A Bayesian Network Approach to Program Generation

Yoshihiko Hasegawa and Hitoshi Iba

***Abstract*—Genetic programming (GP) is a powerful optimization algorithm that has been applied to a variety of problems. This algorithm can, however, suffer from problems arising from the fact that a crossover, which is a main genetic operator in GP, randomly selects crossover points, and so building blocks may be destroyed by the action of this operator. In recent years, evolutionary algorithms based on probabilistic techniques have been proposed in order to overcome this problem. In the present study, we propose a new program evolution algorithm employing a Bayesian network for generating new individuals. It employs a special chromosome called the *expanded parse tree*, which significantly reduces the size of the conditional probability table (CPT). Prior prototype tree-based approaches have been faced with the problem of huge CPTs, which not only require significant memory resources, but also many samples in order to construct the Bayesian network. By applying the present approach to three distinct computational experiments, the effectiveness of this new approach for dealing with deceptive problems is demonstrated.**

***Index Terms*—Bayesian network, estimation of distribution algorithm, genetic programming (GP).**

I. Introduction

IN THIS PAPER, we propose a new program evolution algorithm that adopts a probabilistic model as a genetic operator. Our approach, named program optimization with linkage estimation (POLE) [9], [10], employs a Bayesian network as the probabilistic model. This approach uses a special chromosome called the *expanded parse tree* (EPT) to reduce the size of the conditional probability tables (CPTs) that have previously been a significant problem in the application of prototype tree-based approaches.

Genetic programming (GP) [12], [13] is an extension of the genetic algorithm (GA), and is capable of handling programs and functions. Since GP has flexible structural chromosomes, it has been applied to many diverse problems (e.g., robot engineering, financial engineering, bio informatics) and is considered to be a powerful technique for solving such problems.

The main genetic operator in evolutionary algorithms (EAs) is a crossover operator. The well-known theory called schema [7], [11] analyzes the search mechanism in EAs. The schema theory advocates that the solutions (solution candidates) evolve by combining good partial structures by crossover operations, and this combination improves the fitness values of the population. The partial structures are called building blocks. Although the crossover operator is a powerful operator in GA and GP, it sometimes destroys the building blocks because it is randomly carried out regardless of the particular application of the technique.

Recently, a new approach to EAs based on probabilistic techniques has been proposed to overcome the defects of traditional crossover operators. In particular, increasing attention has been paid to probabilistic model building GA (PMBGA) [21], estimation of distribution algorithm (EDA) [15], and probabilistic model building GP (PMBGP) [30] methods.

Since GA and GP have much in common, research on prototype tree-based PMBGPs has taken advantage of many of the techniques devised in PMBGA. However, due to the differences between GA and GP, there arises a critical problem in the simple application of EDA techniques to GP. Since GP uses many symbols during the search (e.g., functions $F = \{+, -, \times, \ldots\}$ and terminals $T = \{x, y, \Re, \ldots\}$), the number of possible symbols at each node becomes very large. When the number of possible symbols is large, the size of CPTs, which express the quantitative relationships between nodes, also becomes large. As a result, many samples are required to construct the Bayesian network (see Section III). Furthermore, all the programs generated by a Bayesian network have to be syntactically correct.

The characteristics of our approach are listed below.

- A Bayesian network is used for estimating the interactions between the nodes.
- A special chromosome called the EPT is employed to reduce the number of possible symbols.

This paper is structured as follows. In the next section, we introduce related studies of PMBGA and PMBGP. Section III describes the effect of large CPT size, and how this problem is overcome by our proposed technique. In this section, we also describe the basic features of the EPT. Details of the proposed algorithm are described in Section IV. In Section V, we compare the performance of our method with that of prior PMBGPs, selecting three benchmark tests for experiments. We analyze and discuss the results obtained in the experiments in Section VI. Finally, we conclude our paper in Section VII.

II. Related Works

EAs based on probabilistic models were first constructed in the field of binary GA. Binary PMBGAs can be classified into three groups: univariate models, bivariate models, and multivariate models. In these fields, many algorithms have been proposed. In particular, multivariate algorithms, which use Bayesian networks [20] [e.g., Bayesian optimization algorithm (BOA) [22], estimation of Bayesian network algorithm

Manuscript received November 27, 2006; revised May 8, 2007 and September 18, 2007. First published March 10, 2008; current version published December 2, 2008.

The authors are with the Department of Frontier Informatics, Graduate School of Frontier Sciences, University of Tokyo, Kashiwa, Chiba 277-8561, Japan (e-mail: hasegawa@iba.k.u-tokyo.ac.jp; iba@iba.k.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TEVC.2008.915999

(EBNA) [4], and learning factorized distribution algorithm (LFDA) [19]] are considered to be strong algorithms because of their model flexibility.

Recently, an increasing level of research has been undertaken on PMBGPs. PMBGPs can be broadly classified into two groups: algorithms based on a prototype tree, and those based on grammar-guided GPs.

- Prototype tree-based PMBGP:
  probabilistic incremental program evolution (PIPE) [28], estimation of distribution programming (EDP) [35], extended compact genetic programming (ECGP) [30], and BOA programming (BOAP) [16], [17].
- Context-free grammar-based PMBGP:
  Stochastic grammar-based GP (SG-GP) [26], program evolution with explicit learning (PEEL) [32], grammar model-based program evolution (GMPE) [33], grammar transformations in an EDA [2], and Bayesian automatic programming (BAP) [27].

An advantage of the prototype tree-based approach is that it is easier to apply techniques devised in the field of EDA. For example, PIPE extended population-based incremental learning (PBIL) [1] to program evolution, while ECGP combined extended compact GA (ECGA) [8] with GP. However, there are two problems that are not important in the field of EDA, but are pertinent to program evolution applications: the number of symbols and the syntactic correctness (Section III). Our approach overcomes these two problems by employing the EPT.

Our prototype tree-based approach is designed to be of superior performance to previous approaches of this kind. Previous approaches have used univariate models (PIPE), adjacent models (EDP), and the joint distribution model (ECGP). However, from the viewpoint of grasping more general interactions between the nodes, an estimation of the Bayesian network can be considered to be a superior approach. The BOAP method is an existing program evolution method that adopts a Bayesian network. Although BOAP, which employs a chromosome called *zigzag tree* based on a lambda function and is a completely different expression from the standard parse tree (SPT) or the EPT, estimates the Bayesian network, the use of BOAP may lead to the problem of huge CPT size. Furthermore, new individuals generated by the Bayesian network in BOAP are not necessarily syntactically correct. In order to evaluate these incorrect individuals, BOAP uses an heuristic approach. By contrast, our approach employs a Bayesian network and the EPT. Any interactions present can be estimated from a smaller CPT size, and all the programs generated by our approach are syntactically correct.

III. Problem of Large CPT

A. Required Sample Size

The PIPE, EDP, and ECGP methods use the SPT as a simple GP. In these algorithms, all programs are handled as a full $a_{\max}$-ary tree [Fig. 1(a)], where $a_{\max}$ is the maximum arity of the functions. The positions of each node are superposed, and the probability distributions are constructed by counting the frequencies of the symbols.

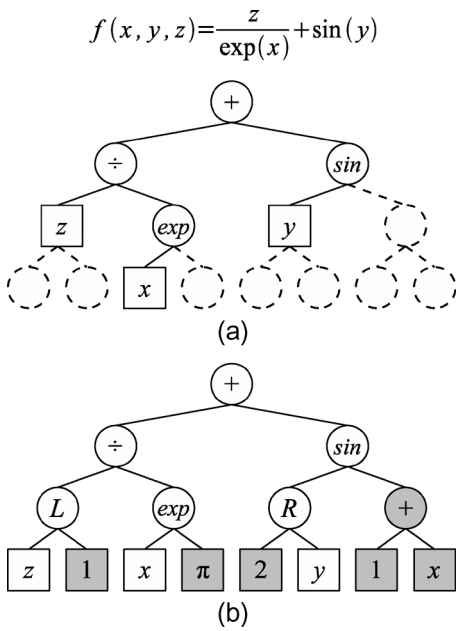

Fig. 1. (a) SPT and (b) the EPT. The gray nodes represent introns. Circles and squares represent functions and terminals, respectively. Both trees are syntactically identical.

In the SPT, both functions and terminals are located at branches in the tree. For example, although terminal z and function $\exp$ in Fig. 1(a) are at the same depth, these nodes are of a different type. The size of a traditional CPT expressing the quantitative relationships among nodes is given by (1), which also represents the number of free parameters in the Bayesian network

$$K = \sum_{i=0}^{n-1} \|\Pi_i\| \left(\|X_i\| - 1\right) \tag{1}$$

where X_i is an ith variable (node), $\|X_i\|$ is the number of possible states of X_i, Π_i represents a set of parent nodes of X_i in Bayesian network, and n denotes the number of nodes. In the case of the SPT, $\|X_i\| = |F \cup T|$ and $\|\Pi_i\| = |F \cup T|^{|\Pi_i|}$.

GP normally uses about ten symbols during the search, in which case $\|X_i\| = |F \cup T| = 10$. In contrast, binary GA uses only two symbols, $\|X_i\| = 2$. This example shows that the size of the CPT for PMBGP based on the SPT is much larger than that for binary PMBGA.

The relation between the sample size and convergence of the learned network has been examined in [5]. They showed that the number of samples required to learn ϵ-close approximation with confidence δ [(2)] is of the order given in (3). In their work, they calculate the upper bounds of sample complexity for an MDL-based metric (MDL, BIC, and AIC) that satisfies (2). In this equation, B^* denotes the target distribution from which training samples are drawn, $\mathrm{Lrn}()$ represents the network learning routine, while function KL is the Kullback–Leibler distance

$$P\left(\mathrm{KL}\left(P_{B^*} \| P_{\mathrm{Lrn}()}\right) > \epsilon\right) < \delta \tag{2}$$

$$O\left(\left(\frac{1}{\epsilon}\right)^{\frac{4}{3}} \log\frac{1}{\epsilon} \log\frac{1}{\delta} \log\log\frac{1}{\delta}\right). \tag{3}$$

This expression, which is valid for noisy cases, shows that the main factor determining the upper bound of sample complexity is not the CPT size, but is the sample noise when given a sufficiently large sample size.

