Distributed Modified Extremal Optimization for Reducing Crossovers in Reconciliation Graph

Keiichi Tamura, Hajime Kitakami, and Akihiro Nakada

***Abstract*—To determine the mechanism of molecular evolution, a reconciliation graph is constructed from two heterogeneous trees, which are referred to as 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. To carry out reconciliation work efficiently, it is necessary to find the state with the minimum number of crossovers of edges between leaf nodes. Reducing crossovers in a reconciliation graph is a combinatorial optimization problem. In this paper, we propose a novel bio-inspired algorithm called distributed modified extremal optimization (DMEO). This algorithm is a hybrid of population-based modified extremal optimization (PMEO) and the distributed genetic algorithm model that is used for reducing crossovers in a reconciliation graph. We have evaluated DMEO using actual data sets. DMEO shows better performance compared with PMEO.**

***Index Terms*—extremal optimization, distributed genetic algorithm, evolutionary computation, reconciliation graph**

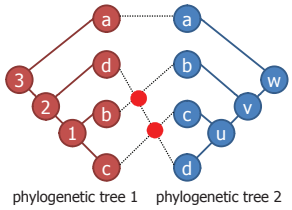
I. Introduction

MOLECULAR biologists need to carry out reconciliation work [1], [2], [3], [4] in order to determine the mechanism of molecular evolution. In reconciliation work, the relation between two heterogeneous phylogenetic trees and the relation between a phylogenetic tree and a taxonomic tree are compared. To compare two trees, we construct a graph called a reconciliation graph that consists of two phylogenetic trees or a phylogenetic tree and a taxonomic tree. Phylogenetic trees and taxonomic trees in a reconciliation graph are referred to as ordered trees. The leaf nodes of these ordered trees face each other. Moreover, leaf nodes with the same label name are connected to each other by an edge. To carry out reconciliation work efficiently, it is necessary to find the state with the minimum number of crossovers of edges between leaf nodes in the reconciliation graph.

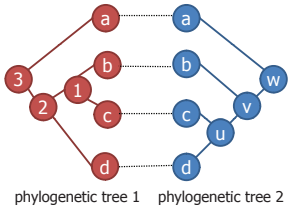
In Fig. 1, phylogenetic tree 1 and phylogenetic tree 2 are inferred from different molecular sequences with four identical species “a,” “b,” “c,” and “d.” The leaf nodes of phylogenetic tree 1 and those of phylogenetic tree 2 face each other. Leaf nodes representing the same species are connected to each other. The reconciliation graph shown in Fig. 1(a) has two crossovers. If we reduce crossovers in the reconciliation graph, we can obtain the reconciliation graph shown in Fig. 1(b), which has no crossovers.

Reducing crossovers in a reconciliation graph is a combinatorial optimization problem. The number of combinations increases exponentially as the number of leaf nodes increases. There are some heuristics [5], [6] that can be used for reducing crossovers in a reconciliation graph, and they use a genetic algorithm(GA)[7], extremal optimization (EO) [8], [9], [10], and modified EO (MEO)[11]. In our previous study[12], we proposed population-based modified extremal optimization (PMEO), which is a combination of a population-based approach and MEO.

K.Tamura, H.Kitakami, and A.Nakada are 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).



(a) Reconciliation graph with two crossovers.



(b) Reconciliation graph with no crossovers.

Fig. 1. Examples of reconciliation graphs ((a) shows a reconciliation graph that has two crossovers, and (b) shows a reconciliation graph that has no crossovers).

PMEO shows better performance compared with MEO. However, it is difficult to maintain diversity at the end of alternation of generations. To overcome this difficulty, this paper proposes a novel extremal optimization model called distributed modified extremal optimization (DMEO) for reducing crossovers in a reconciliation graph. DMEO is a hybrid of PMEO and the distributed genetic algorithm (DGA) model [13], [14]. In the DGA model, we divide a population into two or more sub-populations and each sub-population evolves individually. Therefore, DMEO can maintain diversity at the end of alternation of generations.

The main contributions of this study are as follows:

- Distributed modified extremal optimization (DMEO) is proposed. DMEO is a hybrid bio-inspired algorithm that combine PMEO and DGA. Many studies [8], [9], [10], [15], [16], [17], [18] have applied EO to combinatorial optimization problems such as the traveling salesman problem, graph partitioning problem, and image rasterization. Recently, some studies [19], [20] have focused on integrating a population-based approach in EO. To the best of our knowledge, there is no study on population-based EO involving the use of the distributed genetic algorithm model.
- To evaluate the proposed DMEO, we implemented

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

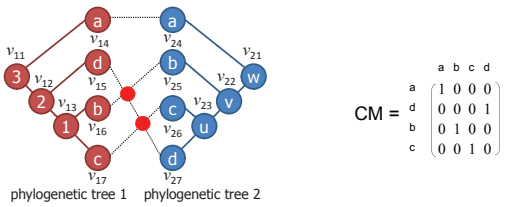


Fig. 2. Problem definition.

