# Effectiveness of Multi-step Crossover Fusions in Genetic Programming

Yoshiko Hanada*, Nagahiro Hosokawa†, Keiko Ono‡ and Mitsuji Muneyasu*
*Faculty of Engineering Science, Kansai University, Osaka, Japan
Email: {hanada,muneyasu}@kansai-u.ac.jp
†Graduate School of Science and Engineering, Kansai University, Osaka, Japan
‡Department of Electronics and Informatics, Ryukoku University, Shiga, Japan
Email: kono@rins.ryukoku.ac.jp

***Abstract*—Multi-step Crossover Fusion (MSXF) and deterministic MSXF (dMSXF) are promising crossover operators that perform multi-step neighborhood search between parents, and applicable to various problems by introducing a problem-specific neighborhood structure and a distance measure. Under their appropriate definitions, MSXF and dMSXF can successively generate offspring that acquire parents' good characteristics along the path connecting the parents. In this paper, we introduce MSXF and dMSXF to genetic programming (GP), and apply them to symbolic regression problem. To optimize trees, we define a neighborhood structure and its corresponding distance measure based on the largest common subtree between parents with considering ordered/unordered tree structures. Experiments using symbolic regression problem instances showed the effectiveness of a GP with the proposed MSXF and dMSXF.**

## I. Introduction

Crossover operators or recombination operators have been considered to be the central component of population-based optimization algorithms. Especially in optimizing combinatorial structures such as a graph, it is important to design the operator to consider problem-specific structures and characteristics. Various operators focusing on the inheritance of parents' characteristics have been discussed in genetic algorithms (GAs) [1], [2], [3]. The design of the crossover operator is important also in genetic programming (GP) [4] of which search mechanism is same as that of GA and focusing on optimizing tree structures. A tree structure is a special case of a graph; however it has some features, such as a symmetric property, a self-similarity and an anisotropy in the landscape of an objective function, which make the design of genetic operators complex. There are various recombination mechanisms to treat tree features in GPs [5], [6], [7], [8]. In addition, it has been considered that mutation operators are essential to acquire a characteristic not observed in population with keeping good characteristics (building blocks), since crossovers might bring a drastic change in solutions and break favorable characteristics.

To treat complex constraints and the design variables dependency in solving combinatorial optimization problems, Multi-step Crossover Fusion (MSXF) [9] and deterministic MSXF (dMSXF) [10] have been proposed. MSXF and dMSXF are a kind of recombination operators and perform a sequence of local search that moves the offspring from its initial point to the other parent. The main difference of these procedures is that the MSXF determines the transition in the local search by Metropolis criterion while dMSXF advances the local search in a deterministic rule. MSXF and dMSXF are defined in a problem-independent manner, and have been successfully applied to various combinatorial optimization problems [10], [11], [12] since the incorporation of local searches into GAs is essential in order to adjust the structural details of solutions [13].

In this paper, we introduce MSXF and dMSXF to GP and compare their search performances, using symbolic regression problems. To optimize tree structures, we define both a neighborhood structure focusing on graphical patterns observed in trees and its corresponding distance metric, for MSXF and dMSXF. In the local search, neighborhood solutions are generated by keeping the largest common subtree between parents in order not to break the parents' characteristics. To extract the largest common tree, two types of tree structures, the ordered tree and the unordered tree, are examined. Through the numerical experiments, we show the effectiveness of MSXF and dMSXF and discuss how the definitions of the neighborhood structure and the setting of parameters affect the search performance.

## II. Multi-Step Crossover Fusions

### A. Multi-Step Crossover Fusion

MSXF is one of the crossover operators which are fused with a local search. In MSXF, a solution, initially set to be one of the parents $p_1$, is stochastically replaced by a relatively good solution from the neighborhood, where the replacement is biased toward the other parent $p_2$. After a certain number of iterations of this process, the best one among the generated solutions is selected as an offspring.

The procedure of MSXF is summarized as follows. The set of offspring generated by parents $p_1, p_2$ is defined as $C(p_1, p_2)$.

**Procedure of MSXF**

1. Let $p_1$, $p_2$ be parents and set their offspring $C(p_1, p_2) = \phi$.
2. $k$=1. Set the initial search point $x_1 = p_1$ and add $x_1$ into $C(p_1, p_2)$.

3. /Step $k$/ Prepare $N(x_k)$ neighborhoods composed of $\mu$ solutions generated randomly from the current solution $x_k$. $\forall y_i \in N(x_k)$.
4. Sort neighborhoods $y_i$ in $N(x_k)$ in ascending order according to the distance $d(y_i, p_2)$, and a member $y$ is selected from $N(x_k)$ randomly, but a smaller index is preferred.
5. The member $y$ is probabilistically accepted as the next solution $x_{k+1}$ according to Metropolis criterion, i.e., $y$ is accepted with probability 1 if $E(y) < E(x_k)$; otherwise, with the probability

$$P(y) = \exp(-\Delta E/T), \tag{1}$$

where $\Delta E = E(y) - E(x)$. Let the next search point $x_{k+1}$ be $y$, and $x_{k+1}$ is added into $C(p_1, p_2)$ if the transition is accepted; otherwise, go to 4.
6. Set $k = k + 1$ and go to 2., until $k = k_{max}$ or $x_k$ equals $p_2$.

MSXF requires two parameters, $k_{max}$ and $\mu$. $k_{max}$ is the number of steps of local search and $\mu$ is the number of generated solutions at each step of the local search. In the procedure of MSXF, at most $k_{max} \cdot \mu$ solutions would be generated, and $C(p_1, p_2)$ is comprised of the best neighbor solutions, i.e., $\{x_1, x_2, \cdots, x_{k_{max}}\}$.