In EAs, population size is given by $M \sim 1000$. If we adopt the selection rate $P_s \sim 0.1$, the sample size is about 100. With these values, we cannot ignore the effect of CPT size. In a Bayesian network, each value in the CPT has to be estimated by given samples (sample size). If the CPT size is large when the number of samples is small, each parameter is not estimated precisely. For example, if $\|X_i\| = 10$ and a node has two parents in the Bayesian network, the CPT size for this node is 900. This value is larger than the sample size, and so we can suppose that the parameters cannot be precisely estimated.

In the present study, we approximately investigate the effect of CPT size from the viewpoint of confidence intervals. We approximate the estimation of parameters in CPT using the multinomial distribution.

For the case of sample size N and the number of possible cases K, let $\mathbf{n} = (n_0, n_1, \ldots, n_{K-1})$ denote the frequencies of each case in a multinomial distribution ($N = \sum_i n_i$), and $\mathbf{p} = (p_0, p_1, \ldots, p_{K-1})$ represent the parameters for the multinomial distribution ($1 = \sum_i p_i$).

The confidence intervals of $100(1-\alpha)$ percent (α is a constant) for a multinomial distribution are given by (4) [25]. In (4), $p_j^{\pm}$ denotes the upper $(+)$ and lower $(-)$ bounds of the intervals, Δ_j is given in (5) and $\chi^2_{\alpha/K}(1)$ is a $100\alpha/K$ percentage point (upper probability) for a chi-squared distribution of one degree of freedom

$$p_j^{\pm} = \frac{\chi^2_{\frac{\alpha}{K}}(1) + 2n_j \pm \sqrt{\Delta_j}}{2\left(N + \chi^2_{\frac{\alpha}{K}}(1)\right)} \tag{4}$$

$$\Delta_j = \chi^2_{\frac{\alpha}{K}}(1)\left[\chi^2_{\frac{\alpha}{K}}(1) + \frac{4n_j(N-n_j)}{N}\right]. \tag{5}$$

We wish to estimate the lower bound on the sample size that is required to make the confidence intervals smaller than $\eta(n_j/N)$, where η denotes a constant parameter ($\eta \ll 1$). Thus, the sample size N has to satisfy the condition

$$p_j^{+} - p_j^{-} \leq \frac{n_j}{N}\eta. \tag{6}$$

For the case $n_j = N/K (j \in \{0, 1, \cdots, K-1\})$, (6) gives the lower bound on N, in which case

$$N \geq -\frac{(2+\eta^2-2K)\chi^2_{\frac{\alpha}{K}}(1)}{\eta^2} + \chi^2_{\frac{\alpha}{K}}(1)\frac{\sqrt{(\eta^2(-2+K)^2+4(-1+K)^2)}}{\eta^2}. \tag{7}$$

By using the condition $\eta^2 \ll 1$, (7) can be simplified to the expression in (8), in which we have used $\chi^2_{\alpha}(1) = Z^2_{\alpha/2}$, where Z_α is a 100α percentage point (upper probability) in a normal distribution

$$N \geq \frac{4\chi^2_{\frac{\alpha}{K}}(1)}{\eta^2}(K-1) = \frac{4Z^2_{\frac{\alpha}{2K}}}{\eta^2}(K-1). \tag{8}$$

As can be seen from (8), the number of samples required to reliably estimate the parameters is almost proportional to K. Thus, the estimation of parameters in large CPTs requires many samples. Since the size of the CPT increases rapidly with the number of possible symbols, the required sample size thus increases dramatically with the number of possible symbols. Larger numbers of samples in turn require a larger number of fitness evaluations.

In the present approach, we use a special chromosome called the EPT to overcome the above problem. In the next section, we describe this chromosome in detail.

B. Expanded Parse Tree (EPT)

An EPT is an expression technique for tree structures (programs, functions, etc.) first proposed for an algorithm named evolutionary programming with introns (EP-I) [34]. The EPT was originally devised to apply GA crossover to GP, and its use enables an application of GA operators to GP.

GP and GA are both based on the same concept of natural evolution, and consequently have much in common. GA uses linear arrays and crossover is performed by swapping the subarrays of two individuals. On the other hand, GP adopts tree structures and crossover is undertaken by exchanging subtrees. Tree structures can be converted to linear arrays (and also fixed length by inserting *empty* nodes) as in GA. However, the node types (function or terminal) of each position differ among individuals, and this difference gives rise to syntactic destruction if GA-style crossover is applied to GP.

If the length of chromosomes is fixed and the node types (function of terminals) at each position are fixed among all individuals, GA crossover can be applied. EP-I extended the SPT to satisfy the above requirement by inserting selector nodes and adding introns beneath unused arguments. In EP-I, this extended expression is called an EPT. Fig. 1(b) shows an example of the EPT, which is syntactically identical to the SPT [Fig. 1(a)] used in traditional GP. An EPT is expressed by a full $a_{\max}$-ary tree ($a_{\max}$ is the maximum arity of the functions). The EPT inserts selector operators (L and R) into the SPT in order to ensure that terminal nodes are positioned at the leaves of the full $a_{\max}$-ary tree. In this figure, L and R denote selector operators and operate according to $R(x, y) = y$ and $L(x, y) = x$. However, with these extra operators, the length of individuals differs because there are operators which have fewer arity than the maximum arity [in Fig. 1, sin is of one arity, which is fewer than 2, the maximum arity]. EP-I attaches introns to the unused parameters in order to overcome this problem. For instance, exp is converted to the 2-arity function evaluated as $\exp(x, y) = \exp(x)$. In Fig. 1(b), gray nodes stand for introns, which will not be interpreted during evaluation. In this way, a SPT of maximum arity $a_{\max}$ and maximum depth D_P can be converted to a full $a_{\max}$-ary EPT with a depth of D_P.

In the EPT, node types at each position are identical for all the individuals. In Fig. 1 for example, nodes represented by circles are functions while nodes represented by squares are terminals. We can see that all function nodes are positioned at the branches and all terminals at leaves in the tree. GA-style crossover can then be applied to the EPT.

In the previous section, it has been noted that the size of the CPT is very large when applying a Bayesian network to a SPT. This is because we have to consider the possibility of both F and T at each position in the SPT. This is in contrast to the EPT of the present approach, in which function nodes are always positioned at the branches, and the terminal nodes at the leaves [Fig. 1(b)].

The number of possible states of a SPT is given by $|T \cup F|$. At the leaves in a SPT, the number of possible states is given by $|T|$. In an EPT, the corresponding quantities are given by $|F \cup \{L\}| = |F| + 1$ (we only use L for a selector node) at the branches in the tree and $|T|$ in the leaves. Clearly, the number of possible states in the EPT is smaller than that in the SPT. Thus, the number of free parameters K can be represented compactly in the EPT. As can be seen with reference to (1), the CPT size depends on the number of possible symbols. Therefore, by using the EPT, the size of the CPT can be reduced.

In our approach, we employ the EPT to express individuals. In EP-I, both L and R nodes are used for selector operators. However, in the interests of restricting CPT size, we use only the L operator. Furthermore, the restriction of selector nodes to L not only reduces CPT size, but also reduces noise. For example, let us consider the expression of the function $f(x) = x$ in the EPT. In the case of using both L and R, a terminal x can appear at any position in the leaves. On the other hand, if we only use L for a selector node, a terminal x can only appear in the leftmost position in the leaves. This will facilitate the construction of the Bayesian network.

IV. Proposed Algorithm

Algorithm 1: POLE

Inputs:

A set of parameters (Table I), Evaluation function

Outputs:

Best individual

$g = 0$;
$\mathcal{D}_0 =$ Generate M random individuals;
$terminate = false$;
while $(terminate)$ {
  $evaluate(\mathcal{D}_g^{sel})$;
  $\mathcal{D}_g^{sel} =$ Select better $M \times P_s$ individuals from $\mathcal{D}_g$;
  $\mathcal{G}_g =$ Find the network with K2 Algorithm;
  $\Theta_g =$ Estimate parameter using $\mathcal{D}_g^{sel}$;
  $\mathcal{D}_g =$ Generate M individuals using $\mathcal{B}_g = \langle \mathcal{G}_g, \Theta_g \rangle$;
  $g = g + 1$;
  if (terminate criteria are met) {
    $terminate = true$;
  }
}

TABLE I
Main Parameters for POLE. The Missing Concrete Values (Denoted by the "−" Symbols) Are Problem Dependent

Symbol	Meaning	Value
M	Population size	−
G	Generation to run	−
D_P	Maximum tree depth	−
P_s	Selection rate	0.1 (MAX, DMAX) 0.2 (Royal Tree)
R_P	Parent range	2 (MAX, DMAX) 4 (Royal Tree)
k	The number of max incoming edge	∞
P_e	Elite rate	0.005
P_F	Function selection rate at the init.	0.8

In this section, we explain the details of the proposed POLE. A flowchart of POLE is described below, which is identical to that of other PMBGAs and PMBGPs.

Step 1) Initialization of individuals.
Initial individuals are generated randomly.

Step 2) Evaluation of individuals.
Evaluate every individual and assign a fitness value.

Step 3) Selection of individuals.
According to the fitness values, individuals used for construction of the Bayesian network and the estimation of parameters are selected. The number of individuals selected is given by $M \times P_s$, where M is the population size and P_s is the selection rate. Any selection methods can be employed—we use a truncation selection.

Step 4) Construction of Bayesian network.
A Bayesian network is constructed using the samples, and parameters are also estimated. In our implementation, the networks are constructed from scratch at each generation.[1]

Step 5) Generation of new individuals.
New individuals are generated using the constructed Bayesian network. If we want to use an elitist strategy, the best $M \times P_e$ individuals are copied from the previous generation, where P_e denotes the elite rate.

Steps 2–5 are repeated until the termination criteria are met. The pseudocode for POLE is described in Algorithm 1. In this code, the subscript g denotes variables of generation g.