MDEO for reducing crossovers in a reconciliation graph. Moreover, we evaluated DMEO using two actual data sets for experiments. Experimental results shows that DMEO outperforms PMEO.

The rest of the paper is organized as follows. Section II presents the problem definition. Section III explains PMEO. Section IV proposes DMEO. Section V presents experimental results, and Section VI concludes the paper.

II. Problem Definition

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 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}.

In the reconciliation graph, OT_1 and OT_2 are located face to face. If a leaf node of OT_1 has the same label name as that of OT_2, then the two leaf nodes are connected to each other. In Fig.2, phylogenetic tree 1 is OT_1 and phylogenetic tree 2 is OT_2. The leaf node set L_1 has four nodes, v_{14}, v_{15}, v_{16}, and v_{17}. Similarly, L_2 has four nodes, v_{24}, v_{25}, v_{26}, and v_{27}. There are four label names in L_{leaf}, "a," "b," "c," and "d." Two leaf nodes v_{14} and v_{24} are connected because they have the same label name "a."

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 $C(M)$ returns the number of crossovers:

$$C(M) = \sum m_{j,\beta} m_{k,\alpha} [1 \le j < k \le n, 1 \le \alpha < \beta \le n], \quad (1)$$

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

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

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 $OL_2 = [v_{24}, v_{25}, v_{26}, v_{27}]$. Fig. 2 also shows M. For example, the $(0,0)$th-element $m_{0,0}$ is 1 because $\mathcal{L}(v_{14})$ equals $\mathcal{L}(v_{24})$. Similarly, the $(1,1)$th-element $m_{1,1}$ is 0 because $\mathcal{L}(v_{15})$ does not equal $\mathcal{L}(v_{25})$.

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

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

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. Population-based Modified Extremal Optimization

EO[8], [9], [10] 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. EO divides an individual I into n components O_i $(1 \le i \le n)$. Let λ_i be the fitness value of O_i. 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 of a component are repeated. 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.

Modified EO (MEO) [11] generates two or more neighbor individuals as candidates for the next generation individual. The best neighbor individual among the candidates is selected as the next generation individual. Moreover, MEO uses roulette selection to select a component. First, MEO selects $O_{selected}$ with roulette selection. 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 $O_{selected}$. Third, the generated I' is stored into $Candidates$. Finally, MEO selects the best individual from $Candidates$.

Population-based MEO (PMEO) [12] involves a population-based approach. There are two or more individuals in a population. Alternation of generation is repeatedly performed for every individual by using MEO. To improve the search efficiency, individuals copy a sub-structure of an individual that has good sub-structures at each alternation of generations. This operation resembles the crossover operation in genetic programing (GP). However, one side only copies a sub-structure of another side. Copying of good sub-structures leads to a high probability of generation of a good individual. As a result, efficient search can be performed by maintaining diversity.

IV. Distributed Modified Extremal Optimization

This section explains the main concept of DMEO and the algorithm of DMEO for reducing crossovers in a reconciliation graph.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A. Main Concept

DMEO is a hybrid of PMEO and DGA model. DMEO divides the entire population into two or more sub-populations. A sub-population evolves individually by PMEO. Moreover, from each sub-population, some individuals are selected and transferred to another sub-population. In return, the same number of migrants are received from another sub-population. DMEO repeats the following two steps:

(1) Sub-populations should be made to evolve through one or more generations by using PMEO.
(2) Some individuals of a sub-population are migrated to another sub-population.

Each sub-population evolves individually. Each sub-population converges to the separate best solution. Therefore, DMEO can maintain diversity at the end of alternation of generations.

B. Definition of Individual and Component

A reconciliation graph is defined as an individual. A component of an individual is defined as a pair of leaf nodes with the same label name:

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

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, $\mathcal{AS}(T, lname)$ is a set of ancestor nodes of a leaf node in T that has the label name $lname$.

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

$$\lambda_i = \frac{\mathcal{C}(M) - \mathcal{C}(M, i)}{\mathcal{C}(M)}, \tag{4}$$

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

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 $\lambda_4 = 3/4$.