MSXF is applicable to various problems by introducing a problem-specific neighborhood structure and a distance measure, and it has been shown to perform very well on job scheduling problem (JSP).

### B. Deterministic Multi-Step Crossover Fusion

dMSXF [10] is a specific case of MSXF. dMSMF has been shown to outperform other heuristic methods under the definition of sophisticated neighborhood structures and distance metrics on JSP and traveling salesman problem (TSP) [11].

dMSXF also performs multi-step local search from parent $p_1$ in the direction approaching the other parent $p_2$. Each transition in the local search is accepted by the deterministic rule composed of a distance measure, while MSXF determines it by Metropolis criterion according to the quality of a solution. The procedure of dMSXF is as follows.

**Procedure of dMSXF**

1. Let $p_1$, $p_2$ be parents and set their offspring $C(p_1, p_2) = \phi$.
2. $k$=1. Set the initial search point $x_1 = p_1$ and add $x_1$ into $C(p_1, p_2)$.
3. /Step $k$/ Prepare $N(x_k)$ composed of $\mu$ neighbors generated from the current solution $x_k$. $\forall y_i \in N(x_k)$ must satisfy $d(y_i, p_2) < d(x_k, p_2)$.
4. Select the best solution $y$ from $N(x_k)$. Let the next search point $x_{k+1}$ be $y$, and $x_{k+1}$ is added into $C(p_1, p_2)$.
5. Set $k = k + 1$ and go to 2. until $k = k_{max}$ or $x_k$ equals $p_2$.

At step 3 of the procedure of dMSXF, every neighborhood candidates $y_i$ $(1 \leq i \leq \mu)$ generated from $x_k$ must be closer to $p_2$ than $x_k$. In addition, dMSXF necessarily moves its transition toward $p_2$ even if all solutions in $N(x_k)$ are inferior to the current solution $x_k$. dMSXF requires two parameters, $k_{max}$ and $\mu$ and $C(p_1, p_2)$ is comprised of $\{x_1, x_2, \cdots, x_{k_{max}}\}$.

### C. Generation Alternation Model

The generation alternation model we used in this paper is outlined below. This model focuses on a local search performance and it showed effectiveness in combinatorial optimization problems [3], [10], [11].

**Flow of GP**

1. Generate the initial population composed of $N_{pop}$ of random solutions, individuals, $\{x_1, x_2, \cdots, x_{N_{pop}}\}$.
2. Reset indexes $\{1, 2, \cdots, N_{pop}\}$ to each individual randomly.
3. Select $N_{pop}$ pairs of parents $(x_i, x_{i+1})$ $(1 \leq i \leq N_{pop})$ where $x_{N_{pop}+1} = x_1$.
4. MSXF (dMSXF) is applied to each pair $(x_i, x_{i+1})$.
5. For each pair $(x_i, x_{i+1})$, select the best individual $c$ from offspring $C(x_i, x_{i+1})$ generated by parents $(x_i, x_{i+1})$ and replace the parent $x_i$ with $c$.
6. Go to 1 until some terminal criterion is satisfied, e.g., generations and/or the number of evaluations.

## III. Designs of MSXF for Tree Representation

To apply MSXF or dMSXF, a neighborhood structure and its corresponding pair-wise distance measure should be defined. Both multi-step crossover operators would work very well with sophisticated definitions that can express characteristics of problem-specific structures. Thus these definitions, especially the definition of the neighborhood structure, are most important to improve the search performances of MSXF and dMSXF.

### A. Definition of Distance Measure

In tree structures, a graphical pattern, a motif, is one of the most useful information to comprehend features of a tree. Extracting important frequent patterns occurring from tree structured data have been discussed in not only for the design of the crossover in GP but also for a tree mining in the semi-structured data such as large collections of web pages [14], [15].

Here we introduce a distance measure based on different nodes derived from both the largest common subtree (subgraph) and the subtrees consisting of different edges between two trees. The largest common subtree is the connected subtree that possesses the largest set of the common edges between trees without considering the node's symbols. Figure 1 shows an example of the largest common subtree and the different subtrees between two trees in a symbolic regression problem instance. In this example, the symmetric property of the relation between nodes, i.e., the order between the children nodes, is ignored. Here, $com(A)$ and $com(B)$ denote the largest common subtrees respectively in tree $A$ and $B$, and $dif(A)$ and $dif(B)$ indicate the different subtrees. These different subtrees are calculated by removing edges (links) and nodes included in $com(A)$ or $com(B)$ from $A$ or $B$.

Using some definition of the largest common subtree and the different subtrees, the distance between tree $A$ and $B$,

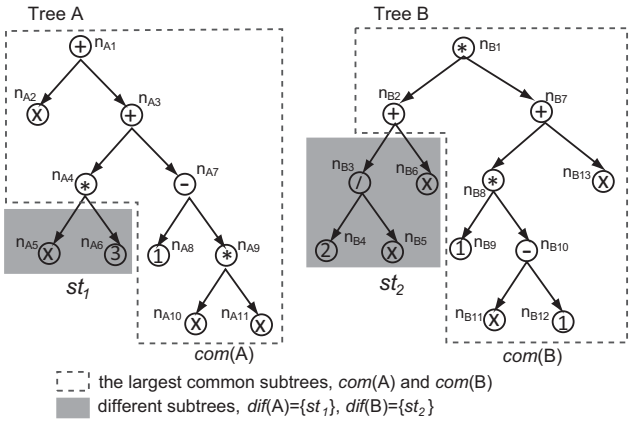


Fig. 1. The largest common subtree and the different subtrees (unordered tree; UT)