A. Initialization of Individuals

In the initialization phase, POLE initializes tree structures. In traditional GP, there are two well-known initialization methods: **Grow** (random shape) and **Full** (maximum-depth tree). POLE can use both initialization methods. When using the Grow method in POLE, a random function node is selected with a

[1]In preliminary experiments, we tried updating the network every $2 \sim 5$ generations to reduce the computational cost of the algorithm, but found that this greatly degraded the performance of the search.

probability of P_F and a terminal node with a probability of $1-P_F$ from the root node. If a function is selected at the branch of the EPT, then the function is moved to the leaves by inserting selector nodes L. At the leaves of the full tree, select a terminal node unconditionally. With these procedures, Grow can be simulated in the EPT. If the function selection probability P_F is 1, then this technique is identical to Full. In all the experiments described in the present study, Grow is used.

B. Scoring the Network

In order to estimate the underlying interactions in POLE, we have to search for the network that gives the maximum posterior probability of $\mathcal{G}$ given data $\mathcal{D}$, that is argmax $P(\mathcal{G}|\mathcal{D})$.

We use the Bayesian information criteria (BIC) [31] approximation to evaluate the posterior probability $P(\mathcal{G}|\mathcal{D})$. BIC is also used in EBNA and LFDA. The BIC is expressed by

$$\mathrm{BIC}(\mathcal{G},\mathcal{D}) = \sum_{i,j,k} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{K}{2} \log N \tag{9}$$

where $\mathcal{G}$ and $\mathcal{D}$ stand for the network and the data, respectively, K is the number of parameters, which here also means CPT size, N_{ijk} is the number of times that X_i is in its kth state and Π_i is in its jth state, and $N_{ij} = \sum_k N_{ijk}$.

C. Construction of Networks

Algorithm 2: K2 Algorithm used in POLE

Inputs:

The number of maximum incoming edges k, Data $\mathcal{D}$, Parent candidacies $Cand(X_i)$

Outputs:

The network which approximately maximize $\mathrm{BIC}(\mathcal{G},\mathcal{D})$

```
for (i = 0; i < n; i++) {
    G = empty network;
    P_old = BIC(G, D);
    findmore = true;
    while (findmore && |Π_i| < k) {
        Z = node in Cand(X_i) − Π_i, that maximizes
                BIC(addedge(G, Z → X_i), D);
        P_new = BIC(addedge(G, Z → X_i), D);
        if (P_new > P_old) {
            P_old = P_new;
            G = addedge(G, Z → X_i);
        }
        else {
            findmore = false;
        }
    }
}
```

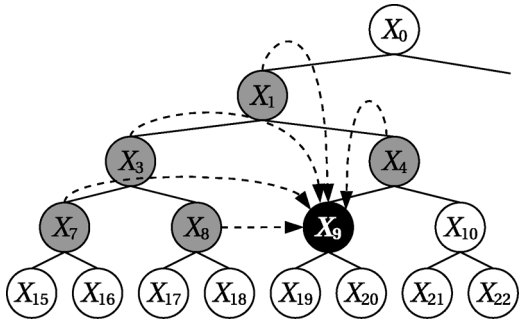

Fig. 2. The parent candidates for X_9 for the case $R_P = 2$ are represented by the gray nodes.

Construction of networks from given data is known to be a computationally expensive task. Many algorithms have hitherto been proposed; we employed a combination of the BIC and K2 algorithm [3]. The K2 algorithm can relatively quickly construct a Bayesian network, because this algorithm orders the variables and limits the number of parent candidates. In the K2 algorithm, the parent candidates of the ith node are nodes that have smaller indices.

Although the K2 algorithm can relatively quickly construct a Bayesian network, since the GP requires many nodes during the search, the K2 algorithm may not be fast enough to produce a satisfactory required runtime for the overall method. In the tree structure, interactions between neighboring nodes may be stronger than those between distant nodes. Our algorithm takes advantage of this assumption to enable a faster network construction.

Let $U(X_i, R_P)$ be a node positioned at R_P levels above X_i. In POLE, the nodes that satisfy the two following conditions are candidates to be parents of X_i (order of the nodes is based on breadth-first traversal).

- Nodes that belong to the subtree rooted at $U(X_i, R_P)$.
- Nodes whose indices are smaller than i.

Fig. 2 shows an example of the construction of parent candidates according to the present approach. The gray nodes in this figure represent candidates to be parents of $X_9(R_P = 2)$.

The K2 algorithm used in our approach is defined in Algorithm 2 in pseudocode. In this pseudocode, $Cand(X_i)$ is the set of nodes that are parent candidates for node X_i. The function $addedge(\mathcal{G}, A \rightarrow B)$ returns a network where edge $A \rightarrow B$ is added to $\mathcal{G}$.

Fig. 3(a) shows an example of a Bayesian network structure constructed by the POLE method.

D. Generation of New Individuals

New individuals are generated from the network $\mathcal{G}$ and CPT Θ. First, nodes that have no parent nodes are determined using the probability distribution. Next, nodes whose parents are already fixed are generated using the conditional probabilities. Since the Bayesian network is a directed acyclic graph (DAG), all nodes can be decided in turn. Fig. 3(b) shows an example of the node generation process corresponding to the network shown in Fig. 3(a).

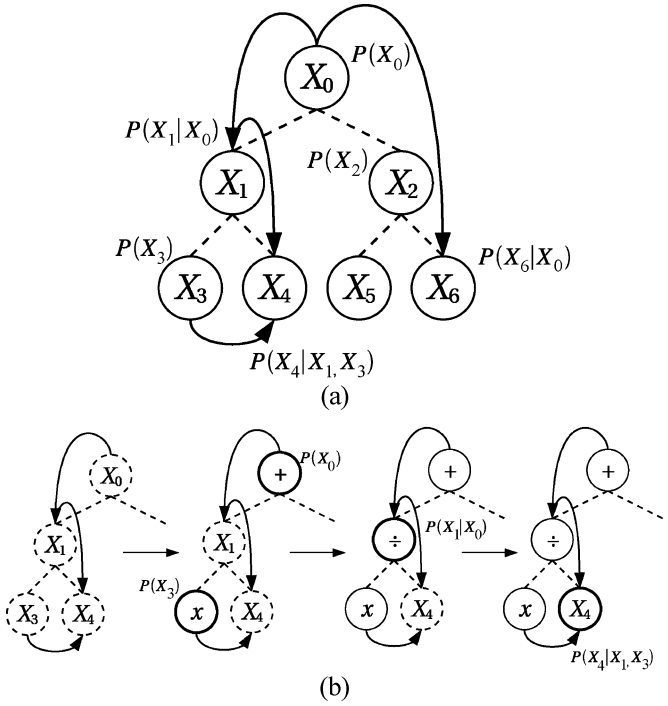

Fig. 3. (a) An example of a Bayesian network structure and (b) the generation procedure of programs.

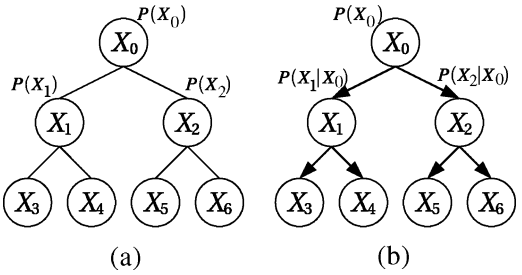

Fig. 4. Network structures of (a) the PIPE model and (b) the EDP model.

V. Comparative Experiments

In order to investigate the search performance of our approach, we applied POLE to three problems: the MAX problem, the deceptive MAX (DMAX) problem and the royal tree problem. The main parameters are listed in Table I.

In the experiments, we compared the performance of the four models listed below to compare and contrast the performance of the present POLE method with the other three methods.

- **POLE**.
  This is the proposed method of this study. To estimate interactions between nodes, a Bayesian network is constructed. The maximum number of incoming edges per node is unlimited. Grow is used to initialize individuals.
- **Univariate model (PIPE model)**.
  This algorithm is a univariate case of POLE. No networks are constructed in this algorithm, and no interactions between the nodes are estimated [Fig. 4(a)]. The search mechanism resembles that of PIPE. Grow is again used to initialize individuals.
- **Adjacent model (EDP model)**.
  This model constructs parent and child relationships in the tree structure according to EDP. However, in contrast to traditional EDP, this algorithm uses the EPT. As illustrated in Fig. 4(b), EDP does not estimate the network using promising individuals, but instead uses a fixed network. This model estimates $P(X_{ij}|X_i)$, where X_{ij} is a jth child node of X_i. Grow is again used to initialize individuals.
- **Simple GP (SGP)**.
  This algorithm is a simple implementation of GP. In the experiments, $P_e = 0.005$, $P_c = 0.995$, $P_m = 0$, and $P_r = 0$ are used. Crossover points are selected with bias. In applying the crossover operator to two individuals, we selected the first crossover point from function nodes with a probability of 0.9, and from terminals with a probability of 0.1. The second crossover point is selected under the condition that the depth of both individuals does not exceed the depth limitation. The tournament size is set to $T_s = 2$. Once again, Grow is used to initialize individuals.

In all algorithms, the population size is determined in the following way. We start from $M = 100$, and increase the population size by $\sqrt[5]{10}$ times ($M = 100, 160, 250, 400, 630, 1000, 1600, \ldots, \max M = 25000$). For each population size, we execute 20 runs. If the algorithm obtains the solution optimum 20 times from the 20 runs, we stop increasing the population size and calculate the average number of fitness evaluations F_{avr}. This method of determining the population size has previously been employed in [4] and [22]. Since F_{avr} tends to have a large standard deviation, we calculate F_{avr} ten times, and conduct a t-test using the average of F_{avr}.

All algorithms are terminated when they obtain the optimum solution. Furthermore, we also adopted another termination criterion. Because POLE, the univariate model and the adjacent model tend to converge faster than SGP, these algorithms stop when the best fitness value at each generation is not improved for ten consecutive generations. In contrast, SGP stops when the fitness values are not improved from generation g to generation $2g (g > 10)$.

We performed a t-test (Welch, two-tailed) for each experiment. We calculate the P-value for the obtained data (e.g., the average of POLE and the average of Univariate) to exhibit any statistically significant differences between the results of the different approaches.