C. Algorithm

The algorithm of DMEO for reducing crossovers in a reconciliation graph consists of two steps: (1) Evolution Step and (2) Migration Step (Algorithm 1). First, an initial population divided to p sub-populations (p is the number of sub-populations). In the Evolution Step (step 5), all sub-populations are made to evolve through m generations by using the function $\mathbf{PMEO}(SubP_i, m)$ (m is migration interval). In Migration Step (step 6), some individuals of a sub-population are migrated to another sub-population. Finally, the best individual from all sub-populations (step 7 and step 8).

Algorithm 1 $\mathbf{DMEO}$

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

Algorithm 2 $\mathbf{PMEO}(\mathbf{P}, \mathbf{m})$

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: $C \leftarrow C \cup \mathbf{GNI}(I, O_{selected})$
9: $n \leftarrow n + 1$
10: **end while**
11: $I \leftarrow \mathbf{BEST}(C)$
12: **end for**
13: $\mathbf{CSS}(P)$
14: **end for**

D. Evolution Step

In the Evolution Step, all sub-populations are made to evolve through m generations by using the function $\mathbf{PMEO}$ (Algorithm 2). First, for each individual, the state of the individuals in P is changed by using MEO. Second, the function $\mathbf{CSS}$ copies a good sub-structure of an individual to another individual.

In the MEO steps, for each individual, the following steps are executed. Initially, the function evaluates the fitness value λ_i (step 3). 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, the function generates an neighbor individual from I with the function $\mathbf{GNI}$. The function $\mathbf{GNI}$ generates a neighbor individual by changing the state of component $O_{selected}$. Third, the neighbor individual is stored in C (step 8). Finally, the best individual in C is selected and I is replaced by it (step 11).

The state of $O_{selected}$ is changed by changing the order of child nodes in an intermediate node that is an ancestor node of $O_{selected}$. The processing steps of $\mathbf{GNI}$ are as follows. First, the outputs of $\mathcal{AS}(T_1, \mathcal{L}(O_{selected}))$ or $\mathcal{AS}(T_2, \mathcal{L}(O_{selected}))$ are stored in the set $Ancestors$. Then,

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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

Algorithm 3 $\mathbf{CSS}(\mathbf{P})$

1: Select individual $SI \in P$ by roulette selection (selection rates are the fitness values of components).
2: **for all** $I \in P, I \neq SI$ **do**
3: **for** $i = 1$ to n **do**
4: 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.
5: **end for**
6: Select $O_{selected}$ by roulette selection (selection rates are $diff_i$).
7: $A \leftarrow \mathcal{AS}(T_1, \mathcal{L}(O_{selected}))$ or $\mathcal{AS}(T_2, \mathcal{L}(O_{selected}))$
8: $C \leftarrow \phi$
9: **for all** $a \in A$ **do**
10: Generate a new individual I' from I by changing the order of child nodes in a.
11: $C \leftarrow C \cup I'$
12: **end for**
13: $I \leftarrow \mathbf{BEST}(C)$
14: **end for**

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 $\mathcal{AS}(T_1, \mathcal{L}(O_2))$ returns $\{v_{12}, v_{11}\}$ and $\mathcal{AS}(T_2, \mathcal{L}(L_2))$ 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 of state.

Algorithm 3 shows the function **CSS**. At the beginning, an individual SI in P is selected by roulette selection (step 1). Each individual of P 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 3, 4, and 5). Second, $O_{selected}$ is selected by roulette selection (step 6). Next, $\mathcal{AS}(T_1, \mathcal{L}(O_{selected}))$ or $\mathcal{AS}(T_2, \mathcal{L}(O_{selected}))$ is stored in A (step 7). 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 9, 10, 11, and 12). Finally, the function selects the best individual from C (step 13).

E. Migration Step

In the Migration Step, some individuals of a sub-population are migrated to another sub-population. The distributed genetic algorithm model requires $number\ of\ sub-populations$, $migration\ rate$, $migration\ interval$, and $migration\ model$. The first three items are user-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 $selection\ method$. Moreover, the proposed algorithm uses the random ring migration topology. The most basic migration topology is the ring migration topology. In this topology, individuals are transferred between directionally adjacent sub-populations. In the random ring migration topology, an arrival sub-population to which individuals are to be migrated is decided at random.

V. PERFORMANCE EVALUATION

We performed four experiments for evaluating the performance of DMEO. In the experiments, the two data sets listed in Table I are 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.

Experiment 1 measured the number of crossovers of the best individual at each generation to compare DMEO and PMEO. Experiment 2 also measured the number of crossovers of the best individual at each elapsed time to compare DMEO and PMEO. Experiment 3 measured frequency of the number of crossovers of best individuals in fixed generations. Experiment 4 measured the number of crossovers of the best individual at each generation by changing the number of sub-populations.