$d(A, B)$, is defined as

$$d(A,B) = \frac{1}{2}(|N_{com_d(A)}| + |N_{com_d(B)}| + |N_{dif(A)}| + |N_{dif(B)}|) \quad (2)$$

where $N_{com_d(A)}$ and $N_{com_d(B)}$ are the set of nodes that possess a different node's symbol located at the same place respectively in the subtree $com(A)$ and $com(B)$, and $|\cdot|$ means the number of components. $N_{dif(A)}$ and $N_{dif(B)}$ indicate the set of nodes respectively included in $dif(A)$ and $dif(B)$. In the example shown in Fig. 1, $N_{com_d(A)} = \{n_{A_1}, n_{A_2}, n_{A_4}, n_{A_7}, n_{A_9}, n_{A_{11}}\}$, $N_{com_d(B)} = \{n_{B_1}, n_{B_2}, n_{B_8}, n_{B_{10}}, n_{B_{12}}, n_{B_{13}}\}$ and $|N_{com_d(A)}| = |N_{com_d(B)}|$=6. The components in the difference subtrees are respectively $N_{dif(A)}$=$\{n_{A_5}, n_{A_6}\}$ ($|N_{dif(A)}|$=2) and $N_{dif(B)}$= $\{n_{B_3}, n_{B_4}, n_{B_5}, n_{B_6}\}$ ($|N_{dif(B)}|$=4).

The calculation of the largest common subtree depends on the intended tree structure. Here, two types of tree structures, unordered tree (UT) and ordered tree (OT), are considered and two kinds of the distance based on UT or OT are introduced. UT and OT are kinds of a directed acyclic connected graph with a fixed root node, and the former does not consider the order of nodes among children nodes while the latter does. The symmetric property in descendant nodes is ignored under the definition of UT. Figure 1 is an example of $com(A)$, $com(B)$, $dif(A)$ and $dif(B)$ under the definition of UT. In the difinition of UT, for example, the tree "- + $x$ 1 $x$" and the tree "- $x$ + $x$ 1" are considered equal as shown in Fig. 2.

On the other hand, the children of each node have a specific order in OT. This tree representation discriminates the tree "- + $x$ 1 $x$" from the tree "- $x$ + $x$ 1" as shown in Fig. 2. Figure 3 illustrates the largest common subtrees $com(A)$ and $com(B)$, and the different subtrees $dif(A)$ and $dif(B)$ under the definition of OT. The trees in this example are same as those of Fig. 1. The component of each subtree is respectively $N_{com_d(A)}$=$\{n_{A_1}, n_{A_2}, n_{A_7}, n_{A_9}, n_{A_{11}}\}$, $N_{com_d(B)}$ =$\{n_{B_1}, n_{B_2}, n_{B_8}, n_{B_{10}}, n_{B_{12}}\}$ ($|N_{com_d(A)}|$= $|N_{com_d(B)}|$=5), $N_{dif(A)}$=$\{n_{A_4}, n_{A_5}, n_{A_6}\}$ ($|N_{dif(A)}|$=3) and $N_{dif(B)}$=$\{n_{B_3}, n_{B_4}, n_{B_5}, n_{B_6}, n_{B_{13}}\}$ ($|N_{dif(B)}|$=5).

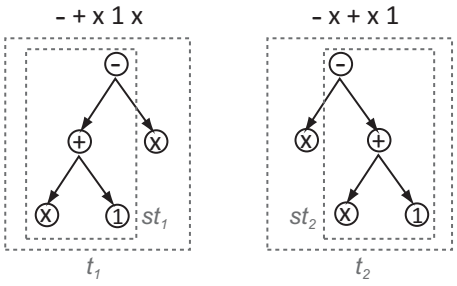


Fig. 2. The symmetric property and the difference between UT and OT. $t_1$ and $t_2$ are extracted under UT, while $st_1$ in "- + $x$ 1 $x$" and $st_2$ in "- $x$ + $x$ 1" are regarded as the largest common trees under OT.

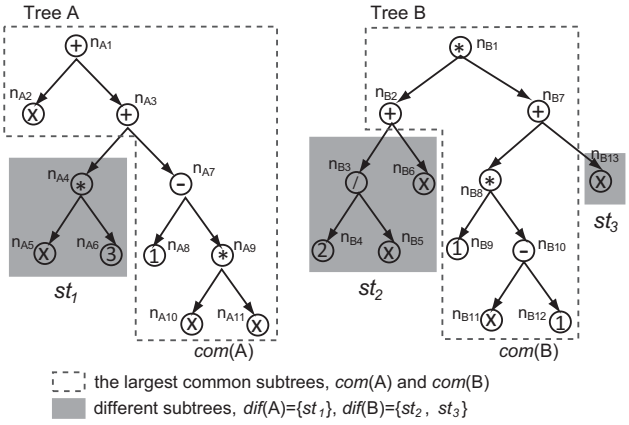


Fig. 3. The largest common subtree and the different subtrees (ordered tree; OT)

The distance between $A$ and $B$ is 9 ($\frac{1}{2}$(6+6+2+4)=9) if we treat the tree as UT, while the distance is 9 ($\frac{1}{2}$(5+5+3+5)=9) if the tree is treated as OT[1]. $com(A)$ and $com(B)$ are calculated at each step in the local search in MSXF and dMSXF; nevertheless, they can be calculated by a simple dynamic programming with the computation cost of $O(|N_A| \cdot |N_B|)$, where $|N_A|$ and $|N_B|$ are the number of nodes respectively included in tree $A$ and $B$ [16]. If several candidates of the largest subtree exist, one subtree is randomly selected.