A. Experiment 1: MAX Problem

1) Problem Description: The MAX problem [6], [14] is a benchmark test for investigating the mechanism of GP crossover, and is widely used as a benchmark test for PMBGP [33], [35]. An objective of the MAX problem is to find the function that returns the largest value under the limitation of a maximum tree depth. In this problem, three symbols are used

$$F = \{+, \times\}, \quad T = \{0.5\}. \tag{10}$$

An optimal solution can be obtained with following procedure. First, create the value "2" using four "0.5" symbols and three "+" symbols. Then, multiply the created "2"s using "×" symbols. The optimal value for a given maximum depth D_P can thus be represented by $2^{2^{D_P-3}}$.

2) Results and Analysis: Table II presents the number of fitness evaluations and the corresponding standard deviations. Fig. 5 visualizes some of the data of Table II and describes the relationship of tree size versus the average number of fitness evaluations for the four models. Here, tree size means the number of nodes contained in the optimum structure. In the MAX problem, tree size can be represented by $2^{D_P} - 1$. As

TABLE II
THE NUMBER OF FITNESS EVALUATIONS FOR THE MAX PROBLEM. VALUES OF THE STANDARD DEVIATION (STDEV) FOR EACH CASE ARE GIVEN IN PARENTHESES

		$D_P = 5$	$D_P = 6$	$D_P = 7$
POLE	Average	487	1758	5588
	Stdev	(21)	(59)	(75)
Univariate	Average	497	1786	5688
	Stdev	(21)	(108)	(46)
Adjacent	Average	581	2496	7401
	Stdev	(28)	(186)	(213)
SGP	Average	1209	3962	11141
	Stdev	(268)	(825)	(3906)

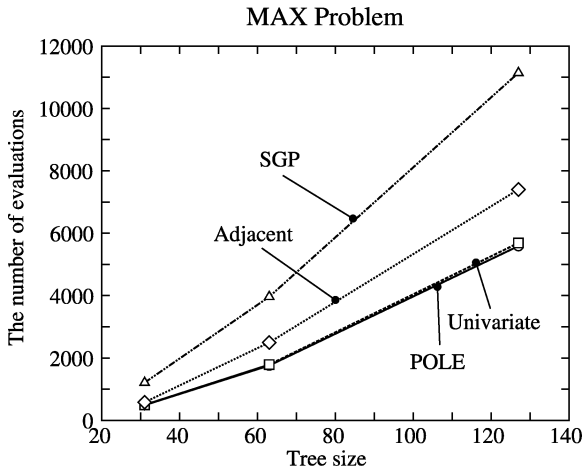

Fig. 5. The number of fitness evaluations required by the four methods for the MAX problem.

can be seen from Fig. 5, SGP requires more evaluations than the other three models. POLE returns the best performance, although its results are indistinguishable from those of the Univariate model on the scale used to plot Fig. 5. Table III details the results of the t-tests. According to Table III, the P-value for the POLE and the Univariate averages is smaller than 1%. This means that the difference in the averages of POLE and the Univariate is statistically significant (at 1% significant level). Although the difference between POLE and the Univariate is statistically significant, the difference is very small. This is because the MAX problem has no deceptiveness, and the optimum can be obtained in a hill-climbing fashion using the Univariate model. The Adjacent model performs better than SGP, but has a poorer performance than either POLE or the Univariate model. Since there are no interactions between nodes in the MAX problem, unnecessary interaction estimation reduces the search performance. In contrast, since our approach estimates the interactions using promising individuals, unnecessary relations are not estimated.

B. Experiment 2: Deceptive MAX (DMAX) Problem

1) Problem Description: We also applied the proposed method to a deceptive MAX problem (DMAX problem), which

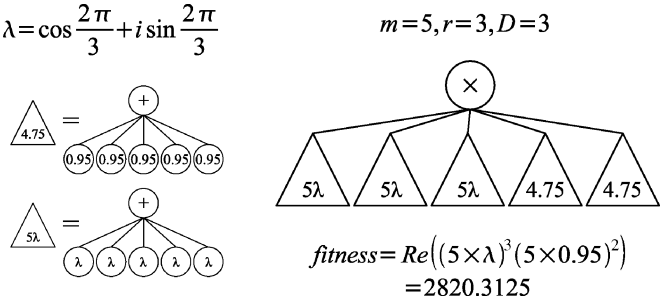

Fig. 6. The optimum individual for the DMAX problem ($m = 5$, $r = 3$, $D_P = 3$).

TABLE III
t-TEST. THE VALUES REPRESENT P-VALUES FOR THE MAX PROBLEM

	$D_P = 5$	$D_P = 6$	$D_P = 7$
POLE vs Univariate	3.30×10^{-1}	4.86×10^{-1}	2.69×10^{-3}
POLE vs Adjacent	1.85×10^{-7}	1.47×10^{-7}	3.06×10^{-11}
POLE vs SGP	1.27×10^{-5}	1.37×10^{-5}	1.50×10^{-3}

is an extension of the MAX problem. Because, as shown above, the original MAX problem does not have deceptiveness, this problem is very easily solved by PMBGPs that do not consider interactions. We extended the MAX problem by adding deceptiveness, and we call this extended problem the *deceptive MAX* problem (DMAX) problem.

The main objective of the DMAX problem is identical to that of the original MAX problem: to find the functions which return the largest *real* value under the limitation of a maximum tree depth D_P. However, the symbols used in the DMAX problem are different from those used in the MAX problem. The DMAX problem uses the symbols given in (11), where add_m and $\mathrm{multiply}_m$ are m-arity functions defined by (13). Terminal λ is a complex value defined by (12). A fitness value of an individual is simply the real part of its function; if the value of a function is $a + bi$ (where $a, b \in \mathbb{R}$, $i = \sqrt{-1}$), then its fitness value is a. The DMAX problem can be represented by three parameters, m (arity), r (power), and D_P (maximum tree depth)

$$F = \{\mathrm{add}_m, \mathrm{multiply}_m\}$$
$$T = \{\lambda, 0.95\} \tag{11}$$
$$\lambda^r = 1,\ \lambda \in \mathbb{C},\ r \in \mathbb{N} \tag{12}$$

$$\mathrm{add}_m(a_0, a_1, \ldots, a_{m-1}) \equiv \sum_{i=0}^{m-1} a_i$$
$$\mathrm{multiply}_m(a_0, a_1, \ldots, a_{m-1}) \equiv \prod_{i=0}^{m-1} a_i \tag{13}$$

$$\mathrm{fitness}_i = \mathrm{Re}(\mathrm{individual}_i). \tag{14}$$

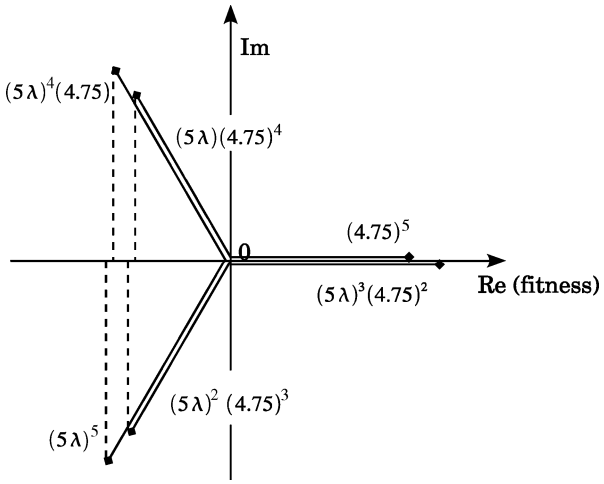

Fig. 7. The optimum individual for the DMAX problem ($m = 5$, $r = 3$, $D_P = 3$) plotted in the complex plane.

The difficulty of this problem depends on these three parameters. In this paper, we experimented with $m = 5, r = 3, D_P = 3$ and 4, which correspond to problems possessing a strong deceptiveness. With this setup, $\lambda = \cos(2\pi/3) + i\sin(2\pi/3)$.

Let us consider the optimum value (the maximum value) for the DMAX problem with $m = 5$, $r = 3$, and $D_P = 3$. First, add λ with add_5 to make 5λ. Then, multiply this value using $\mathrm{multiply}_5$ to obtain the value $(5\lambda)^5 = 5^5\lambda^5 = 5^5\lambda^2$. However, $\mathrm{Re}(5^5\lambda^2)$ is a negative value, and it is not a good solution. So, two values out of the five values multiplied have to be real values. The values 5×0.95 are substituted for 5λ for two cases, and the maximum value is then $(5\lambda)^3(0.95 \times 5)^2 = 2820.3125$. For $D_P = 4$, the maximum value is $(5\lambda)^{24}(0.95 \times 5) = 2.83 \times 10^{17}$. Fig. 7 provides visualization in the complex plane of the values that have large absolute values $(D_P = 3)$. The horizontal axis represents the real axis, which also represents the fitness value. Thus, the value that is located at the rightmost position is the global optimum. As can be seen, the replacement of one 0.95×5 by 5λ increases the argument by 120° in the complex plane. So, if such a replacement is undertaken three times, the value returns to the same phase but has a larger absolute value.

In the normal MAX problem, it is very easy to get the maximum value by multiplying $(0.5+0.5+0.5+0.5)$. By contrast, the optimum of the DMAX requires the combination of complex values and real values. The DMAX is thus considered to be a more difficult problem than the MAX problem.

2) Results and Analysis: Table IV describes the number of fitness evaluations required by the four methods, and the corresponding standard deviations. Fig. 8 is a visualization of Table IV. The results of the Univariate model and the Adjacent model are omitted, because both these algorithms were unable to achieve a probability of success $= 100\%$ for $D_P = 4$ even with $M = 25000$.

