sciendo

JAISCR, 2019, Vol. 9, No. 1, pp. 51 – 65
10.2478/jaiscr-2018-0024

ENHANCING ISLAND MODEL GENETIC PROGRAMMING BY CONTROLLING FREQUENT TREES

Keiko Ono[1,3], Yoshiko Hanada[2], Masahito Kumano[3], Masahiro Kimura[3]

[1]*Department of Computer Engineering, Ryukoku University, Shiga 5202194, Japan*
[2]*Faculty of Engineering Science, Kansai University, Japan*
[3]*Department of Electronics and Informatics, Ryukoku University, Shiga 5202194, Japan*

E-mail: hanada@kansai-u.ac.jp, {kumano, kimura, kono}@rins.ryukoku.ac.jp

Submitted: 27th December 2017; Accepted: 11th January 2018

Abstract

In evolutionary computation approaches such as genetic programming (GP), preventing premature convergence to local minima is known to improve performance. As with other evolutionary computation methods, it can be difficult to construct an effective search bias in GP that avoids local minima. In particular, it is difficult to determine which features are the most suitable for the search bias, because GP solutions are expressed in terms of trees and have multiple features. A common approach intended to local minima is known as the Island Model. This model generates multiple populations to encourage a global search and enhance genetic diversity. To improve the Island Model in the framework of GP, we propose a novel technique using a migration strategy based on textit *f*requent trees and a local search, where the frequent trees refer to subtrees that appear multiple times among the individuals in the island. The proposed method evaluates each island by measuring its *a*ctivation level in terms of the fitness value and how many types of frequent trees have been created. Several individuals are then migrated from an island with a high activation level to an island with a low activation level, and vice versa. The proposed method also combines strong partial solutions given by a local search. Using six kinds of benchmark problems widely adopted in the literature, we demonstrate that the incorporation of frequent tree information into a migration strategy and local search effectively improves performance. The proposed method is shown to significantly outperform both a typical Island Model GP and the aged layered population structure method.

Keywords: genetic programming, island model, frequent tree-based migration strategy

1 Introduction

Genetic Programming (GP) is the best-known evolutionary computation (EC) algorithm whose solutions are expressed in terms of a tree. GP is one of the most widely used and practical optimization methods and offers high performance and availability[1]. GP has been used to produce solutions that are competitive with those developed by humans, Hence, numerous researchers have applied GP to a wide variety of fields, including electrical circuits, robotics, and image filters [2, 3]. However, it remains important to explore new application areas and to develop new algorithms with better performance and availability.

Similar to other EC techniques, the most important GP issue concerns the development of a versatile algorithm to prevent premature convergence. For example, genetic algorithms (GAs) often use

spatial information about the solutions to avoid local minima. More specifically, a solution that is represented by a bit string can be projected in a continuous design variable space using methods such as binary coding or gray coding, thus allowing various bias-control methods using spatial information to be employed. These methods can control the search bias according to the landscape of an objective function, resulting in an effective search. In GP, solutions are expressed in terms of trees, so special information cannot be used to control the search bias, because there is no way to map individual solutions within a design variable space directly. Therefore, to prevent a population from converging prematurely in GP, parallel models such as an Island Model GP must be applied [4, 5].

In the Island Model [6], individuals are divided into some islands, and selected individuals migrate from one island to the next. In this way, the Island Model GP retains diversity in an attempt to avoid local minima. Studies on the Island Model GP have resulted in further performance improvement via the following three approaches: changing the "number of islands", changing the "migration rate", and exploring more efficient migration strategies. Andre and Koza [7], Punch [8], and others [9, 10] have investigated the effects of changing the number of islands and the migration rate using several benchmark problems; however, it is difficult to construct a universal theory that can determine optimal parameter values for a given problem. Various other migration strategies have been proposed. For example, Hu et al. [11] developed the hierarchical fair competition (HFC) Island Model GP. HFC organizes islands based on the *f*itness level to ensure fair competitions between individuals. Variants such as adaptive HFC [12] and continuous HFC [13] have also been reported. Hornby introduced the *a*ge feature and proposed the age-layered population structure (ALPS)[14, 15, 16]. ALPS evaluates how long each's genetic material has been evolving. Korns conducted that ALPS was a state-of-the-art island model in GP[17], and found that it performed well in symbolic regression problems.

According to the Building Block Hypothesis in evolutionary computation [18, 19, 20], partial solutions known as building blocks can be assembled into the entire solutions, and a greater number of partial solutions can improve performance. In particular, partial solutions are considered to play an important role in improving GP performance, because each solution is a tree. As we mentioned above, various Island Model GPs have been proposed. However, these models only use fitness to organize islands and define the similarity between individuals or islands By developing an effective migration strategy based on the tree information of the individuals, we attempt to enhance the Island Model GP. In this paper[1], we propose an effective migration strategy that enhances the combination of partial solutions based on the tree information.

In general, it is difficult to specify partial solutions during the search for an optimal solution. Thus, we assume that frequent trees on an island of a later generation are closely related to partial solutions in the framework of the Island Model GP. Here, the frequent trees on an island refer to the subtrees that frequently appear among the individuals on the island. In this paper, we propose to evaluate the activation level of an island on the basis of both fitness and frequent trees, and present a novel migration strategy based on the activation level and a local search. To the best of our knowledge, this is the first attempt to use the tree information of individuals in the Island Model GP.

The remainder of this paper is organized as follows. In Section 2, we formulate the optimization problem discussed in this paper, and give a detailed explanation of the Island Model GP with a random topology as a typical method of the Island Model GP. In Section 3, we describe the proposed migration strategy with a local search method in detail. Section 4 is dedicated to examining the effectiveness of the proposed method via experiments using three types of benchmark problems widely adopted in the literature. Here, we introduce two methods as extreme cases of the proposed method to separately analyze the effects of fitness and frequent trees in the proposed evaluation measure. We present the results of a comparison between the pro-

[1] In our conference paper [21], we proposed a basic framework for such a migration strategy. This paper extends our preliminary paper [21] by improving how to measure the activation level of an Island and newly incorporating a local search mechanism. Moreover, we extensively discuss related work and compared the proposed method with the state-of-the-art method, ALPS, for six kinds of benchmark problems.

posed method, the two methods, ALPS, and conventional Island Model GP with a random topology in solution quality. Moreover, we extensively compare the proposed method and the Island Model GP with a random topology in terms of their computational time and solution quality. Finally, we give our conclusions from this study and some ideas for future work in Section 5.

2 Problem Definition and Island Model Genetic Programming

In the framework of genetic programming (GP), we consider the problem of maximizing a function $f(x)$, where the input variable x is represented as a labelled ordered tree. Given the total population size N and a crossover-mutation-selection strategy, a common approach to improving genetic diversity in GP is to apply the Island Model methods. In the Island Model GP, the total population is partitioned into M subpopulations, or islands, and the islands execute GP searches to maximize $f(x)$ in parallel, although they exchange information by migration. There are several migration topologies for the Island Model, but the random topology exhibits the best performance. Thus, we focus on a typical method of the Island Model with a random topology in the framework of GP, and refer to this as the original Island Model GP. In this paper, we attempt to improve the original Island Model GP for maximizing $f(x)$.