### B. Neighborhood Structure

Here we propose three types of generation methods of neighborhood solutions in the local search in MSXF and dMSXF. In many cases in optimizing tree structures, one of the most important constraints is to keep the number of children nodes that any node should have. For example, in a kind of symbolic regression problem, operations, such as arithmetic operations or other numerical functions, should have 1 or 2 children nodes, while constants and variables have no children nodes. A node that possesses illegal children nodes does not work correctly as an operation, a program or a constant, even if it is simply allowable as a component of an abstract tree

[1]The total distances on UT and OT are equal in the case of these two simple trees but that does not necessarily occur in all cases.

structure. In some tree structured problems, the existence of nonterminal nodes of which the number of children nodes can vary is allowed; however, in this paper, we assume that the number of children nodes that each kind of node should keep is defined in advance and does not change during the optimization.

With considering the constraint about the number of children nodes, the neighborhood solutions are generated based on both the largest common subtree and the different subtree. To describe the procedures of the generation methods briefly, we first summarize the terms we use for the explanations and introduce three operations.

*1) Terms:* The notations with regards to the largest common subtree, different subtrees and relation between nodes are described below.

- $\boldsymbol{com(X)}$ is the largest common subtree in tree $X$.
- $\boldsymbol{dif(X)}$ is the set of subtrees included in tree $X$ that are not observed in another tree $Y$. $dif(X)$ is calculated by subtracting $com(X)$ from $X$.
- $\boldsymbol{N_{com(X)}}$ is the set of nodes included in $com(X)$.
- $\boldsymbol{N_{dif(X)}}$ is the set of nodes included in $dif(X)$.
- $\boldsymbol{N_{com_d(X)}}$ is the set of nodes that possess a different symbol located at the same place between the subtree $com(X)$ and the other tree's largest common subtree $com(Y)$.
- $\boldsymbol{N_{dif_a(X)}}$ is the set of the nodes that are adjacent to a node included in $com(X)$.
- $\boldsymbol{parent(n)}$ denotes the parent node of node $n$.
- $\boldsymbol{child(n)}$ means the set of children nodes of node $n$.
- $\boldsymbol{st(n)}$ denotes a subtree of which the topmost node is $n$.
- $\boldsymbol{n^{arg}}$ means the number of children nodes that node $n$ should keep.
- $\boldsymbol{root(X)}$ denotes the root (topmost) node of tree (or subtree) $X$.

*2) Operations:* Three operations, $Replace$, $Delete$ and $Insert$, with treating the constraint about the number of children nodes are defined as follows.

- $\boldsymbol{Replace(n_1, n_2)}$ is an operation that replaces the symbol of node $n_1$ with the symbol of another node $n_2$. After the replacement, nodes $n_i$ $(1 \leq i \leq n_1^{arg} - n_2^{arg})$ are selected from $child(n_1)$ in some manner and subtrees $st(n_i)$ are deleted from $n_1$ if $n_1^{arg} > n_2^{arg}$, otherwise randomly generated terminal nodes $n_i$ $(1 \leq i \leq n_2^{arg} - n_1^{arg})$ are inserted to $n_1$ as children nodes.
- $\boldsymbol{Delete(n)}$ is applied only if the node $n$ is a nonterminal node. Node $n_c^*$ is selected from $child(n)$ in some manner and relinked to $parent(n)$ as a child node. The node $n$ and subtrees $st(n_c)$ where $n_c \in \{child(n) - n_c^*\}$ are deleted.
- $\boldsymbol{Insert(n_1, n_2)}$ is an operation that inserts the node $n_2$ to the node $n_1$ as a child node, which is applied only if the insertion destination $n_1$ is a nonterminal node. First node $n_c^*$ is selected from $child(n_1)$ in some manner and the subtree $st(n_c^*)$ is deleted from $n_1$. The node $n_2$ is then inserted to $n_1$ as a child node.

*3) Generation Methods of Neighborhood Solutions:* Here we propose three types of generation methods of neighborhood solutions derived from the current solution $x_k$ at step $k$ of the local search in MSXF and dMSXF. We refer to the proposed method as follows.

- Node Replacement (R)
- Internal Node Deletion (D)
- Node Insertion (I)

These methods are consistently designed to generate neighborhood solutions $s_n$ that satisfy the distance condition $d(s_n, p_2) < d(x_k, p_2)$ from $x_k$, not to break the largest common tree between $x_k$ and the parent $p_2$. The parent $p_2$ is the targeted solution that the local search approaches step by step starting from the other parent $p_1$. This distance condition is a specific constraint in the procedure of dMSXF, and in the original proposed procedure of MSXF the neighborhood solutions are generated randomly from the solution $x_k$, however, here we adopt the same generation manner in MSXF as that of dMSXF. This is because the generation manner has been shown to be effective in several combinatorial optimization problems.

The procedure of the node replacement is as follows and Fig. 4 shows the aspect of the replacement. This operation is applied to the node included from $com(x_k)$.

---

**Node Replacement (R):**

1. Select a node $n$ randomly from $N_{com_d(p_2)}$.
2. Find the node $n' \in N_{com(x_k)}$ of which the relative location from the root of the largest common tree, $root(com(x_k))$, in $x_k$ is same as that of $n$ from $root(com(p_2))$.
3. Apply Replace($n'$, $n$) to $x_k$. Select a node $n_c$ randomly from $\{child(n') \cap \neg N_{com(x_k)}\}$ and delete the subtree $st(n_c)$ from $x_k$ if $n'^{arg} > n^{arg}$.

---

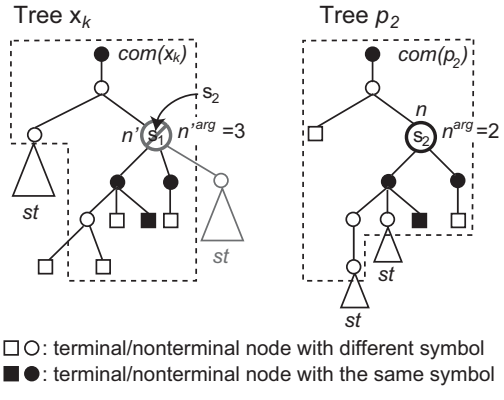


Fig. 4. Aspect of the node replacement: The symbol $s_1$ is replaced with the symbol $s_2$ and the redundant gray subtree is deleted in tree $x_k$.