As shown in Table IV, SGP requires ten times more fitness evaluations than POLE. SGP cannot effectively solve the DMAX problem because of the deceptive fitness landscape of the problem. Fig. 10(a) depicts a local optimum, while

TABLE IV
THE NUMBER OF FITNESS EVALUATIONS FOR THE DMAX PROBLEM. VALUES OF THE STANDARD DEVIATION (STDEV) FOR EACH CASE ARE GIVEN IN PARENTHESES

		$D_P = 3$	$D_P = 4$
POLE	Average	1570	124531
	Stdev	(155)	(23403)
Univariate	Average	1409	–
	Stdev	(107)	–
Adjacent	Average	1440	–
	Stdev	(119)	–
SGP	Average	11105	1632159
	Stdev	(6958)	(645696)

TABLE V
t-TEST. THE VALUES REPRESENT P-VALUES FOR THE DMAX PROBLEM

	$D_P = 3$	$D_P = 4$
POLE vs Univariate	1.58×10^{-2}	–
POLE vs Adjacent	5.10×10^{-2}	–
POLE vs SGP	1.89×10^{-3}	4.14×10^{-5}

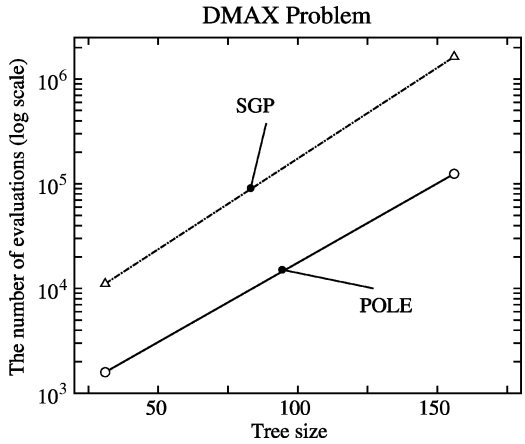

Fig. 8. The number of fitness evaluations for the DMAX problem. The results of the Univariate model and the Adjacent model are omitted, because these models could not obtain the optimum solution for $D_P = 4$ and (tree size $= 156$).

Fig. 10(d) depicts the global optimum of the DMAX problem ($m = 5$, $r = 3$, and $D_P = 3$). When an individual represented by (a) transforms to (d) by a crossover operator, the individual has to go through the intermediate structures represented by (b) and (c). However, the fitness values of (b) and (c) are negative, and the probability that these intermediate structures are selected in the next generation is consequently very low. Fig. 11 shows a transition of local optimum (a) to global optimum (d) represented in the complex plane. Applying the crossover operator once to each individual, the arguments are increased by 120°. By applying crossover three times to the local optimum

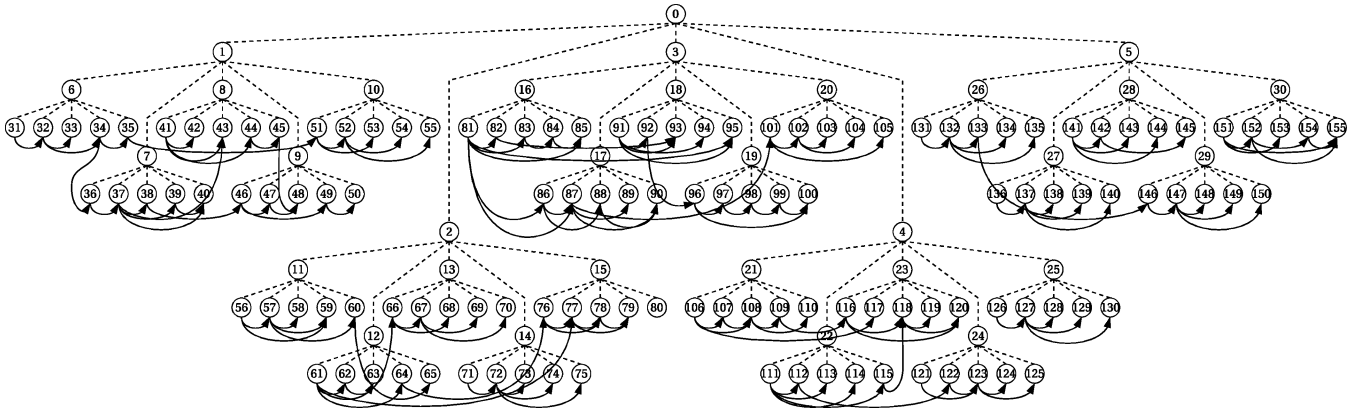

Fig. 9. An example of a network for the DMAX problem ($M = 4000$, $D_P = 4$). This network is for generation $= 15$.

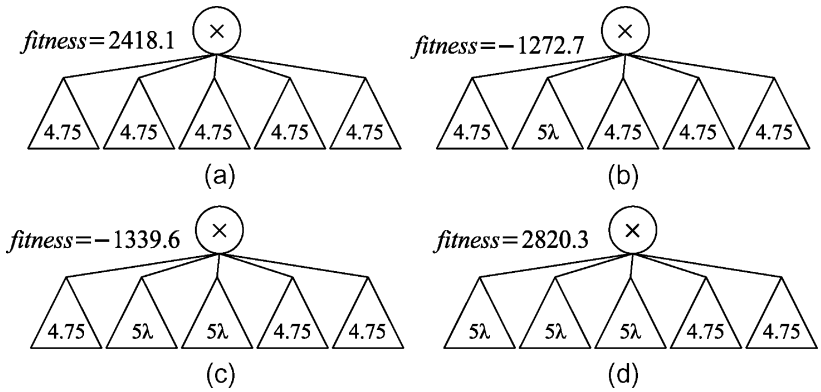

Fig. 10. (a) The local and (d) global optima for the DMAX problem ($m = 5$, $r = 3$, and $D_P = 3$). Parts (b) and (c) are intermediate structures.

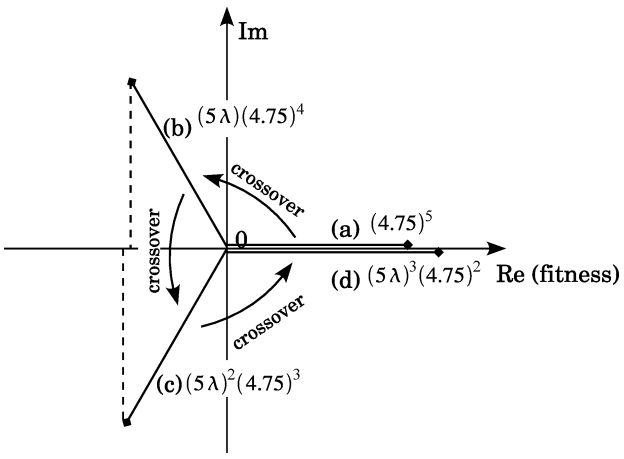

Fig. 11. An abstract image of the DMAX fitness landscape when using GP. The horizontal axis represents a real axis and also measures fitness. Points (a) $\sim$ (d) shown here in the complex plane correspond to respective parts of Fig. 10.

(a), this individual can transform to the global optimum (d). This figure shows that SGP requires many fitness evaluations to obtain the optimum solution once trapped at a local optimum.

The Univariate and the Adjacent models did not estimate the interactions in the DMAX problem, and so these algorithms soon destroyed the building blocks and, as a result, could not obtain the optima for $D_P = 4$. Since our approach effectively estimates the interactions between nodes, POLE can obtain the optimum value with fewer fitness evaluations. Thus, we conclude that POLE is a highly effective method for solving the DMAX problem in comparison with the previous program evolution methods considered here.

We observed the estimated interactions found by POLE for the DMAX problem. Fig. 9 represents the Bayesian network constructed during the search. This is a network of a randomly selected run that found the optimum at generation $= 17$ ($M = 4000$ and $D_P = 4$). We can see that POLE estimates the interactions between terminal nodes. In this problem, the interactions are mainly among terminal nodes. There are two building blocks in this problem: that composed of five λs and that composed of five 0.95 s. Subtrees consisting of a mixture of λ and 0.95 are not allowed in the optimal structure. Since POLE identifies these building blocks, mixed subtrees are not generated. In the DMAX problem there is another type of interaction: there are interactions among building blocks. As explained in Fig. 7, random combinations of 5λ and 5×0.95 do not produce good solutions. For the case $D_P = 3$, $(4.75)^5$ and $(5\lambda)^3(4.75)^2$ are local and global optima, respectively. In Fig. 9, we can see the edges generated between building blocks. POLE can identify these two interactions effectively, and is consequently able to obtain the optimum solutions using a lower number of fitness evaluations.

3) Addition of Extra Nodes: In order to investigate the redundancy of POLE, we added extra nodes to the DMAX

TABLE VI
THE NUMBER OF FITNESS EVALUATIONS FOR THE DMAX PROBLEM WITH EXTRA NODES. VALUES OF THE STANDARD DEVIATION (STDEV) FOR EACH CASE ARE GIVEN IN PARENTHESES

		$D_P = 3$	$D_P = 4$
No addition	Average	1587	124531
(Table IV POLE)	Stdev	(139)	(23403)
Addition of	Average	1806	130916
extra functions	Stdev	(162)	(19527)
Addition of	Average	2033	–
extra terminals	Stdev	(230)	–

problem. We experimented with two distinct settings defined in (15) and (16). In the setting of (15), extra function nodes add_m^* and $\mathrm{multiply}_m^*$ are used. These two functions perform the same functions as add_m and $\mathrm{multiply}_m$, respectively, but are considered as different symbols in the probability distribution. In the second setting of (16), two extra terminals are used in the same fashion

$$F = \{\mathrm{add}_m, \mathrm{add}_m^*, \mathrm{multiply}_m, \mathrm{multiply}_m^*\}$$
$$T = \{\lambda, 0.95\} \tag{15}$$
$$F = \{\mathrm{add}_m, \mathrm{multiply}_m, \}$$
$$T = \{\lambda, \lambda^*, 0.95, 0.95^*\}. \tag{16}$$

Table VI shows the results of this experiment. In this table, the "no addition" results are identical to those of POLE in Table IV. By adding the extra function nodes, the number of fitness evaluations increases a little. In contrast, adding the extra terminal nodes causes POLE to fail to obtain the optimum for $D_P = 4$ even with $M = 25000$. We observed the estimated network for this setup, and found that the network is not constructed at all. As shown in the previous sections, the DMAX problem has interactions between terminal nodes. In the case of adding the extra terminal nodes, the two symbols λ and λ^* are taken to be different symbols in the probability distribution, even though their functionalities are completely identical. Thus, the emergence of mixed building blocks as $(\lambda+\lambda+\lambda^*+\lambda^*+\lambda)$ makes it difficult for POLE to estimate the interactions. Thus, a more sophisticated network construction method is required to identify the interactions for this more difficult case.