We begin by recalling the original Island Model GP. Let L_m ($m = 1, \ldots, M$) denote the set of individuals in the mth island, where $|L_m| = N/M$, and note that the total population L is obtained by

$$L = \bigcup_{m=1}^{M} L_m,$$

and consists of labelled ordered trees. We denote by $\mathcal{L}$ the set of the islands, i.e.,

$$\mathcal{L} = \{L_m;\ m = 1, \ldots, M\}.$$

Let $B_m(n)$ and $W_m(n)$ denote the sets of the best n individuals and the worst n individuals on an island L_m, respectively, with respect to their objective function values, where n is a positive integer.

The migration strategy of the original Island Model GP is as follows:

O0: Set $k \leftarrow 1$.

O1: Set $\mathcal{L}_k \leftarrow \mathcal{L}$.

O2: If $k \leq M/2$, then perform Steps O3 to O9, otherwise, stop.

O3: Choose two islands $L_{m'}$ and $L_{m''}$ at random from $\mathcal{L}_k$.

O4: Choose the best n individuals $B_{m'}(n)$ and the worst n individuals $W_{m'}(n)$ from $L_{m'}$.

O5: Choose the best n individuals $B_{m''}(n)$ and the worst n individuals $W_{m''}(n)$ from $L_{m''}$.

O6: Replace $W_{m'}(n)$ with a copy of $B_{m''}(n)$, and replace $W_{m''}(n)$ with a copy of $B_{m'}(n)$.

O7: Set $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \setminus \{L_{m'}, L_{m''}\}$.

O8: Set $k \leftarrow k + 1$.

O9: Return to Step O2.

The migration occurs at migration time αj ($j = 1, \ldots, J$), where α is a positive integer called the *m*igration interval, and J is an integer with $1 \leq J \leq I/\alpha$. Note that α and n are integer-valued parameters. We used $n = N/(2M)$ in our experiments.

3 Proposed Method

We now consider enhancing the Island Model GP for the problem of maximizing the objective function $f(x)$.

According to the Building Block Hypothesis in EC, it is considered that partial solutions can be assembled into the entire solution. In GP, solutions are expressed by trees, so partial solutions deeply affect performance improvement. Therefore, we hypothesize that there exist *g*ood pieces (sub-trees) as solution seeds, and that each partial solution is constructed by combining some of these pieces in an optimal manner. We also assume that if these pieces are badly incorporated in an individual, they cannot build even partial solutions. In the Island Model GP, each island executes its GP search to maximize $f(x)$, resulting in individuals with high objective function values. Viewed in this light, we consider that the *f*requent trees, namely, those subtrees that frequently appear among the individuals on an island, to be candidates for the good pieces (i.e., solution pieces) in later generations. Thus, we suppose that *i*slands with higher activation levels not only generate individuals (labelled ordered

trees) having higher objective function values, but also newly create more types of frequent trees. We now explain the details of the proposed method.

A labeled ordered tree is a 6-tuple $t = (V_t, E_t, LA_t, \ell a_t, v_t^0, \preceq_t)$, where V_t is the set of nodes, E_t is the set of links, LA_t is the set of labels, $\ell a_t : V_t \to LA_t$ is the labeling function, v_t^0 is the root, and $\preceq_t$ is the sibling relation. For labeled ordered trees t and x, t is called a *s*ub-tree of x when there exists a one-to-one map $\varphi : V_t \to V_x$ such that φ preserves the parent relation (i.e., $(v_1, v_2) \in E_t$ iff $(\varphi(v_1), \varphi(v_2)) \in E_x$), the sibling relation (i.e., $v_1 \preceq_t v_2$ iff $\varphi(v_1) \preceq_x \varphi(v_2)$), and the labels (i.e., $\ell a_t(v) = \ell a_x(\varphi(v))$). For an island L_m, we define a labeled ordered tree t to be its *f*requent tree when the number of individuals that include t as their sub-trees is more than $\lambda|L_m|$, where λ is a positive number.[2] Let T_m denote the set of all frequent trees on island L_m. Note that one of the main aims of migration is to find solution pieces (sub-trees of relatively small sizes), and obtaining T_m is computationally expensive. Thus, we restrict our attention to a subset T_m^β of T_m such that the sizes of sub-trees belonging to T_m^β are relatively small and $|T_m^\beta| = \beta|L_m|$, where β is a positive number.[3] We construct T_m^β by extracting all frequent trees of size k in L_m in order of increasing size k until satisfying $|T_m^\beta| = \beta|L_m|$, where k is an integer with $k \geq 2$. Here, T_m^β is referred to as the *s*et of frequent trees of rate β in L_m. In our migration strategy, we particularly focus on how many types of frequent trees increase on an island. Taking account of our main aim of migration and the computational cost, we deal with T_m^β instead of T_m. Furthermore, we also use FREQT [22, 23] to reduce the computational cost. FREQT efficiently enumerates the frequent trees from a set of labelled ordered trees by using the "rightmost expansion" and "pruning by node-skip and edge-skip" techniques. The algorithm is very efficient and scales almost linearly with the total size of the maximal frequent trees. Therefore, we attempt to efficiently measure the activation level of an island by exploiting a suitable approximation and FREQT.

Let $\hat{x}_m \in L_m$ be the individual that gives the maximum value of $f(x)$ in L_m, i.e.,

$$f(\hat{x}_m) \geq f(x) \quad (\forall x \in L_m).$$

We define $g(L_m)$ to be the number of new types of frequent trees of (αj)th generation in L_m, i.e.,

$$g(L_m) = \left|T_m^\beta(\alpha j)\right| - \left|T_m^\beta(\alpha j) \cap T_m^\beta(\alpha(j-1))\right|,$$

where $T_m^\beta(\alpha j)$ is the set of frequent tree of rate β of (αj)th generation. In order to evaluate each island L_m by a combination of f-evaluation $f(\hat{x}_m)$ and g-evaluation $g(L_m)$, we introduce a measure $F(L_m)$ such that

$$0 \leq F(L_m) \leq 1, \tag{1}$$

$$f(\hat{x}_m) < f(\hat{x}_{m'}) \Rightarrow F(L_m) < F(L_{m'}), \tag{2}$$

$$g(L_m) < g(L_{m'}) \Rightarrow F(L_m) < F(L_{m'}). \tag{3}$$

Although there are many ways to design the evaluation measure $F(L_m)$ of island L_m,[4] we employ a ranking based method in where a good island $L_{m'}$ and a bad island $L_{m''}$ are deterministically chosen from $\mathcal{L}$, that is,