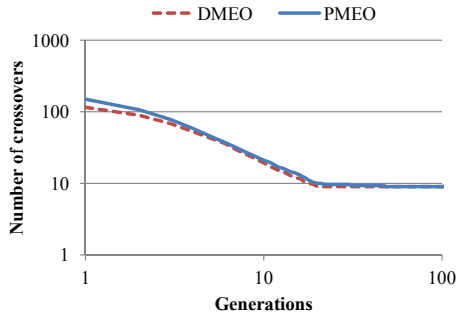
In PMEO and DMEO, the number of individuals in the population was set to 100. The user parameter $num\_of\_candidates$ was set to 100 and m was set to 10000 in PMEO. In DMEO, the user parameter $num\_of\_candidates$, $migration\_interval(m)$, $number\ of\ sub-populations(p)$, and $migration\ rate$ were set to be 100, 10, 5 and 0.05, respectively. The number of individuals in a sub-population is 20. The number of crossovers was the average of three trials.

Experiment 1

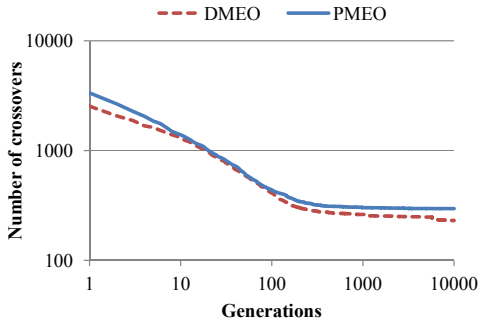
In Experiment 1, we measured the number of crossovers of the best individual in each generation. Figure3(a) and Figure3(b) show the number of crossovers (vertical axis: the number of crossovers, horizontal axis: generations). Fig. 3(a) and Fig. 3(b) show that the number of crossovers of DMEO in each generation was smaller than that in the case of PMEO. DMEO showed better performance compared with PMEO.

The number of crossovers in PMEO is converging into around 300 when we use $Moss$ data set. On the other hand, in DMEO, the number of crossovers is converging into around 250. The diversity of PMEO is small, because the number of sub-populations is one. Therefore, the fitness

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

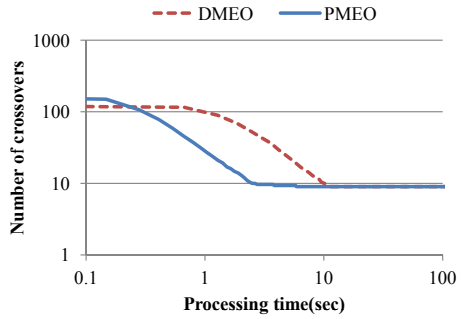


(a) *Housekeeping* data set

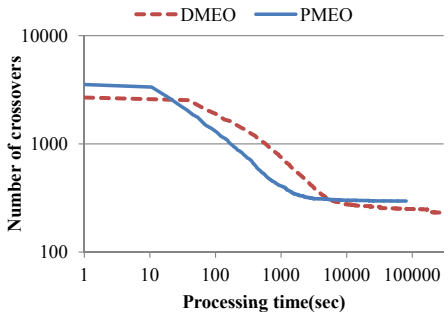


(b) *Moss* data set

Fig. 3. Experiment 1 ((a) and (b) shows the number of crossovers of the best individual) .



(a) *Housekeeping* data set



(b) *Moss* data set

Fig. 4. Experiment 2 ((a) and (b) shows the number of crossovers of the best individual).

value of a individual will not be improved in the end of alternation of generations.

Experiment 2

In Experiment 2, we measured the number of crossovers of the best individual at different time instants. The computation time of DMEO was longer than that in the case of PMEO because the former included the Migration Step. Therefore, it was necessary to compare the number of crossovers for the same computation time.

Fig. 4(a) and Fig. 4(b) show the number of crossovers at different time instants (vertical axis: the number of crossovers, horizontal axis: processing time). The number of crossovers of DMEO becomes smaller than that of PMEO at the end of alternation of generations.

Experiment 3

The number of crossovers of the best individual was measured 100 times for the 10,000th alternation generation. Figure5(a) and Figure5(b) show frequency of the number of crossovers when *Housekeeping* data set is used. The number of crossovers of the optimal solution of *Housekeeping* data set is 9. Both of them can obtain the best solution by 100%. Fig. 6(a) and Fig. 6(b) show the frequency of the number of crossovers for the *Moss* data set. In DMEO, all the numbers of crossovers of optimal solutions were between 200 and 299. On the other hand, they were distributed between 200 and 400 for PMEO. Above all, although 90% of optimal solutions were between 200 and 249 in the case of DMEO, only a few optimal solutions were obtained between 200 and 249 in the case of MEO.