In the different subtrees observed only in $x_k$, a part of nodes and subtrees are deleted with the manner described below. This operation is applied only to a nonterminal node. The aspect

of the deletion operation is shown in Fig. 5. In this figure $st$ indicates an arbitrary subtree.

**Internal Node Deletion (D):**

1. Select a nonterminal node $n$ randomly from $N_{dif_a(x_k)}$.
2. Apply Delete($n$) to $x_k$. Either node relink manner (a) or (b) on the deletion operation is applied in accordance with the location relation between $n$ and $com(x_k)$.
   (a) If $parent(n) \in N_{com(x_k)}$[2], select a node randomly from $child(n)$ and relink it to $parent(n)$ as a child node.
   (b) If $parent(n) \notin N_{com(x_k)}$, i.e., $root(com(x_k)) \in child(n)$[3];
   - if $n \neq root(x_k)$, $root(com(x_k))$ is relinked to $parent(n)$.
   - if $n = root(x_k)$, a nonterminal node $n_c$ satisfying $n_c^{arg} > 1$ is selected from $\{child(n) - root(com(x_k))\}$ as a new topmost node of $x_k$, and $root(com(x_k))$ is relinked to $n_c$. Then a node $n_d$ is selected from $\{child(n_c) - root(com(x_k))\}$ and the subtree $st(n_d)$ is deleted.

If no nonterminal nodes $n_c$ that satisfy the condition exist at step 2 (b), the subtree $st(root(com(x_k)))$ substitutes for the tree $x_k$, i.e., $root(x_k)$ and all the subtrees with regard to children nodes of $root(x_k)$ except for $root(com(x_k))$ are deleted in $x_k$.

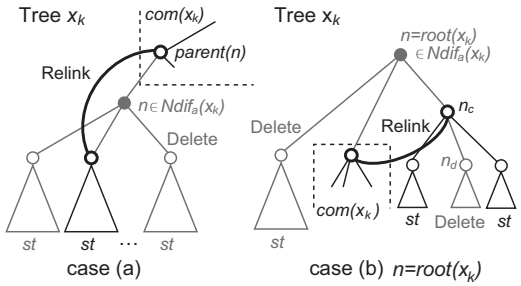


Fig. 5. Aspect of the internal node deletion: Gray edges, nodes and subtrees are deleted while the black bold edge is relinked.

The node insertion is the operation that copies a node, which is not observed in $x_k$, to $x_k$ from $p_2$ as follows. Figure 6 illustrates the aspect of the insertion.

**Node Insertion (I):**

1. Select a node $n$ randomly from $N_{dif_a(p_2)}$.
2. Either procedure (a) or (b) is applied in accordance with the location relation between $n$ and $com(p_2)$.
   (a) If $parent(n) \in N_{com(p_2)}$,
   (a-1) Find the node $n'_p \in N_{com(x_k)}$ of which the relative location from the root of the largest common tree, $root(com(x_k))$, in $x_k$ is same as that of $parent(n)$ from $root(com(p_2))$.
   (a-2) Apply Insert($n'_p$, $n$) to $x_k$. In the insertion, the order of $n$ among the children of $parent(n)$ is kept also among the children of $n'_p$[4]. A node $n_c$ is randomly selected from $\{child(n'_p) \cap \neg N_{com(x_k)}\}$ as the redundant, and the subtree $st(n_c)$ is deleted in $x_k$.
   (b) If $parent(n) \notin N_{com(p_2)}$ [5],
   (b-1) Generate a nonterminal node $n'$ of which symbol is same as that of node $n$ and let $n'$ be a new root node of $x_k$.
   (b-2) Connect $root(com(x_k))$ to $n'$ as a child node. Insert $n^{arg} - 1$ terminal nodes randomly generated to $n$ as children nodes.

[2]The case that $n$ is connected to a leaf of the largest common tree $com(x_k)$

[3]The case that the root node of $com(x_k)$ is connected to $n$

[4]For example, $n$ is inserted to $n'_p$ as the second child node if it is the second child node of $parent(n)$.

At step 2 (a-2), if the insertion destination $n'_p$ is a terminal node, Replace($n'_p$, $parent(n)$) is applied to $x_k$ in advance. If the node $n$ is a nonterminal node, random $n^{arg}$ terminal nodes are previously generated and connected to $n$ as a children nodes.

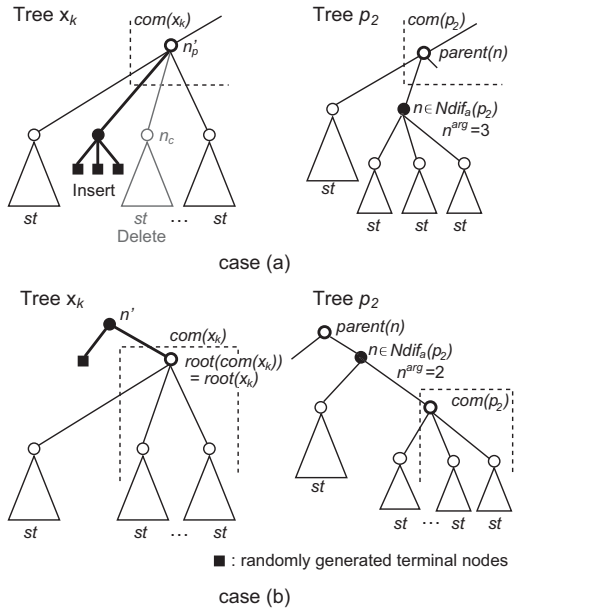


Fig. 6. Aspect of the node insertion: The gray subtree is deleted while the subtree of black solid nodes is inserted.