$$L_{m'} = \operatorname*{argmax}_{L_m \in \mathcal{L}} F(L_m),$$

$$L_{m''} = \operatorname*{argmax}_{L_m \in \mathcal{L}} (1 - F(L_m)).$$

We constructed the evaluation measure $F(L_m)$ in the following way. Let v_m^f denote the normalized maximum fitness value of an island L_m in $\mathcal{L}$ with respect to f-evaluation. Namely,

$$v_m^f = \frac{f(\hat{x}_m)}{\sum_m f(\hat{x}_m)}. \tag{4}$$

Let v_m^g denote the normalized $g(L_m)$ of an island L_m, that is,

$$v_m^g = \frac{g(L_m)}{\sum_m g(L_m)}. \tag{5}$$

Then, we define $F(L_m)$ by

$$F(L_m) = \phi v_m^f + (1 - \phi) v_m^g,^5 \tag{6}$$

where ϕ is a parameter to control the balance between v_m^f and v_m^g. It is easily proved that the conditions (1), (2) and (3) are satisfied. Note that $F(L_m)$ defined by Equation (6) equally treats the effects of $f(\hat{x}_m)$ and $g(L_m)$.

Using a *D*ivide & Conquer technique under evaluation measure $F(L_m)$, the proposed method

[2] λ is a parameter. We used $\lambda = 0.5$ in our experiments.

[3] β is a parameter. We used $\beta = 0.5$ in our experiments.

[4] Our future work includes extensively examining various types of evaluation measure $F(L_m)$.

[5] In our experiments, we used $\phi = 0.5$.

migrates n individuals from an island with a high activation level (called a *g*ood island) to an island with a low activation level (called a *b*ad island), and vice versa, to enhance genetic diversity. The algorithm of the proposed method is as follows:

P0: Generate the initial individuals.

P1: Divide these individuals into M subpopulations.
/*Crossover in each subpopulation L_m*/

P2: Set $j \leftarrow 1$

P3: If $j \leq |L_m|$, then perform Steps P3 to P13, otherwise perform Step P14.

P4: Choose a mother e_m with probability proportional to fitness from L_m[6].

P5: Choose a father e_f with probability proportional to fitness

P6: Crossover between e_m and e_f and generate a child candidate c'_ℓ.
/*Mutation (local search) */

P7: Set $\ell \leftarrow 1$.

P8: Set $c'_\ell \leftarrow e_m$.

P9: If $\ell \leq V$, then perform Step P10 to P12, otherwise perform Step P13[7] .

P10: Mutate c'_ℓ.

P11: Set $\ell \leftarrow \ell + 1$.

P12: Select the best candidate c'_ℓ as a child c_j.

P13: Set $j \leftarrow j + 1$.
/*Migration*/

P14: Set $k \leftarrow 1$.

P15: Set $\mathcal{L}_k \leftarrow \mathcal{L}$

P16: If $k \leq M/2$, then perform Steps P17 to P22, otherwise, stop.

P17: Choose $L_{m'}$ from $\mathcal{L}_k$ with probability proportional to evaluation measure $F(L_m)$, and choose $L_{m''}$ from $\mathcal{L}_k$ with probability proportional to $1 - F(L_m)$.[8]

P18: Choose a set $X_{m'}(n)$ of n individuals from $L_{m'}$ at random.

P19: Choose a set $X_{m''}(n)$ of n individuals from $L_{m''}$ at random.

P20: Move $X_{m'}(n)$from $L_{m'}$ to $L_{m''}$, and move $X_{m''}(n)$ from $L_{m''}$ to $L_{m'}$.

P21: Set $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \setminus \{L_{m'}, L_{m''}\}$.

P22: Set $k \leftarrow k + 1$.

Similar to the original Island Model GP, the migration occurs at migration time αj ($j = 1, \ldots, J$), where α is an integer-valued parameter called the *m*igration interval, and J is an integer with $1 \leq J \leq I/\alpha$. In our experiments, we also used $n = N/(2M)$.

The proposed method aims to enhance genetic diversity more than in the original Island Model GP to avoid premature convergence. Maintaining diversity is closely related to a migration strategy in the Island Model. In particular, we use a novel evaluation measure for migration in the framework of GP, and combine it with a simple Divide & Conquer technique. Several studies have examined Divide & Conquer techniques for the purpose of maintaining diversity and avoiding premature convergence. For example, Fillon et al [24] developed an "island rank" based technique, and Luke et al [25] reported a "diagonal layout" approach. We note that such techniques can be easily incorporated into the proposed method. Moreover, the proposed method includes a local search to enhance the exploration of partial solutions. In the proposed method, individuals on islands with higher activation levels are migrated to islands with lower activation levels to increase whole activation levels. It leads to increase the genetic diversity of individuals in consequence. However, it is known that there are trade-offs between the speed of convergence and preservation of genetic diversity. To overcome these problems, in this paper, we introduce a new real-valued based metric based on frequent trees and apply a local search to maintain a balance between global search and local search. The migration strategy in the proposed method encourages more new kinds of frequent trees on each island L_m, and local search helps to combine useful frequent trees into a solution as candidates of partial solutions.

[6] We used a fitness tournament selection in our experiments

[7] In our experiments, we also used $V = 2$.

[8] We can design various roulette wheels for selecting both good and bad islands. Here, we state the most fundamental one.

4 Experimental Evaluation

By conducting experiments on three benchmark problems that are widely used in the literature, we evaluated the effectiveness of the proposed method.

4.1 Benchmark Problems

We evaluated the effectiveness of the proposed method on well-known benchmark problems, the Santa Fe ant trail problem and the symbolic regression problem [26, 27, 28] for more detailed descriptions). We used these problems to evaluate the proposed measures v_m^f, v_m^g and the local search. In particular, in symbolic regression problems, changes in subtrees that are positioned deep inside in a tree rarely affect the overall fitness value, and in royal tree problems, fitness values are significantly affected by small changes in a tree regardless of their position. On the other hand, the Santa Fe ant trail problem is moderately affected by subtree changes.

4.1.1 Santa Fe Ant Trail Problem

We examined the Santa Fe ant trail problem composed of 144 squares with 21 turns and 89 food units, as shown in Figure 1. Artificial ants can perform three operations (Move, Left and Right) to find the food units. The fitness function $f(x)$ counts the number of food units found by the ant. The total number of operations was limited to 400 in our experiments.

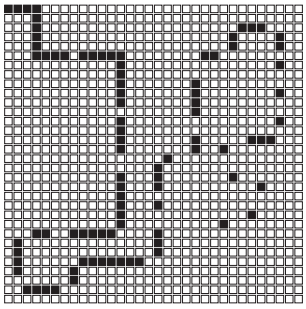


Figure 1. Santa Fe ant trail problem. Filled squares denote food units.