Experiment 4

In Experiment 4, we changed the number of sub-populations in DMEO. Fig. 7 shows the results of Experiment 4 using *Moss* data set. When the number of sub-populatins is four, it has fallen into the local optimal solutions. On the other hand, when the number of sub-pouplations is five or ten, convergence is not early. Therefore, they can obtain better solutions.

VI. Conclusion

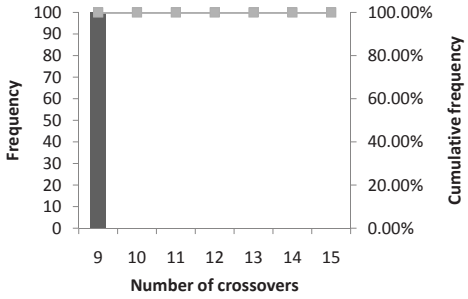
This paper proposes distributed modified extremal optimization (DMEO) for reducing crossovers in a reconciliation graph. The proposed algorithm is a bio-inspired algorithm of population-based modified extremal optimization (PMEO) and the distributed genetic algorithm model. We have evaluated DMEO by using actual data sets. Experimental results show that DMEO is better performance compared with PMEO. In the future work, we will develop extended DMEO for making it applicable to other combination optimization problems.

Acknowledgment

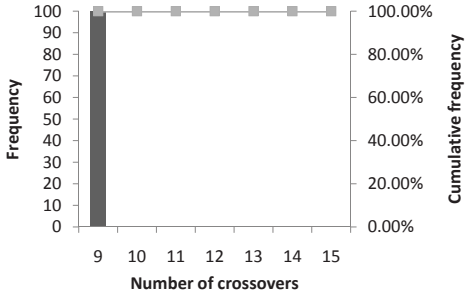
This work was supported in part by a Grant-in-Aid for Young Research (B) (No.23700124) from the Ministry of Education, Culture, Sports, Science and Technology in Japan and a Grant-in-Aid for Scientific Research (C) (2) (No.20500137) from the Japanese Society for the Promotion of Science, Japan.

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.

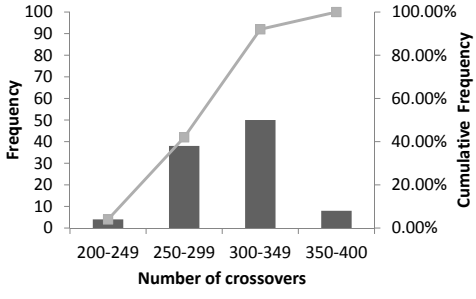


(a) PMEO

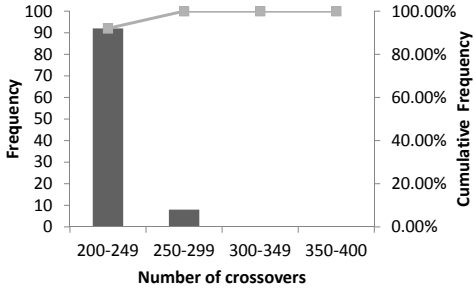


(b) DMEO

Fig. 5. Experiment 3 ($Housekeeping$ data set).



(a) PMEO



(b) DMEO

Fig. 6. Experiment 3 ($Moss$ data set).

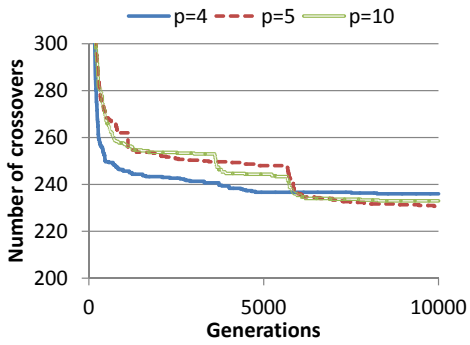


Fig. 7. Experiment 4 ($Moss$ data set).

[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] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

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

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

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

[11] 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.

[12] 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.

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

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

[15] 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.

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

[17] 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.

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

[19] Y. Chen, K. Zhang, and X. Zou, "A population-based hybrid extremal optimization algorithm," in *Proceedings of the 7th international conference on Intelligent Computing: bio-inspired computing and applications*, 2011, pp. 410–417.

[20] M.-R. Chen, Y.-Z. Lu, and G. Yang, "Multiobjective optimization using population-based extremal optimization," *Neural Comput. Appl.*, vol. 17, no. 2, pp. 101–109, 2008.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)