*4) Neighborhood generation in MSXF and dMSXF:* The neighborhood solution set $N(x_k)$, required at step 3 in both MSXF and dMSXF, is calculated as follows.

**Generation of neighborhood solution set**

1. Let $p_1$ and $p_2$ be the starting parent solution and the targeted parent solution, and set the neighborhood candidate $N(x_k) = \phi$.
2. Calculate the largest common subtrees $com(x_k)$ and $com(p_2)$ and the different subtrees $dif(x_k)$ and $dif(p_2)$.
3. Set a neighborhood solution $s_n$=$x_k$.
4. Apply Node Replacement (R), Internal Node Deletion (D) or Node Insertion (I) to $s_n$, by using the subtrees calculated at step 2, and update the subtrees.

[5]In this case, by a feature of the largest common subtree, $root(x_k)$ is identical to $root(com(x_k))$.

5. Go to 4 until the distance $d(s_n, x_k)$ reaches $d(p_1, p_2)/k_{max}$.
6. Add $s_n$ into $N(x_k)$.
7. Go to 3 until $N(x_k) = \mu$ is satisfied.

As described in section III-A, we consider two kinds of tree definitions, UT and OT. These definitions are used for the calculation of the largest common subtree and the distance at step 2, and strongly relate to the performance of generation methods of neighborhood. Hereafter the neighborhood solutions generated by the procedure above are expressed as *UT neighbor* or *OT neighbor* if the common subtree and the distance are calculated under the definition of UT or OT.

At step 4, one of three generation methods is selected stochastically with the ratio $R : D : I$, which is one of parameters of the proposed method.

## IV. Numerical Experiments

In this section, we evaluated the search performances of MSXF and dMSXF. First we show the search performance of MSXF. MSXF with UT neighbor and MSXF with OT neighbor are compared. We then discuss how the setting of the ratio among Replacement, Deletion and Insertion (R:D:I) affects obtained trees. Finally we compare the performances between MSXF and dMSXF. To assess the performances of MSXF and dMSXF, the results of 1 point crossover (1X) are shown as a comparative criterion.

### A. Problem domain and instances

We evaluated the performance of MSXF and dMSXF on symbolic regression problem that is one of basic problem domains for assessing the search performance. We picked two kinds of functions of one variable as follows. The first is an easy instance that can be expressed by only basic arithmetic operators, while the second one consists of arithmetic operators and several numerical functions.

**Instance (I):** The function that should be identified is as follows. The number of sample points is 21, and they are placed at even interval in the domain [-1, 1]. This function is unimodal.

$$f_1^{opt} = x^4 + x^3 + x^2 + x \tag{3}$$

**Instance (II):** The function that should be identified is as follows. The number of sample points is 101, and they are placed at even interval in the domain [0, 10]. This instance is a multimodal function.

$$f_2^{opt} = x \sin x(\cos x - 1) \tag{4}$$

To identify these instances, we adopted the following nonterminal and terminal node sets. To make instances harder, operators and constants not used in the correct functions shown in Eq. (3) and Eq. (4) are included in the sets.

**Nonterminal nodes:** $V^{NT}$={addition(+), subtraction(-), multiplication(*), division(/), modulo(%), $\sin$, $\cos$, $\exp$, $\ln$}; The function $\sin$, $\cos$, $\exp$ and $\ln$ are unary and should have 1 child nodes, and other operators are binary and should have 2 children nodes.
**Terminal nodes:** $V^T$={$x$, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}; The variable is only $x$ and others are constants.

The fitness function is the sum of error of each sample point of $f_1^{opt}$ or $f_2^{opt}$, and we assume that the population of GP reaches the optimal solution when the fitness value becomes less than 0.01 on Instance (I), and less than 2.0 on Instance (II).

### B. Effectiveness of MSXF

Using the two symbolic regression problem instances, we discuss the performances of MSXFs using UT neighbor and OT neighbor. For the comparison, the performance of 1X is also shown.

In the experiments, the population size was set to 100. The generation alternation model adopted for 1X was the same as that of dMSXF described in section II-C. The objective of the experiments is to examine the effect of crossovers, therefore neither mutations or bloat controlling strategies were applied. The number of offspring $N_{cross}$ generated by each pair of parents was set to 50, 100 and 200. For MSXF, $k_{max}$ was set to 2, 5 and 10. $\mu$ is calculated as $N_{cross}/k_{max}$. The initial temperature $T_0$ was set to 0.1 and updated as $T_{t+1} = 0.1 * T_t$ every 2 generations. The ratio R:D:I was set to 1:1:1 in MSXFs. Initial solutions were generated randomly but keeping the node generation rate as shown in Table I. All nodes included in each kind were generated with equal probability. The number of nodes included in each initial solution was restricted to less than or equal to 25.

TABLE I
Node Generation Rate

| | | |
|---|---|---|
| Nonterminal node ($V^{NT}$) | unary | 0.5 |
| | binary | 0.5 |
| Terminal node ($V^T$) | variable | 0.8 |
| | constant | 0.2 |

Each run was terminated after 40 generations. These parameters were determined from several preliminary experiments.

Table II and Table III show the results on Instance (I) and Instance (II). These results show the success rate, the average of the number of the nodes constituting the obtained tree, and the average depth of trees out of 50 trials. At each trial, the same initial population was used in the GPs.