4.1.2 Royal Tree Problem

The royal tree problem [29] is composed of function nodes and terminal nodes. We examined the D-level royal tree problem, which contains function nodes $\{A, B, C, D\}$ and terminal nodes $\{t_x, t_y, t_z\}$, and the E-level royal tree problem, which consists of function nodes are $\{A, B, C, D, E\}$ and terminal nodes are$\{t_x, t_y, t_z\}$, respectively. These problems search for the entire perfect tree, which is composed of smaller perfect trees. Figure 2 shows perfect trees at each level, where each perfect tree has a score value. The fitness function $f(x)$ is evaluated by summing each score of a perfect tree within a tree structure (solution), and the total score is doubled when the tree is an optimal solution. It is known that the optimal value of $f(x)$ is 6144 in the D-level problem, and $f(x)$ is 144880 in the E-level problem.

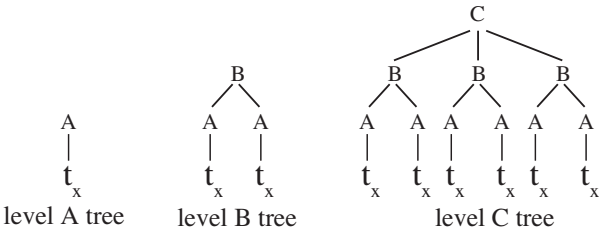


Figure 2. A Perfect level-A, level-B, and level-C royal tree.

4.1.3 Symbolic Regression Problem

We investigated the symbolic regression problem for the function space $\mathcal{X}$ constructed by the labeled ordered trees of functional nodes {+, −, ×, /, sin, cos} and terminal nodes {s, 0.05, 0.10, 0.15, 0.20, …, 0.95, 1.00}, where s is a variable value. Our training set was composed of 30 data points $\{(s_j, x_*(s_j)) \in \mathbf{R}^2 \ ; \ j = 1, \ldots, 30\}$, where $s_j = 0.2(j-1)$, and $x_*(s) \in \mathcal{X}$ is the true function to be identified. For any $x(s) \in \mathcal{X}$, we define the fitness $f(x)$ as

$$f(x) = 1000 - 50 \sum_{j=1}^{30} |x(s_j) - x_*(s_j)|,$$

and consider the problem of maximizing $f(x)$. In our experiments, we employed two functions (Function 1 to 2) from [26] and one function (Function 3) from [30] for $x_*(s)$,

$$\begin{aligned}
\text{F1}: \quad x_*(s) &= (2 - 0.3s)\sin(2s)\cos(3s) + 0.11s^2, \\
\text{F2}: \quad x_*(s) &= s\cos(s)\sin(s)\left(\sin^2(s)\cos(s) - 1\right), \\
\text{F3}: \quad x_*(s) &= s^8 + 3\cos(s) + s^5 + s.
\end{aligned}$$

4.2 Comparison Methods

The proposed method evaluates the activation level of each island $L(m)$ for migration as a combination of f-evaluation $f\left(x_f(m)\right)$ (i.e., how high the best fitness value is) and g-evaluation $g(L(m))$ (i.e., how many types of frequent trees have been created). To analyze the effects of f-evaluation and g-evaluation separately, we first introduce two methods as extreme cases of the proposed method, and compare their solution quality in Section 4.4. Moreover, to analyze the effects of the proposed method, we also compare the performance of ALPS and the simple Island approach (see Section 4.2.5 for more detailed description) in terms of solution quality in Section 4.4.

4.2.1 f-evaluation

In the same way as for the implantation of the proposed method, a good island $L_{m'}$ and a bad island $L_{m''}$ are deterministically chosen from $\mathcal{L}^k$ in Step P3. First, we investigate the f-evaluation-based method in which the evaluation measure $F(L_m)$ of island L_m is defined by

$$F(L_m) = v_m^f. \tag{7}$$

This is referred to as the f-evaluation method. Note that the conditions (1) and (2) are satisfied for the evaluation measure $F(L_m)$ defined by Equation (7).

4.2.2 g-evaluation

Next, we investigate the g-evaluation based method in which the evaluation measure $F(L_m)$ of island L_m is defined by

$$F(L_m) = v_m^g. \tag{8}$$

This is referred to as the g-evaluation method. Note that the conditions (1) and (3) are satisfied for the evaluation measure $F(L_m)$ defined by Equation (8).

4.2.3 Ranking-based evaluation

The proposed method enables adaptive migration based on the real-valued f-evaluation and, g-evaluation given in Equation(4) and (5). As a comparison method, we introduce the rank-based evaluation in terms of the evaluation measure function $F(L_m)$ of the proposed method to analyze which is more suitable, real-value based or rank based evaluations.

Let r_m^f denote the rank of an island L_m in $\mathcal{L}$ with respect to f-evaluation. Namely, when L_m is ranked in $\mathcal{L}$ according to $f(\hat{x}_m)$, L_m is the r_m^fth rank in $\mathcal{L}$. Let r_m^g denote the rank of an island L_m in $\mathcal{L}$ with respect to g-evaluation. Namely, when L_m is ranked in $\mathcal{L}$ according to $g(L_m)$, L_m is the r_m^gth rank in $\mathcal{L}$. Then, we define $F(L_m)$ by

$$F(L_m) = \frac{\left(M - r_m^f\right) + \left(M - r_m^g\right)}{2(M-1)}. \tag{9}$$

For example, if L_m is the top-ranked island with respect to f-evaluation and the top-ranked island with respect to g-evaluation. We refer to this method as the *r*anking-based method in this paper.

4.2.4 ALPS

The proposed method of adaptively controlling the migration between islands using frequent trees is an enhanced version of the Island Model GP. In our experiments, we compare the proposed method with the ALPS method, which is a current state-of-the-art Island Model GP for symbolic regression problems.

The ALPS has an assembly line structure based on the *a*ge of each individual, where the age indicates that how long the individual's genetic material has been evolving within the population. The age is measured by the number of times an individual has been selected as a parent. To ensure fair competitions and restrict breeding among individuals, each layer has a maximum age, and individuals can only breed in their layer or the layer one level below. If an individual is selected to reproduce multiple times, its age only increases by 1 in one generation, and the individual moves to the highest layer possible. In the lowest layer, individuals are renewed periodically at random. HFC, which uses the same fair competition model as ALPS separates individuals into different layers based on their fitness values and fair competitions among individuals are achieved in terms of fitness. However, HFC tends to push individuals toward the same local minima, because individuals with similar fitness values often inhabit the nearby region. Thus, ALPS introduced the age attribute to enhance genetic diversity. We briefly explain the ALPS. Let $x_m^n \in L_m$ be an individual, and $age(x^n)$ be the age of x^n, and δ be an integer-valued parameter that indicates the interval between generations for reconstructing the popula-

tion according to age.

A0: Set $k \leftarrow 1$ and $L_k \leftarrow \emptyset$.

A1: If k is the generation in which the population is to be reconstructed, then perform Steps A2 to A15.

