press*, vol. 79, pp. 163–174, 2002.

[7] K. Tamura, Y. Mori, and H. Kitakami, "Reducing crossovers in reconciliation graphs with extremal optimization (in japanese)," *Transactions of Information Processing Society of Japan*, vol. 49, no. 4(TOM 20), pp. 105–116, 2008.

[8] K. Tamura, H. Kitakami, and A. Nakada, "Distributed modified extremal optimization using island model for reducing crossovers in reconciliation graph," *Engineering Letters*, vol. 21, no. 2, pp. 81–88, 2013.

[9] N. Hara, K. Tamura, and H. Kitakami, "Modified eo-based evolutionary algorithm for reducing crossovers of reconciliation graph," in *Proceedings of NaBIC 2010*, 2010, pp. 169–176.

[10] R. Tanese, "Distributed genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, 1989, pp. 434–439.

[11] T. C. Belding, "The distributed genetic algorithm revisited," in *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, pp. 114–121.

[12] W. D. Whitley, S. B. Rana, and R. B. Heckendorn, "Island model genetic algorithms and linearly separable problems," in *Selected Papers from AISB Workshop on Evolutionary Computing*, 1997, pp. 109–125.

[13] F. Seredyński, "New trends in parallel and distributed evolutionary computing," *Fundamenta Informaticae*, vol. 35, no. 1-4, pp. 211–230, Jan. 1998.

[14] S. Boettcher and A. G. Percus, "Extremal optimization: Methods derived from co-evolution," in *Proceedings of GECCO 1999*, 1999, pp. 825–832.

[15] S. Boettcher and A. Percus, "Nature's way of optimizing," *Artificial Intelligence*, vol. 119, no. 1-2, pp. 275–286, 2000.

[16] S. Boettcher, "Extremal optimization: heuristics via coevolutionary avalanches," *Computing in Science and Engineering*, vol. 2, no. 6, pp. 75–82, 2000.

[17] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized criticality. an explanation of 1/f noise," *Physical Review Letters*, vol. 59, pp. 381–384, 1987.

[18] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[19] K. Tamura and H. Kitakami, "Island-model-based distributed modified extremal optimization with tabu lists for reducing crossovers in reconciliation graph," in *Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2014, 12-14 March, 2014, Hong Kong*, pp. 1–6.

[20] S. Hiroshi, O. Isao, and K. Shigenobu, "A new generation alternation model of genetic algorithms and its assessment," *J. of Japanese Society for Artificial Intelligence*, vol. 12, no. 5, pp. 734–744, 1997.

[21] S. Meshoul and M. Batouche, "Robust point correspondence for image registration using optimization with extremal dynamics," in *Proceedings of DAGM-Symposium 2002*, 2002, pp. 330–337.

[22] S. Boettcher and A. G. Percus, "Extremal optimization at the phase transition of the 3-coloring problem," *Physical Review E*, vol. 69, 066703, 2004.

[23] T. Zhou, W.-J. Bai, L.-J. Cheng, and B.-H. Wang, "Continuous extremal optimization for lennard-jones clusters," *Physical Review E*, vol. 72, 016702, 2005.

[24] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review E*, vol. 72, 027104, 2005.

[25] G.-q. Zeng, Y.-z. Lu, and W.-J. Mao, "Modified extremal optimization for the hard maximum satisfiability problem," *Journal of Zhejiang University SCIENCE C*, vol. 12, no. 7, pp. 589–596, 2011.

[26] X. Fu and J. Yu, "A hybrid algorithm based on extremal optimization with adaptive levy mutation and differential evolution and application," in *Proceedings of the 5th International Conference on Natural Computation*, ser. ICNC'09, 2009, pp. 12–16.