From Tables II and III, MSXFs perform better than the conventional crossover and find compact trees. In both instances, the size of tree, the number of nodes and the tree depth, becomes smaller in accordance with increase in $k_{max}$ in any settings of parameters. In the easy instance, MSXF completely outperforms 1X regardless of the settings of $k_{max}$. A marked improvement in performance is found, especially in OT distance, in the difficult instance by solving with large

TABLE II
PERFORMANCE OF MSXF WITH THE DISTANCE MEASURES BASED ON UT AND OT ON INSTANCE (I)

| | 1X | MSXF | | | | | |
|---|---|---|---|---|---|---|---|
| | | UT neighbor | | | OT neighbor | | |
| $N_{cross}$=50 | | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 |
| Success Rate | 0.80 | 0.96 | 0.98 | 0.94 | 1.00 | 1.00 | 0.94 |
| Number of nodes | 41.63 | 22.64 | 17.88 | 15.85 | 20.03 | 17.40 | 15.95 |
| Depth of tree | 12.85 | 9.05 | 7.36 | 6.71 | 7.76 | 6.98 | 6.71 |
| $N_{cross}$=100 | | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 |
| Success Rate | 0.74 | 0.94 | 1.00 | 0.98 | 1.00 | 1.00 | 0.98 |
| Number of nodes | 58.12 | 24.60 | 20.03 | 17.05 | 22.05 | 18.76 | 17.00 |
| Depth of tree | 16.73 | 9.55 | 8.05 | 7.01 | 8.28 | 7.34 | 6.79 |
| $N_{cross}$=200 | | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 |
| Success Rate | 0.86 | 0.98 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Number of nodes | 83.72 | 27.93 | 21.06 | 17.93 | 25.16 | 19.97 | 17.81 |
| Depth of tree | 19.46 | 10.41 | 8.54 | 7.42 | 8.93 | 7.77 | 7.07 |

TABLE III
PERFORMANCE OF MSXF WITH THE DISTANCE MEASURES BASED ON UT AND OT ON INSTANCE (II)

| | 1X | MSXF | | | | | |
|---|---|---|---|---|---|---|---|
| | | UT neighbor | | | OT neighbor | | |
| $N_{cross}$=50 | | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 |
| Success Rate | 0.30 | 0.08 | 0.46 | 0.40 | 0.20 | 0.70 | 0.52 |
| Number of nodes | 53.29 | 23.55 | 16.26 | 14.15 | 22.51 | 15.24 | 13.91 |
| Depth of tree | 16.64 | 9.47 | 7.41 | 6.49 | 9.28 | 7.01 | 6.53 |
| $N_{cross}$=100 | | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 |
| Success Rate | 0.50 | 0.24 | 0.56 | 0.56 | 0.30 | 0.84 | 0.84 |
| Number of nodes | 59.01 | 26.40 | 18.05 | 15.02 | 23.61 | 16.96 | 14.61 |
| Depth of tree | 17.51 | 10.33 | 8.22 | 6.96 | 10.11 | 7.67 | 6.71 |
| $N_{cross}$=200 | | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 | $k_{max}$=2 | $k_{max}$=5 | $k_{max}$=10 |
| Success Rate | 0.38 | 0.24 | 0.62 | 0.74 | 0.46 | 0.86 | 0.90 |
| Number of nodes | 110.65 | 30.71 | 19.13 | 15.35 | 27.57 | 17.66 | 15.34 |
| Depth of tree | 30.21 | 11.72 | 8.82 | 7.29 | 11.18 | 8.07 | 7.15 |

populations, but the performances of MSXFs become worse in the case of small $k_{max}$. For the difficult instance, we can see the setting of $k_{max}$ has intensified impact on the performance of MSXFs under a limited population size while the difference in performance is not observed in the easy instance from the perspective of the success rate.

From the comparison between UT neighbor and OT neighbor in MSXF, the latter shows the superior success rate to that of UT neighbor and can find smaller trees. The problem instances adopted in this paper have nonterminal nodes that work unsymmetrically on children nodes, e.g. the subtraction. In such an operator, the numerical result is different between $st_1$-$st_2$ and $st_2$-$st_1$. The definition of OT neighbor considers the symmetric property between children nodes and can treat, for example, the difference between $st_1$-$st_2$ and $st_2$-$st_1$, while UT neighbor ignores the difference as shown in Fig. 2. This would be one of reasons that OT neighbor works well on these symbolic regression problem instances.

### C. The Ratio Among Neighborhood Generation Methods

Table IV and V show the performances in the difficult instance under the different settings of the ratio R:D:I in MSXF with UT neighbor and MSXF with OT neighbor.

The number of offspring $N_{cross}$ generated by each pair of parents was set to 100 and $k_{max}$=5. Other parameters of the GPs were the same as those described in the previous section. Among the settings, the ratio 3:3:4 means that Insertion has slightly high priority to other two operations. The ratio 3:4:3 indicates that Deletion is selected with some high possibility. The last 4:3:3 is biased to Replacement. The ratio 3:3:4 makes the tree larger by the insertion, while the ratio 3:4:3 reduces the size of tree by the deletion.

TABLE IV
COMPARISON AMONG THE SETTINGS OF THE RATIO R:D:I (UT NEIGHBOR)

| R:D:I | 1:1:1 | 3:3:4 | 3:4:3 | 4:3:3 |
|---|---|---|---|---|
| Success Rate | 0.56 | 0.40 | 0.44 | 0.60 |
| Number of nodes | 18.05 | 21.15 | 16.45 | 16.89 |
| Depth of tree | 8.22 | 8.76 | 7.63 | 7.73 |

TABLE V
COMPARISON AMONG THE SETTINGS OF THE RATIO R:D:I (OT NEIGHBOR)

| R:D:I | 1:1:1 | 3:3:4 | 3:4:3 | 4:3:3 |
|---|---|---|---|---|
| Success Rate | 0.84 | 0.72 | 0.80 | 0.82 |
| Number of nodes | 16.96 | 19.80 | 15.36 | 15.17 |
| Depth of tree | 7.67 | 8.33 | 7.10 | 7.18 |

The relation of tree sizes, 3:3:4 $>$ 4:3:3 $>$ 3:4:3, is found in both UT neighbor and OT neighbor, and we can see that the setting of R:D:I affects directly to the tree size. From the perspective of the success rate, the ratio biased to the replacement outperforms other settings.