A2: If $L_k \leftarrow \emptyset$, then generate initial individuals on the island L_k (Level k) and set $age(x_k^n) \leftarrow 1$.

A3: If $|L_k| < |L_m|$, then generate $|L_m| - |L_k|$ individuals in the Level k island L_k and set $age(x_k^n) \leftarrow 1$.

A4: Set $j \leftarrow 1$.

A5: If $j \leq |L_m|$, then perform Steps A4 to A9, and otherwise perform Step A10.

A6: Select an individual x_m^n according to the proportion of fitness values in L_m.

A7: Set $flag(x_m^n) \leftarrow 1$.

A8: Apply genetic operators such as crossover and mutation.

A9: Set $j \leftarrow j+1$.

A10: Set $j \leftarrow 1$.

A11: If $j \leq |L_m|$, then perform Steps A12 to A14, and otherwise perform Step A15.

A12: If $flag(x_m^j)$ is 1, then $age(x^j) \leftarrow age(x^j)+1$, and otherwise perform Step A13.

A13: If $age(x^j) > \delta \times k$, then move x^j to the Level $k+1$ island L_{k+1}, and otherwise perform Step A14.

A14: Set $j \leftarrow j+1$.

A15: Set $k \leftarrow k+1$.

In the Steps A1 and A2, the age of each individual x^n is initialized to 1.

4.2.5 Simple Approach

A simple and promising approach to enhancing genetic diversity for the Island Model GP is to increase the number of islands, although this approach necessarily causes an increase in the total population size. As a baseline method, we first investigate the original island model GP within the same island structure as the proposed method in Section 4.4, where the number of islands is M and the number of individuals in each island $L^i(m)$ is N/M. By varying the positive integer h, we investigate the original island model GP with $M+h$ islands, each of which has N/M individuals. In Section 4.6, we compare the proposed method with this naive method in terms of computational time and solution quality.

4.3 Experimental Setting

For the Island Model GP and the benchmark problems, we used the following standard parameter settings [26, 27, 28]: Recombination rate is 0.9; Tournament selection is used (size = 7, no-elitist); Total population size is $N = 2,000$; Number of islands is $M = 10$; Migration interval is $\alpha = 50$; Initial individuals are created by using "ramped half-and-half" with maximum depth; The number of trials in each experiment is 30.

All our experimentation were undertaken on a single PC running Linux with six Intel Xe on X5675 3.07GHz processors, and 8 GB RAM.

4.4 Performance Evaluation

We first compared the proposed method with the original Island Mode GP, ALPS, and 1X methods, where the 1X method indicates the simplest GP that is a sequential GP with a normal simple crossover. The 1X method is considered as a baseline method in this paper. Figure 3 shows the history of the best fitness values as a function of the number of evaluations and plots the average of $\max_{1\leq m\leq M} f(\hat{x}_m)$ over the 30 trials. In the experiments, we conducted 30 trials. Here, the pink lines indicate the results of 1X.

We first compared the 1X method with the other methods in terms of solution quality. From Figure 3, the 1X method performs worse than the other methods in the Santa Fe ant trail, royal tree, symbolic regression problems. These problems contain local minima that need an effective migration strategy to improve performance.

Figure 3(a) shows the results of the Santa Fe ant trail problem. We can see that it is difficult to obtain a global optimum of 89 food units without an effective strategy. The original Island Model GP gives comparable performance to ALPS. Moreover, the proposed method outperforms both the original method and ALPS. Figures 3(b) and (c) show the results of the royal tree problems. In the Class-D

case, we can see that the 1X method converged to a local minimum, and that the proposed method performs similarly to the original Island Model GP and ALPS. These results indicate that the island models such as the Original Island Model GP and proposed methods and ALPS can avoid local minima.

In the Class-E royal tree problem, the fitness of the 1X method did not change once the number of generations had exceeded 6×10^5, which suggests that the 1X method again converged to a local minimum. The proposed method outperformed the original Island Model GP and ALPS, and was the only method to reach the global optimum. The royal tree problems are typical benchmark problems for evaluating the effectiveness of frequent trees, and are therefore suitable for evaluating the proposed method. These results verify the effectiveness of the proposed evaluation measure that takes subtree information into account.

Figure 3(d), (e), (f) show the results for the symbolic regression problems using functions A, B, and C, respectively. Form Figure 3(f), we can see that the 1X method obtained lower fitness values than the other methods, whereas the performance of the proposed method is comparable to that of both the original Island Model GP and ALPS. Furthermore, Figures 3(d), and (e) suggest that the proposed approach can outperform the other methods. These results imply that considering the number of new types of frequent trees $g(L_m)$ improves the solution quality, and the proposed evaluation measure effectively enhances performance. That is, incorporating both $g(L_m)$ and the fitness value $f\left(x_m^f\right)$ leads to better performance.

4.5 Comparison with Naive Approaches

In Section 4.4, we demonstrated the effectiveness of the proposed method, and showed that the 1X method converges to local minima in all problems. Next, we compared the proposed method with M islands and the original Island Model GP with $M+h$ islands ($h = 0, 10, 20$) in terms of solution quality and computational time. Table 1 gives the results of the Santa Fe Ant trail problem. Tables 2 and 3 present the results of the royal tree problem of Class-D and Class-E, respectively. Tables 4, 5, and 6 list the results of the symbolic regression problems with functions A, B, and C, respectively. Tables 1 to 6 each present the solution quality $\max_{1 \le m \le M} f\left(x_m^f\right)$ of the final generation and its computation time.

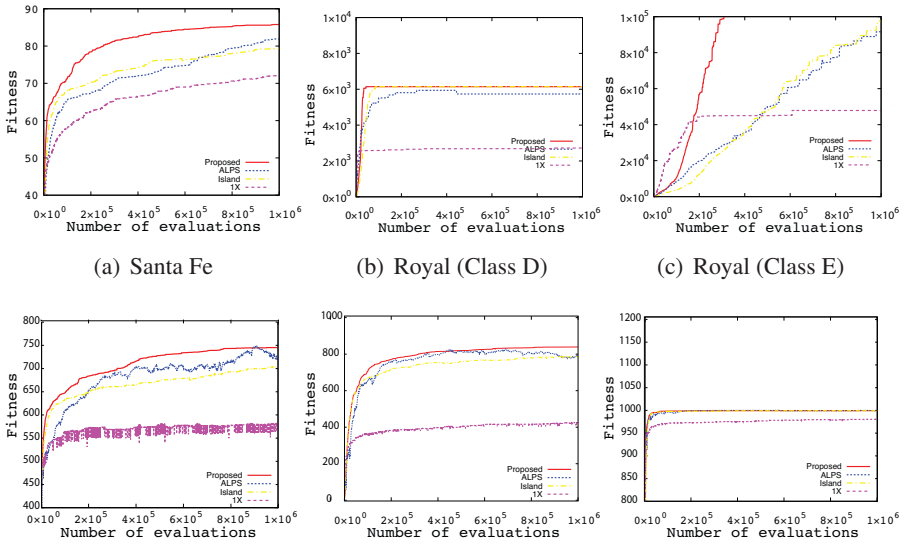
From Tables 1 to 6, we can first confirm that the original Island Model GP with $h = 0$ is faster than the proposed method, and that the original Island Model GP with $h = 20$ gives a higher-quality solution than with $h = 0$, although it requires significantly more computation time. However, the proposed method was faster than the original Island Model GP for $h = 10$ in all problems. Moreover, the proposed method gave higher-quality solutions than the original Island Model GP for $h = 10$, and the processing time required by the proposed method was less than five min. The proposed method did not require significantly more computational time, because it only extracts frequent trees of relatively small size at the migration times. Indeed, FREQ can incrementally extract the frequent trees of size k in L_m in order of increasing size k. The algorithm is very efficient and scales almost linearly with the total size of the maximal frequent trees. As the proposed method extracts frequent trees of relatively small sizes only at migration times using FREQ, its computational complexity does not increase significantly, unlike in the original Island Model GP. In fact, the frequent trees extracted by the proposed method have a size of at most three. Also, the executions to extract frequent trees were performed only at migration times. These results demonstrate the effectiveness of the proposed method.