4) Effect of Selection Pressure: In this experiment, we used the parameter value $P_s = 0.1$ in the POLE method. This means that the best $0.1 \times M$ individuals are used for the construction of networks and the estimation of parameters. In order to investigate the effect of P_s on search performance, we repeated the experiment varying P_s, while keeping all other parameters unchanged. Specifically, we considered values $P_s = 0.05, 0.1, 0.2, 0.5$, for $D_P = 4$ and $M = 6300$, and observed the probability of success at each generation to quantify the effect of P_s.

Fig. 12 shows the probability of success for each value of P_s. As can be seen, smaller values of P_s tend to obtain the optimum with fewer generations. However, some runs for $P_s = 0.05$ failed to find the optimum. This means that stronger selection pressure (smaller P_s) gives rise to faster convergence. On the other hand, every run at $P_s = 0.5$ failed to obtain the optimum. This is because individuals selected with weaker selection pressure contain significant noise, which makes it difficult for POLE to estimate the interactions.

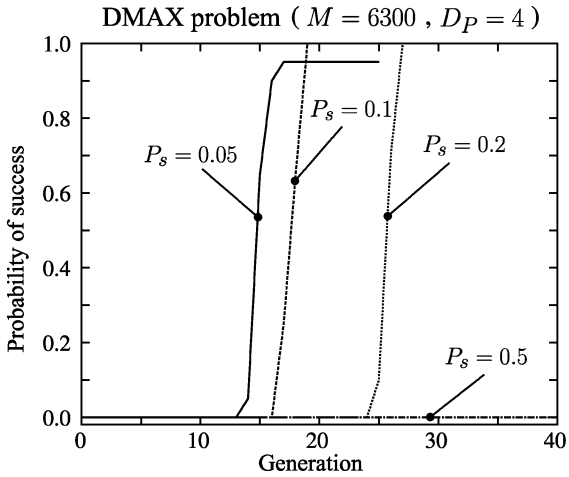

Fig. 12. The probability of success for $P_s = 0.05, 0.1, 0.2, 0.5$ ($D_P = 4$) for the DMAX problem. All parameters other than P_s are unchanged between experiments.

C. Experiment 3: Royal Tree Problem

1) Problem Description: To show the effectiveness of POLE, we applied our approach to the royal tree problem [24]. The royal tree problem is an extension of the royal road function [18], which is designed to investigate the functionality of schema in GA. Similarly, the royal tree problem can clarify the search mechanism in GP. In the royal tree problem, the optimum structure can be obtained by combining the building blocks using crossover operators.

The royal tree problem uses a set of functions defined by alphabetical symbols $F = \{A, B, C, D, \ldots\}$, and the terminal nodes $T = \{x, y, \ldots\}$. The optimal solution of the royal tree problem is given by the *perfect tree* state. Fig. (13a) and (b) are examples of a perfect tree. In perfect trees, alphabets of functions descend by one from a root to leaves in a tree. Furthermore, a function A has a terminal x.

A fitness value of an individual is given by the score of its root node ($\mathrm{fitness} = \mathrm{Score}(X_0)$). The value $\mathrm{Score}(X_i)$ denotes the score of X_i and is expressed by

$$\mathrm{Score}(X_i) = wb_i \sum_j (wa_{ij} \times \mathrm{Score}(X_{ij})) \tag{17}$$

where X_{ij} is a jth child (in a tree structure, counting from the left) of X_i, and wa_i and wb_{ij} are weights defined as follows:

- wa_{ij}
  - — $\mathrm{Full\ Bonus} = 2$
    If a subtree rooted at X_{ij} has a correct root and is a perfect tree.
  - — $\mathrm{Partial\ Bonus} = 1$
    If a subtree rooted at X_{ij} has a correct root but is not a perfect tree.
  - — $\mathrm{Penalty} = 1/3$

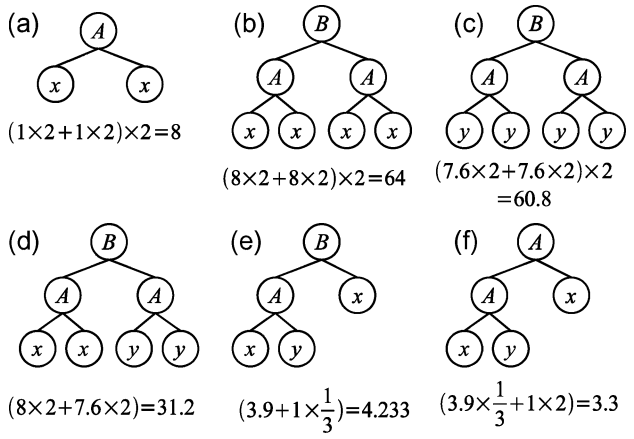

Fig. 13. Examples of the scoring system for $\mathrm{Score}(x) = 1$ and $\mathrm{Score}(y) = 0.95$.

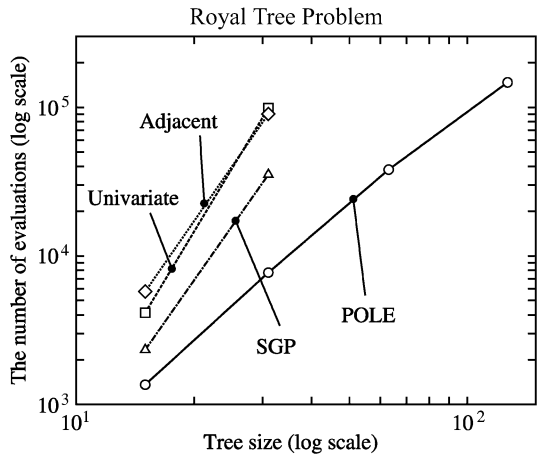

Fig. 14. The number of fitness evaluations in the royal tree problem. Results of the Univariate and the Adjacent models are partly omitted, because these models could not obtain the optimum solution when $D_P = 6$ (tree size $= 63$) or $D_P = 7$ (tree size$= 127$).

If X_{ij} is not a correct root.

- wb_i
  - — Complete Bonus $= 2$
    If a subtree rooted at X_i is a perfect tree.
  - — Otherwise $= 1$

The original royal tree problem only adds the score in the case of a terminal x. However, the authors of the royal tree problem noted that it is possible to add deceptiveness to this problem by offering an intermediate score to terminals y, z, etc. [24]. In this comparative experiment, we set $\mathrm{Score}(x) = 1, \mathrm{Score}(y) = 0.95$, and also define a perfect tree using y [Fig. 13(c)]. Examples of scores in the deceptive version of the royal tree are presented in Fig. 13(a)–(f).

The original royal tree problem uses a set of functions that have increasing arity ($A(a_0)$, $B(a_0, a_1)$, $C(a_0, a_1, a_2)$, etc.) to make the problem more difficult. Since the royal tree problem used in this experiment has deceptiveness, and so is already difficult, we omitted this feature and assumed all functions have an arity of 2.

In this problem, the Selection rate and Parent range are set to 0.2 and 4, respectively.

2) Results and Analysis: Fig. 14 shows the average number of fitness evaluations over 20 runs. In this figure, the number of fitness evaluations of SGP, the Univariate model and the Adjacent model at $D_P = 6$ and $D_P = 7$ are not shown because these algorithms could not achieve a $\mathrm{probability\ of\ success} = 100\%$ for these values of D_P, even with $M = 25000$. As can be seen from Fig. 14, POLE requires fewer fitness evaluations than the other methods. For $D_P = 6$ and $D_P = 7$, only POLE could obtain the optimum solution with a probability of 100%.

This problem has both parent–child interactions (e.g., children of E must be D) and lateral interactions (e.g., siblings of x must be x). Since the Univariate model does not consider interactions between nodes, it tends to become trapped around a local structure that is a mixture of x and y terminals [an example of such a structure is shown in Fig. 13(d)]. Since the Adjacent model only takes into account interactions between parents and children in the tree structure, it often failed to find the optimum solutions.

Table VIII shows the results of applying the t-test. From these values, we can say that the difference between the averages due to the POLE method and the Univariate, the Adjacent or SGP methods is statistically significant at a significance level of 1%.

Most runs of SGP succeeded in obtaining the optimum structure. However, SGP sometimes became trapped around a local optimum that is composed of the terminal y [i.e. structures like that shown in Fig. 13(c)]. Once SGP obtains such a structure, it is very difficult for this method to escape from such a local optimum. As was the case for the DMAX problem, this problem arises due to the fact that intermediate structures between the local and the global optima have lower fitness values than these two optima.

POLE constructs a Bayesian network that is very flexible. Since POLE succeeds in grasping the interactions in the royal tree problem, this method is able to obtain the optimum solution with significantly fewer fitness evaluations. We can thus conclude that the proposed method is a highly effective method of solving the royal tree problem with deceptiveness.

We examined the estimated interactions found by POLE for the royal tree problem. Fig. 15(a) and (b) depict the Bayesian network of generations 2 and 10, respectively, constructed during the evolution. This is a network of a randomly selected run that found the optimum solution at $\mathrm{generation} = 15$ ($M = 4000$. The depth is set to $D_P = 6$ due to limitations of space). From Fig. 15(a), we can see that POLE identifies the interactions among parents and children in the earlier generations. The optimal structure of the royal tree problem has a vertical ordering of function nodes: function D must have C, and C must have B, and so on. In earlier generations, the POLE method mainly constructs function nodes. In later generations [Fig. 15(b)], POLE instead estimates the interactions between the terminal nodes. After deciding the function nodes, there are two types of optima: the optimum that consists of x [Fig. 13(b)] and the optimum that consists of y [Fig. 13(c)]. The probability that a terminal node positioned next to terminal $x(y)$ is $x(y)$ and is very high. POLE grasps these interactions, and this estimation enables POLE to escape from local optima.