TABLE VI
COMPARISON BETWEEN MSXF AND DMSXF

| | 1X | MSXF | | dMSXF | |
|---|---|---|---|---|---|
| | | UT neighbor | OT neighbor | UT neighbor | OT neighbor |
| Success Rate | 0.50 | 0.56 | 0.84 | 0.36 | 0.66 |
| Number of nodes | 59.01 | 18.05 | 16.96 | 30.31 | 24.75 |
| Depth of tree | 17.51 | 8.22 | 7.67 | 10.40 | 10.14 |

### D. Comparison of MSXF and dMSXF

Table VI compares the search performance of MSXF and dMSXF in the difficult instance. For a comparative criterion, the result of 1X is shown. In this experiment, the number of offspring $N_{cross}$ generated by each pair of parents was set to 100 and $k_{max}$=5. The ratio R:D:I was set to 1:1:1. Other parameters of the GPs were the same as those described in section IV-B.

From Table VI, OT neighbor is superior to UT neighbor, in dMSXF. dMSXF with OT neighbor also outperforms 1X with obtaining compact trees; however, we can see MSXF is more suitable for optimizing trees with proposed neighborhood structures.

## V. CONCLUSION

In this paper, we introduced MSXF and dMSXF to GP with the neighborhood structures based on the largest common subtree between parents, and examined their performance, using symbolic regression problem instances. Two types of tree structures, the ordered tree and the unordered tree, are examined to extract the largest common trees and generate neighborhoods. From the experiments, it has been shown that the former that consider the symmetric property in trees is suitable for the problem instances. In this paper, we only show results of symbolic regression problem instances but they have been shown to be effective on a simple artificial ant problem in our preliminary experiments. For the examination we chose a conventional crossover to assess the search performance but we should compare them with some state-of-arts. This task is left as a future goal.

## ACKNOWLEDGMENT

This work was partially supported by Grant-in-Aid for Young Scientists (B), 22700158 of the Japan Society for the Promotion of Science (JSPS).

## REFERENCES

[1] Ono, I., Kobayashi, S.: A Genetic Algorithm Taking Account of Characteristics Preservation for Job Shop Scheduling Problems, Proc. of the International Conference on Intelligent Autonomous Systems 5, pp. 711–718 (1998).

[2] Sakuma, J. and Kobayashi, S.: Extrapolation-Directed Crossover for Job-shop Scheduling Problems: Complementary Combination with JOX, Proc. Genetic and Evolutionary Computation Conference 2000 , pp. 973–980 (2000).

[3] Nagata, Y.: New EAX Crossover for Large TSP Instances, Proc. Parallel Problem Solving from Nature IX, pp. 372–381 (2006).

[4] Koza, J. R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992).

[5] Francone, F. D., Conrads, M., Banzhaf, M. and Nordin, P.: Homologous Crossover in Genetic Programming, Proc. Genetic and Evolutionary Computation Conference 1999, Vol. 2, pp. 1021–1026 (1999).

[6] Poli, R., McPhee, N. F. and Rowe, J. E.: Exact Schema Theory and Markov Chain Models for Genetic Programming and Variable-length Genetic Algorithms with Homologous Crossover, Genetic Programming and Evolvable Machines, Vol. 5, No. 1, pp. 31–70 (2004).

[7] Hasegawa, Y. and Iba, H.: A Bayesian Network Approach to Program Generation. Proc. of IEEE Trans. Evolutionary Computation, Vol. 12, No. 6, pp.750–764 (2008).

[8] Beadle, L. and Johnson, C. G.: Semantically Driven Crossover in Genetic Programming, Proc. IEEE World Congress on Computational Intelligence 2008, pp. 111-116 (2008).

[9] Yamada, T. and Nakano, R.: Scheduling by Genetic Local Search with Multi-Step Crossover, Proc. Parallel Problem Solving fron Nature IV , pp. 960–969, (1996).

[10] Ikeda, K., and Kobayashi, S.: deterministic Multi-step Crossover Fusion: A Handy Crossover for GAs, Proc. Parallel Problem Solving fron Nature VII, pp.162–171 (2002).

[11] Hanada, Y., Hiroyasu, T. and Miki, M.: Genetic Multi-step Search in Interpolation and Extrapolation domain, Proc. of Proc. Genetic and Evolutionary Computation Conference 2007, pp. 1242–1249 (2007).

[12] Hanada, Y., Muneyasu, M. and Asano, A.: Effectiveness of Genetic Multi-Step Search on Unsupervised Design of Morphological Filters for Noise Removal, IPSJ Transactions on Mathematical Modeling and Its Applications, vol. 3, no.3, pp.154–165 (2010).

[13] Freisleben, B., Merz, P.: New Genetic Local Search Operators for the Traveling Salesman Problem, Parallel Problem Solving fron Nature IV, pp. 890–899 (1996).

[14] Abe, K., Kawasoe, S., Asai, T., Arimura, H. and Arikawa, S.: Optimized Substructure Discovery for Semi-structured Data, Proc. 6th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 1–14, (2002).

[15] Zaki, M. J.: Efficiently Mining Frequent Trees in a Forest, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80 (2002).

[16] Masuda, S., Yoshikoka, H. and Tanaka, E.; Algorithm for finding one of the largest common subgraphs of two three-dimensional graph structures, Electronics and Communications in Japan, Fundamental Electronic Science, Vol. 81, No. 9, pp. 48–53 (1998).