Table 1. Results of the Santa Fe ant trail problem.

Method	Fitness	Time[sec.]
Proposed	85	101
Original ($h = 0$)	79.3	42
Original ($h = 10$)	84.3	117
Original ($h = 20$)	80.6	162

Table 2. Results of the royal tree (Class-D) problem.

Method	Fitness	Time [sec.]
Proposed	6144	21
Original ($h = 0$)	6144	11
Original ($h = 10$)	6144	145
Original ($h = 20$)	6144	178



(a) Santa Fe (b) Royal (Class D) (c) Royal (Class E)

(d) Function A (e) Function B (f) Function C

Figure 3. Best fitness values achieved by the comparative algorithms.

Table 3. Results of the royal tree (Class-E) problem.

Method	Fitness	Time [sec.]
Proposed	113202	100
Original ($h = 0$)	97577	47
Original ($h = 10$)	116238	160
Original ($h = 20$)	118141	197

Table 4. Results of the symbolic regression problem (Function A).

Method	Fitness	Time [sec.]
Proposed	739	163
Original ($h = 0$)	704	117
Original ($h = 10$)	661	629
Original ($h = 20$)	683	1032

Table 5. Results of the symbolic regression problem (Function B).

Method	Fitness	Time [sec.]
Proposed	815	169
Original ($h = 0$)	786	87
Original ($h = 10$)	814	424
Original ($h = 20$)	859	568

Table 6. Results of the symbolic regression problem (Function C).

Method	Fitness	Time [sec.]
Proposed	999	111
Original ($h = 0$)	999	81
Original ($h = 10$)	1000	276
Original ($h = 20$)	1000	350

4.6 Behavior Analysis

We hypothesized that an island L_m will have a high activation level and maintains diversity when $g(L_m)$ is large (i.e., many types of frequent trees are newly created in L_m). In Section 4.4, we compared various methods in terms of solution quality, and showed that the effectiveness of the proposed method incorporating $g(L_m)$ can be effective. Here, we analyze the effects of the evaluation measure $F(L_m)$ and the local search. Table 7 presents the final generation results of each method, where #OPT indicates the number of trials in which the optimal solution was found. These are averages over 30 trials. First, we verify that the ranking-based and proposed methods outperform 1X and the original Island model in all problems. In the symbolic regression problem with function A, ALPS outperformed the ranking-based and proposed methods; however, the ranking-based and proposed methods achieved better performance than ALPS in the other

Table 7. Effects of the local search

	Ant		Royal (Class D)		Royal (Class E)	
	Fitness	#Opt	Fitness	#Opt	Fitness	#Opt
1X(M=1)	72.1	1	2720	9	47878.3	9
Original(M=10)	79.3	5	6144	30	97577.3	20
ALPS	82.7	4	5734	29	103974.8	22
f-evaluation	85.4	13	6144	30	107243.7	25
g-evaluation	84.6	11	6144	30	97514.9	22
Ranking-based	*85.8	16	6144	30	107072.4	25
Proposed(without LS)	83.9	9	6144	30	99693.6	23
Proposed	**85.2**	**12**	**6144**	**30**	***113201.9**	**27**
	FuncA		FuncB		FuncC	
	Fitness	#Opt	Fitness	#Opt	Fitness	#Opt
1X	565.2	0	424.3	0	980.7	19
Original	704.1	0	786.4	2	999.2	29
ALPS	*751.1	0	793.0	8	1000.0	30
f-evaluation	745.2	0	811.7	1	999.4	30
g-evaluating	740.0	0	818.1	0	999.6	30
Ranking-based	725.0	0	824.1	0	1000.0	30
Proposed(without LS)	733.6	0	*839.3	2	999.5	29
Proposed	**739.1**	**0**	**815.0**	**2**	**999.4**	**30**

problems. These results show that both the ranking-based and proposed methods outperform other island models and a conventional method, Moreover, incorporating a local search into the proposed method, we can observe that the performance was improved.

Next, we analyze why the proposed method performed much better in the Class-E royal tree problem. The royal tree problem is a typical benchmark for evaluating how many partial solutions have been found. The entire solution to this problem consists of subtrees called perfect trees, and fitness values are evaluated by summing the scores of these perfect trees, which are determined according to the depth of each subtree. Therefore, we hypothesized that a higher activation level would be maintained on each island, leading to more candidates of partial solutions being discovered by the small changes in subtrees enabled by the local search. The results shown in Table 7 demonstrated that the proposed method with local search achieved the best fitness value. These results imply that the proposed method incorporating both the proposed evaluation measure $F(L_m)$ and a local search offer improved performance, which supports our hypothesis.

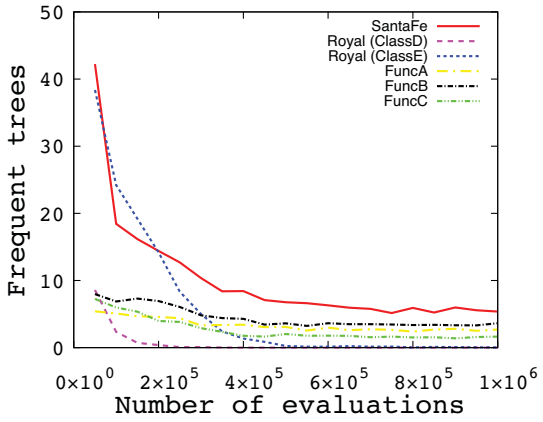


Figure 4. Number of new types of frequent trees given by proposed method.