3) The Effect of Selection Pressure: In the royal tree problem, we used the parameter value $P_s = 0.2$ in applying the POLE method. In order to investigate the effect of this parameter P_s on the performance of the algorithm, we varied this parameter over the values $P_s = 0.05$, 0.1, 0.2, 0.5 keeping the values of other parameters unchanged. Specifically, we fixed $D_P = 7$

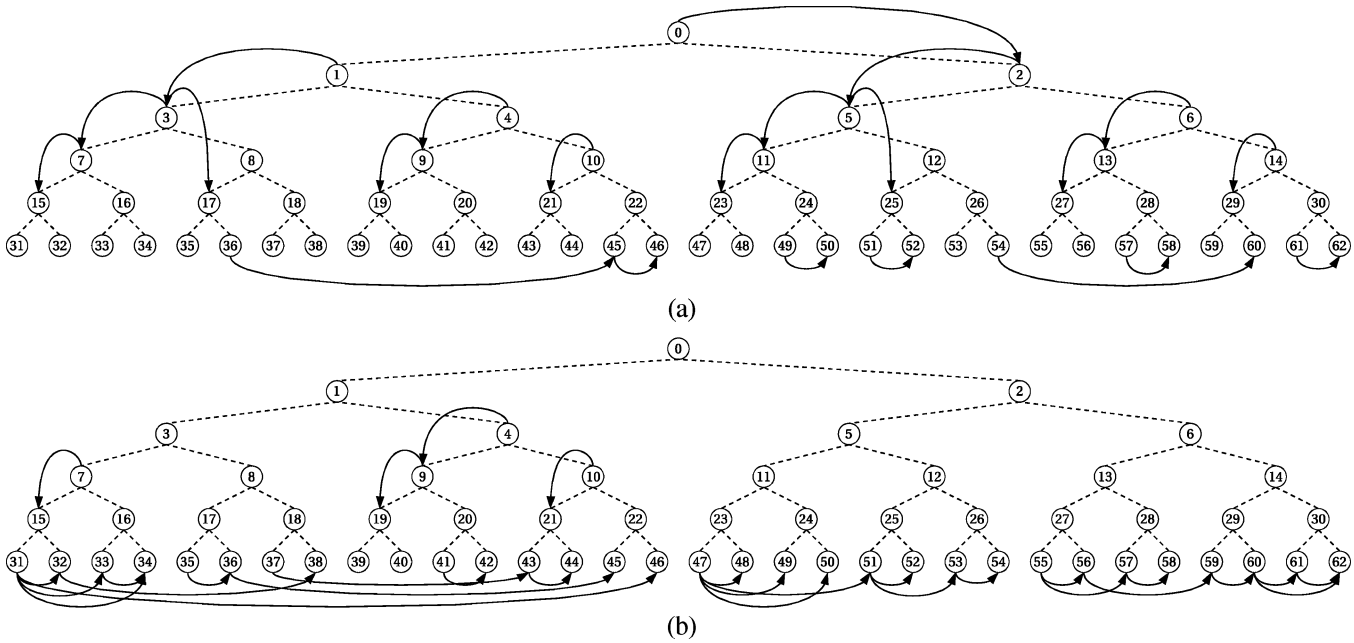

Fig. 15. An example of a network for the royal tree problem ($M = 4000$, $D_P = 6$). Networks are shown at (a) generation= 2 and (b) generation= 10 .

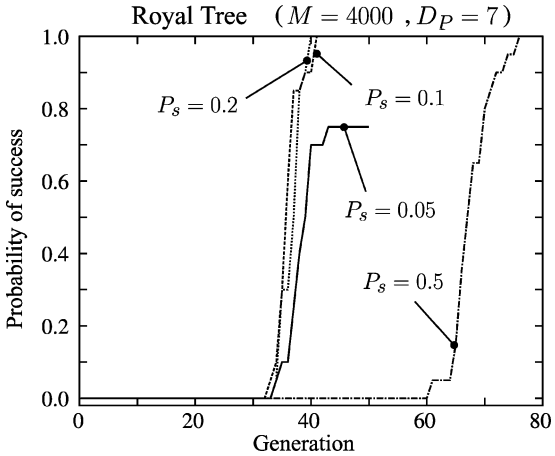

Fig. 16. The probability of success for $P_s = 0.05$, 0.1, 0.2, 0.5 for the royal tree problem ($D_P = 7$, $M = 4000$).

TABLE VII
THE NUMBER OF FITNESS EVALUATIONS FOR ROYAL TREE PROBLEMS. VALUES OF THE STANDARD DEVIATION (STDEV) FOR EACH CASE ARE GIVEN IN PARENTHESES

		$D_P = 4$	$D_P = 5$	$D_P = 6$	$D_P = 7$
POLE	Average	1357	7711	38039	147030
	Stdev	(339)	(1291)	(5847)	(13191)
Univariate	Average	4135	98837	–	–
	Stdev	(864)	(29131)	–	–
Adjacent	Average	5752	90080	–	–
	Stdev	(1791)	(20016)	–	–
SGP	Average	2331	35238	–	–
	Stdev	(989)	(16234)	–	–

and $M = 4000$. We observed the probability of success at each selection rate, undertaking 20 runs for each value of P_s.

Fig. 16 shows the probability of success for each value of P_s. For this problem, the parameter P_s has a smaller effect on search performance in comparison with the effect of this parameter on the search performance for the DMAX problem. As we previously explained, the royal tree problem has more complicated interactions than those of the DMAX problem. To obtain an optimum structure, POLE has to estimate the interactions between functions at earlier generations, and those between terminals at later generations in the royal tree problem. Thus, the royal tree problem requires more fitness evaluations than those of the DMAX problem and this complicated network estimation reduces the effect of the selection pressure.

*4) Expanded Parse Tree (*EPT*) Versus Standard Parse Tree (SPT):* In Section III-B, we have argued that the EPT is effective when applying probabilistic models to tree structures. In this section, we repeated the experiment using the SPT, while keeping network models unchanged in order to examine the effect of the EPT on search performance. We considered the network models of Bayesian network, Univariate and Adjacent network (these three methods are referred to as Bayesnet + SPT, Univariate + SPT, and Adjacent + SPT respectively, in Table IX). As pointed out in the previous section, when using the SPT, we have to take into account the syntactical correctness. In this experiment, we filled empty positions in the SPT [Fig. 1(a) dashed line circles] with introns.

Table IX describes the number of fitness evaluations required by the six methods, and the corresponding standard deviations. The methods labeled here by Bayesnet + EPT, Univariate + EPT, and Adjacent + EPT are identical to algorithms used through Section V (described as POLE, the Univariate model, and the Adjacent model, respectively, in the previous sections). Thus, the values of these three methods are taken from Table VII. Table X shows the results of applying the t-test.

TABLE VIII
t-TEST. THE VALUES REPRESENT P-VALUES FOR THE ROYAL TREE PROBLEM

	$D_P = 4$	$D_P = 5$
POLE vs Univariate	7.89×10^{-7}	3.83×10^{-6}
POLE vs Adjacent	2.20×10^{-5}	3.62×10^{-7}
POLE vs SGP	1.32×10^{-2}	4.46×10^{-4}

TABLE IX
THE NUMBER OF EVALUATIONS IN TWO TYPES OF CHROMOSOMES: EPT AND SPT DENOTE THE EXPANDED PARSE TREE AND THE STANDARD PARSE TREE, RESPECTIVELY

		$D_P = 4$	$D_P = 5$	$D_P = 6$	$D_P = 7$
Bayesnet + EPT	Average	1357	7711	38039	147030
(Table 7 POLE)	Stdev	(339)	(1291)	(5847)	(13191)
Bayesnet + SPT	Average	1429	11003	–	–
	Stdev	(362)	(2554)	–	–
Univariate + EPT	Average	4135	98837	–	–
(Table 7 Univariate)	Stdev	(864)	(29131)	–	–
Univariate + SPT	Average	4584	117432	–	–
	Stdev	(1338)	(38959)	–	–
Adjacent + EPT	Average	5752	90080	–	–
(Table 7 Adjacent)	Stdev	(1791)	(20016)	–	–
Adjacent + SPT	Average	7765	112494	–	–
	Stdev	(2220)	(34258)	–	–

TABLE X
t-TEST. THE VALUES REPRESENT P-VALUES BETWEEN THE EXPANDED PARSE TREE AND THE STANDARD PARSE TREE FOR EACH PROBABILISTIC MODEL

	$D_P = 4$	$D_P = 5$
Bayesnet EPT vs SPT	6.51×10^{-1}	2.90×10^{-3}
Univariate EPT vs SPT	3.86×10^{-1}	2.44×10^{-1}
Adjacent EPT vs SPT	3.92×10^{-2}	8.28×10^{-2}

For the case of the Bayesian network, the EPT is superior to the SPT for $D_P = 5, 6, 7$. For $D_P = 5$, we can see that the EPT required fewer evaluations (with this difference being statistically significant). For $D_P = 6, 7$, although a method using the EPT (POLE) succeeded in obtaining the optimum, the SPT-based method failed to obtain the optimum. With increasing depth tree structures contain more noise. Thus, in deeper trees using the SPT, it is difficult to construct a Bayesian network. For the case of the Univariate approach, there is not a statistically significant difference between the results for the EPT and the SPT. Since the Univariate model does not have to construct a network, it is considered that the effects of large CPT size and noise on search performance are smaller. For the case of the Adjacent network approach, the EPT showed better results.

From these results, we can say that the EPT is particularly effective when constructing networks. This is because in the EPT all function nodes are positioned at the branches and all terminals at leaves in the tree. As pointed out in Section III-B, this reduces the size of the CPT. At the same time, an EPT constructed using only L greatly reduces noise. This noise reduction is also effective for constructing networks.

VI. DISCUSSION

We have employed a Bayesian network to estimate interactions between nodes, and have compared the search performance of the proposed method with other algorithms. Since the Univariate and Adjacent models do not use promising solutions to estimate the interaction between nodes, these algorithms can easily destroy building blocks. These algorithms are thus not suitable for problems where there are interactions between nodes. We have tested the proposed method by applying it to the MAX problem (no interactions), the DMAX problem (lateral interactions) and the royal tree problem (parent–child and lateral interactions). Because the Univariate model does not estimate interactions, it could not always solve the DMAX or the royal tree problem. The Adjacent model considers parent–child interactions, and it could solve the royal tree problem slightly better than the Univariate model. However, the Adjacent model can only estimate parent–child interactions, and so it cannot search as effectively as the present POLE method. Since our approach learns structures using given samples, it required fewer fitness evaluations than the other methods tested, and succeeded in obtaining the solution to all three test problems considered.