We also investigate the history of $g(L_m)$ in the proposed method. We hypothesized that an island L_m will have a high activation level and maintains diversity when $g(L_m)$ is large (i.e., many types of frequent trees are newly created in L_m). We now examine the history of $g(L_m)$ given by the proposed method for each of the benchmark problem. Figure 4 plots the average of $\bar{g}$ over the 30 trials as a function of the number of evaluations, where the average of the number of new types of frequent trees

is

$$\bar{g} = \frac{1}{M}\sum_{m=1}^{M} g(L_m).$$

We can see that the Santa Fe ant trail and royal tree (Class-E) problems consist of more frequent trees than the other problems during the first half of the generation process. Moreover, the Class-E problem decays much more slowly than the other problems, and the $\bar{g}$ is much lower than for the Class-D problem. However, the Santa Fe ant trail and royal tree problems are generally more affected by subtrees than the symbolic regression problems. Therefore, in Figure 4, these results imply that the proposed method would continue to create new types of frequent trees in these problems.

Finally, we investigated the relationship between $\bar{g}$ and the solution quality by comparing the average of the $\bar{g}$ of good trials and bad trials in each problem, where good trials indicate those that lead to optimal solutions. The total number of trials was 30. Figure 5 shows the results for the Santa Fe ant trail problem and Figure 6 shows the results for the Class-E royal tree problem. From Figure 5, we can see that there is no significant difference in $\bar{g}$ between good trails and bad trials. However, in the Class-E problem, the number of frequent trees in good trials is clearly much higher than that in bad trials. The partials solutions play an important role in determining the solution quality of a royal tree problems; therefore, we considered that the number of new types to have increased in the Class-E royal tree problem. From these results, we speculate that incorporating the fitness $f\left(x_m^f\right)$ restricts the search to better areas, and incorporating $g(L_m)$ makes it easier to find pieces of the partial solutions. These results support our hypothesis.

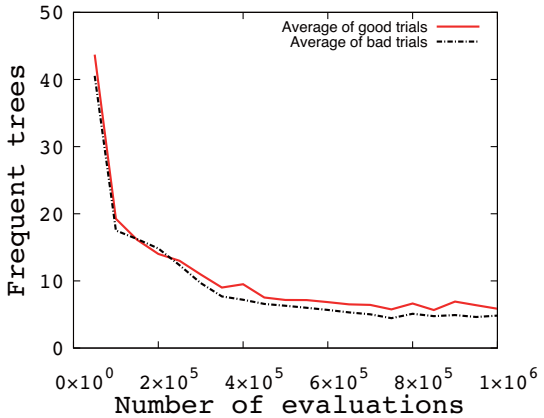


Figure 5. Comparison of good trials and bad trials in terms of the number of new types of frequent trees of the proposed method in the Santa Fe ant trail problem.

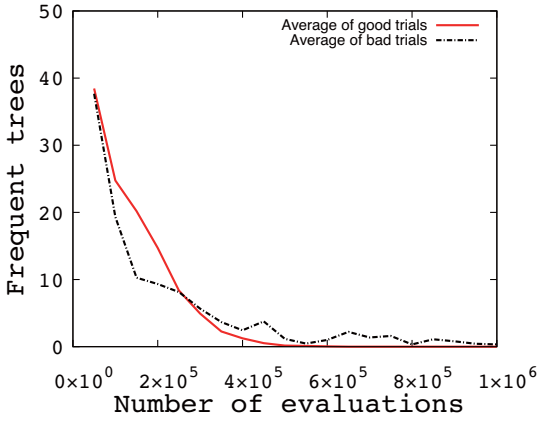


Figure 6. Comparison of good trials and bad trials in terms of the number of new types of frequent trees of the proposed method in the royal tree problem of Class-E.

Conclusions and Future Work

To improve the Island Model GP, it is essential to obtain an effective migration strategy. In this paper, we have described an enhanced Island Model GP in which the migration pair changes adaptively according to each island's condition for maintaining genetic diversity. We in this paper introduced a real value-based novel evaluation measure based on fitness and frequent trees according to the Building Block Hypothesis. The proposed method evaluates each island's activation level in terms of both the fitness value and how many types of frequent trees are newly created. Based on this evaluation measurement, some individuals migrate from an island with a higher activation level to an island with a lower activation level, and vice versa. Therefore, the proposed method adaptively changes the migration process in terms of the proposed evaluation measurement. Moreover, the proposed method incorporates a local search method for further performance improvement. We assumed that local searches are effective for discovering frequent trees as candidates of partial solutions to the problems like the royal tree problems, in which the total fitness of an individual is given by accumulating sub-

trees' fitness scores. The local search process can change a part of a solution and enhance the combination of partial solutions.

Using well-known benchmark problems widely adopted in the literature, we confirmed that the proposed migration strategy offers significant performance advantages over the 1X, original Island Model GP, and ALPS methods. We also analyzed the behavior of the proposed method and demonstrated its validity. From the experimental results, we observed that the proposed method offers good performance, and confirmed that incorporating the information about the frequent trees and the local search into the migration strategy gives improved results. Although we have only presented results for the Santa Fe ant problem, two royal tree problems and three symbolic regression problems, similar results can be achieved for other symbolic regression problems. In future work, we will evaluate our method using other benchmark problems.

The proposed method essential implements a migration strategy based on subtrees. Several subtree-based crossovers have been proposed: the context-aware crossover [31, 32] and the subtree based crossover [33, 19]. As a first step, the proposed method adopted the simplest crossover and had not yet been applied to these enhanced crossover techniques. Therefore, we will work to appropriately integrate the subtree-based crossover into the proposed method for further performance improvement. On the other hand, Burke et al. extensively examined the relation between diversity measure and fitness [34]. The proposed method measures genetic diversity on the basis of the frequent trees and fitness. Thus, we will also attempt to incorporate other measures into the proposed method.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers 26730133 and 26330290.

References

[1] J. Koza, in Genetic Programming and Evolvable Machines, vol. 11 (2010), vol. 11, pp. 251–284

[2] Y. Shichel, E. Ziserman, M. Sipper, in Proceedings of 8th European Conference on Genetic Programming (Eurogp2005). (to appear). xxx.tex; 12/06/2005; 9:07; p.24 Genetically Programming Backgammon Players, Draft 25 (SpringerVerlag, 2005), pp. 143–154

[3] B. Samie, G. Dragffy, A. Pipe, Y. Liu, in Proceedings of the Third European Conference on Genetic Programming EuroGP'00 (2011), pp. 73–84

[4] V.S. Gordon, D. Whitley, D. Whitley, in Proceedings of the 5th International Conference on Genetic Algorithms (1993), pp. 177–183

[5] D. Whitley, Statistics and Computing 4, 65 (1994)

[6] J.J.G. Chrisila B. Pettey, Michael R. Leuze, in Proceeding Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application (1987), pp. 155–161

[7] D. Andre, J.R. Koza, in Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'96, Volume III (1996), pp. 1163–1174

[8] W.F. Punch, in Proceedings of the Third Annual Conference Genetic Programming (1998), pp. 308–313

[9] F. Fernández, M. Tomassini, W.F. Punch III, J.M. Sanchez, in Proceedings of the Third European Conference on Genetic Programming EuroGP'00 (2000), pp. 283–293

[10] F. Fernández, G. Galeano, J.A. Gómez, in Proceedings of the 5th European Conference on Genetic Programming EuroGP'02 (2002), pp. 326–336

[11] J. Hu, E.D. Goodman, K. Seo, in Genetic Programming Theory and Practice, ed. by R.L. Riolo, B. Worzel (Kluwer, 2003), chap. 6, pp. 81–98

[12] J. Hu, E.D. Goodman, K. Seo, M. Pei, in GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, ed. by W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, N. Jonoska (Morgan Kaufmann Publishers, New York, 2002), pp. 772–779

[13] J. Hu, E. Goodman, K. Seo, Z. Fan, in Evolutionary Computation, vol. 13 (2005), vol. 13, pp. 241–277

[14] G.S. Hornby, in In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (ACM, 2009), pp. 795–802

[15] G.S. Hornby, in Genetic Programming Theory and Practice VII, ed. by R.L. Riolo, U.M. O'Reilly, T. McConaghy (Springer, 2009), Genetic and Evolutionary Computation, pp. 87–102

[16] G. Hornby, in GECCO (2006), pp. 815–822

[17] M.F. Korns, in Genetic Programming Theory and Practice IX, ed. by R. Riolo, E. Vladislavleva, J.H. Moore (Springer, Ann Arbor, USA, 2011), Genetic and Evolutionary Computation, pp. 129–151

[18] R. Poli, N.F. McPhee, Evolutionary Computation 11(2), 169 (2003)

[19] N.F. McPhee, B. Ohs, T. Hutchison, in Proceedings of the 11th European conference on Genetic programming EuroGP'08 (2008), pp. 134–145

[20] S.C. Roberts, D. Howard, J.R. Koza, in Proceedings of the 4th European conference on Genetic programming EuroGP'01 (2001), pp. 160–175

[21] K. Ono, Y. Hanada, M. Kumano, M. Kimura, in IEEE Congress on Evolutionary Computation (IEEE, 2013), pp. 2988–2995

[22] T. Asai, K. Abe, S. Kawasoe, H. Sakamoto, S. Arikawa, in Proceedings of SIAM International Conference on Data Mining SDM'02 (2002), pp. 158–174

[23] T. Asai, H. Arimura, T. Uno, S. ichi Nakano, in Proc. of the 6th Intl. Conf. on Discovery Science (Springer-Verlag, 2003), pp. 47–61

[24] C. Fillon, A. Bartoli, in Proceedings of the 9th European conference on Genetic Programming EuroGP'06 (2006), pp. 13–23

[25] S. Luke, G.C. Balan, L. Panait, in Proceedings of the 2003 international conference on Genetic and evolutionary computation GECCO'03: Part II (2003), pp. 1729–1739

[26] K. Yanai, H. Iba, in Proceedings of the 2003 Congress on Evolutionary Computation CEC'03, vol. 3 (2003), vol. 3, pp. 1618–1625

[27] W.B. Langdon, R. Poli, Foundations of genetic programming (Springer, 2002)

[28] W. Langdon, R. Poli, in Proceedings of the Third Annual Conference Genetic Programming GP'98 (1998), pp. 193–201

[29] A. In, K. Kinnear, B. Punch, D. Zongker, E. Goodman, in Advances in Genetic Programming II, MIT Press (1996)

[30] C. Tuite, M. O'Neill, A. Brabazon, in GECCO (Companion) (ACM, 2013), pp. 151–152

[31] H. Majeed, in Proceedings of the 2005 workshops on Genetic and evolutionary computation GECCO'05 (2005), pp. 378–381

[32] H. Majeed, C. Ryan, in Proceedings of the 9th European conference on Genetic Programming EuroGP'06 (2006), pp. 36–48

[33] J. McDermott, U.M. O'Reilly, L. Vanneschi, K. Veeramachaneni, in Proceedings of the 14th European conference on Genetic programming EuroGP'11 (2011), pp. 190–202

[34] E. Burke, S. Gustafson, G. Kendall, IEEE Transactions on Evolutionary Computation **8**(1), 47 (2004)

Keiko Ono received her B.S., M.S., and Ph.D. degrees in engineering from Doshisha University, Kyoto, Japan, in 2001, 2003 and 2007, respectively. In April 2009, she joined Doshisha University as Assistant Professor of the Organization for Research Initiatives and Development. In April 2010, she joined Ryukoku University, Kyoto, Japan. Currently, she is an Associate Professor in the Department of Electronics and Informatics. Her research interests include parallel computing, evolutionary optimization and machine learning. She is a member of the Institute of Electrical and Electronics Engineers (IEEE), the Japanese Society for the Information Processing (IPSJ), the Japanese Society for Evolutionary Computation (JPNSEC).

Yoshiko Hanada received the B. E., M. E. and Ph. D. (Engineering) from Doshisha University, Japan in 2002, 2004 and 2007, respectively. She was a Japan Society for the Promotion of Science (JSPS) Research Fellow from April 2006 to March 2008. In April 2008, she joined Faculty of Engineering Science, Kansai University, Osaka, Japan, and has been an associate professor since 2016. Her research interests include the development of novel evolutionary algorithms, and combinatorial optimization. She is a member of IEEE, INFORMS, IPSJ, IEICE and IEEJ.

Masahito Kumano received the B.E. degree from the Ritsumeikan University, Kyoto, Japan, in 1991, and the Ph.D. degree in Engineering from Kobe University, Kobe, Japan, in 2008. In April 1991, he joined Ryukoku University, Kyoto, Japan.

Currently, he is a senior teaching associate in the Department of Electronics and Informatics. His research interests include computer science, visualization, and machine learning. He is a member of the Japanese Society for Artificial Intelligence (JSAI), the Information Processing Society of Japan (IPSJ), the Institute of Electronics, Information and Communication Engineers (IEICE), the Institute of Electrical and Electronics Engineers (IEEE), and the Association for Computing Machinery (ACM).

Masahiro Kimura received his B.S., M.S., and Ph.D. degrees in mathematics from Osaka University, Osaka, Japan, in 1987, 1989, and 2000, respectively. In April 1989, he joined Nippon Telegraph and Telephone (NTT) Corporation, Tokyo, Japan, and mainly worked at NTT Human Interface Laboratories and NTT Communication Science Laboratories. In April 2005, he joined Ryukoku University, Kyoto, Japan. Currently, he is a professor in the Department of Electronics and Informatics. His research interests include complex networks science, data mining, and machine learning. He is a member of the Japanese Society for Artificial Intelligence (JSAI), the Mathematical Society of Japan (MSJ), the Japan Society for Industrial and Applied Mathematics (JSIAM), and the Institute of Electronics, Information and Communication Engineers (IEICE).