We should also discuss the performance of the four methods used in the present study with regard to their computational time requirements. Since POLE constructs Bayesian networks from promising individuals, the evolution steps require more computational time than that required by the Univariate or Adjacent models, which do not require the construction of networks from the given samples. However, it is the step evaluating the population that accounts for the bulk of the computational time. Furthermore, POLE does not estimate interactions between distant nodes, which limits the computational time devoted to network construction in the POLE method. The computational time of POLE is not prohibitively long in comparison with algorithms that do not undertake a network construction.

In this paper, we have carried out ten sets of experiments in order to conduct the t-tests. The t-test uses the property of the asymptotic normality of average value. Thus, when the number of repetitions of an experiment is relatively small, such as the ten repetitions of the present study, it may perhaps be considered that the power of the t-test is relatively low. However, for the case of the royal tree problem ($D_P = 6, 7$), only POLE could solve the problem with a probability of 1. For the DMAX problem with $D_P = 4$, the Univariate and the Adjacent models failed to obtain the optimum, while SGP required ten times more fitness evaluations compared with POLE. It may be said that POLE is superior to these other methods for these problems, even on the basis of a relatively small number of experiments.

We have shown the effectiveness of the EPT in Bayesian network-based program evolution. By using the EPT, more sophisticated graphical models applied in EDA, such as a mixture of Bayesian [23] and Markov networks [29], can be introduced to POLE with almost no modification. Algorithms employing these graphical models are reported to be more powerful than those based on a simple Bayesian network.

Furthermore, a hybrid approach using crossover and mutation can also be considered. Previous research [36] has reported that the combination of the EDP and GP approaches is highly effective in boosting the search performance.

In this paper, we have demonstrated the effectiveness of POLE mainly from the viewpoint of search performance. However, there may be some applications where estimated networks or estimated probabilities themselves are important. In such applications, it can be considered that POLE may not be only used as an optimization algorithm, but also as a data mining tool.

VII. Conclusion

We have proposed a new PMBGP technique named POLE that is based on the EPT and that estimates interactions between nodes by employing a Bayesian network. By applying the present method along with three other methods to a set of benchmark problems, it has been shown that the performance of POLE is superior to these other methods. We hope to report the extension and application of POLE in the future.

References

[1] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-94-163, 1994.

[2] P. A. N. Bosman and E. D. de Jong, "Grammar transformations in an EDA for genetic programming," Inst. Information and Computing Sciences, Utrecht Univ., Utrecht, The Netherlands, Tech. Rep. UU-CS-2004-047, 2004.

[3] G. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, vol. 9, pp. 309–347, 1992.

[4] R. Etxeberria and P. Larrañaga, "Global optimization using Bayesian networks," in *Proc. 2nd Symp. Artif. Intell. (CIMAF-99)*, 1999, pp. 332–339.

[5] N. Friedman and Z. Yakhini, "On the sample complexity of learning Bayesian networks," in *Proc. 12th Conf. Uncertainty in Artif. Intell.*, 1996, pp. 274–282.

[6] C. Gathercole and P. Ross, "An adverse interaction between crossover and restricted tree depth in genetic programming," in *Proc. 1st Annu. Conf. Genetic Programming*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., 1996, pp. 291–296.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Mach. Learn.*. Reading, MA: Addison-Wesley, 1989.

[8] G. Harik, "Linkage learning via probabilistic modeling in the ECGA," IlliGAL Rep. (99010), 1999.

[9] Y. Hasegawa and H. Iba, "Estimation of Bayesian network for program generation," in *Proc. 3rd Asian-Pacific Workshop on Genetic Programming*, Hanoi, Vietnam, 2006a, pp. 35–46.

[10] Y. Hasegawa and H. Iba, "Optimizing programs with estimation of Bayesian network," in *Proc. 2006 World Congr. Comput. Intell.*, Vancouver, BC, Canada, 2006b, pp. 5527–5534.

[11] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[12] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[13] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA: MIT Press, 1994.

[14] W. B. Langdon and R. Poli, "An analysis of the MAX problem in genetic programming," in *Proc. 2nd Annu. Conf. Genetic Programming*, J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, Eds., 1997, pp. 222–230.

[15] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms*. Norwell, MA: Kluwer, 2002.

[16] M. Looks, "Learning Computer programs with the Bayesian optimization algorithm," Master thesis, Washington Univ., Sever Inst. Technol., St. Louis, MO, 2005.

[17] M. Looks, B. Goertzel, and C. Pennachin, "Learning computer programs with the Bayesian optimization algorithm," in *Proc. 2005 Conf. Genetic Evol. Comput., GECCO 2005*, H. G. Beyer *et al.*, Ed., Washington, DC, 2005, vol. 2, pp. 747–748.

[18] M. Mitchell, S. Forrest, and J. H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," in *Proc. 1st Euro. Conf. Artif. Life, 1991 Towards a Practice of Autonomous Syst.*, F. J. Varela and P. Bourgine, Eds., 1992, pp. 245–254.

[19] H. Mühlenbein and T. Mahnig, "FDA—A scalable evolutionary algorithm for the optimization of additively decomposed functions," *Evol. Comput.*, vol. 7, no. 4, pp. 353–376, 1999.

[20] R. E. Neapolitan, *Learning Bayesian Networks*. Upper Saddle River, NJ: Pearson Education, 2004.

[21] M. Pelikan, "Bayesian optimization algorithm: From single level to hierarchy," Ph.D. dissertation, Univ. Illinois at Urbana–Champaign, Urbana, IL, 2002, also IlliGAL Rep. No. 2002023.

[22] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. Genetic Evol. Comput. Conf. GECCO 1999*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., Orlando, FL, 1999, vol. I, pp. 525–532.

[23] J. M. Peña, J. A. Lozano, and P. Larrañaga, "Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks," *Evol. Comput.*, vol. 13, no. 1, pp. 43–66, 2005.

[24] W. F. Punch, "How effective are multiple populations in genetic programming," in *Proc. 3rd Annu. Conf. Genetic Programming 1998:*, J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, Eds., 1998, pp. 308–313.

[25] C. P. Quesenberry and D. C. Hurst, "Large sample simultaneous confidence intervals for multinomial proportions," *Technometrics*, vol. 6, pp. 191–195, 1964.

[26] A. Ratle and M. Sebag, "Avoiding the bloat with probabilistic grammar-guided genetic programming," in *Proc. 5th Int. Conf. Artif. Evol. 2001*, P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, Eds., Creusot, France, 2001, vol. 2310, Lecture Notes in Computer Science, pp. 255–266.

[27] E. N. Regolin and A. T. R. Pozo, "Bayesian automatic programming," in *Proc. 8th Eur. Conf. Genetic Programming*, M. Keijzer, A. Tettamanzi, P. Collet, J. I. van Hemert, and M. Tomassini, Eds., Lausanne, Switzerland, 2005, vol. 3447, Lecture Notes in Computer Science, pp. 38–49.

[28] R. P. Sałustowicz and J. Schmidhuber, "Probabilistic incremental program evolution: Stochastic search through program space," in *Machine Learning: ECML-97*, M. van Someren and G. Widmer, Eds. Berlin, Germany: Springer-Verlag, 1997, vol. 1224, pp. 213–220.

[29] R. Santana, "Estimation of distribution algorithms with Kikuchi approximations," *Evol. Comput.*, vol. 13, no. 1, pp. 67–97, 2005.

[30] K. Sastry and D. E. Goldberg, "Probabilistic model building and competent genetic programming," in *Genetic Programming Theory and Practice*, R. L. Riolo and B. Worzel, Eds. Norwell, MA: Kluwer, 2003, ch. 13, pp. 205–220.

[31] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[32] Y. Shan, R. I. McKay, H. A. Abbass, and D. Essam, "Program evolution with explicit learning: A new framework for program automatic synthesis," in *Proc. 2003 Congr. Evol. Comput. CEC2003*, R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, Eds., Canberra, 2003, pp. 1639–1646.

[33] Y. Shan, R. I. McKay, R. Baxter, H. Abbass, D. Essam, and N. X. Hoai, "Grammar model-based program evolution," in *Proc. 2004 IEEE Congr. Evol. Comput.*, G. Greenwood, Ed., Portland, OR, 2004, pp. 478–485.

[34] M. Wineberg and F. Oppacher, "A representation scheme to perform program induction in a canonical genetic algorithm," in *Lecture Notes in Computer Science*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Berlin, Germany: Springer-Verlag, 1994, vol. 866, Parallel Problem Solving From Nature III, pp. 292–301.

[35] K. Yanai and H. Iba, "Estimation of distribution programming based on Bayesian network," in *Proc. 2003 Congr. Evol. Comput. CEC2003*, R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon, Eds., Canberra, pp. 1618–1625.

[36] K. Yanai and H. Iba, "Program evolution by integrating EDP and GP," in *GECCO 2004: Proc. 2004 Conf. Genetic Evol. Comput.*, K. Deb *et al.*, Ed., Seattle, WA, 2004, pp. 774–785.

Yoshihiko Hasegawa received the B.Sc. degree in physics and the M.S. degree in engineering from the University of Tokyo, Tokyo, Japan, in 2003 and 2005, respectively. Currently, he is working towards the Ph.D. degree at the Department of Frontier Informatics, Graduate School of Frontier Sciences, University of Tokyo.

His interests include estimation of distribution algorithm.

Hitoshi Iba graduated from the Department of Science, University of Tokyo, Tokyo, Japan, in 1985 and received the Ph.D. degree from the Department of Engineering, University of Tokyo, in 1990.

Since then, he had been working at the ElectroTechnical Lab (ETL). He joined the Department of Electronical Engineering, University of Tokyo, in April, 1998. He was an Associate Professor from 1998 to 2004. He is currently a Professor with the Graduate School of Frontier Sciences, University of Tokyo. His research interest includes evolutionary computation, genetic programming, bio-informatics, foundation of artificial intelligence, machine learning, robotics, and vision.

Dr. Iba is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the *Journal of Genetic Programming and Evolvable Machines* (GPEM).

.