Latent Variable Model for Estimation of Distribution Algorithm Based on a Probabilistic Context-Free Grammar

Yoshihiko Hasegawa and Hitoshi Iba

Abstract—**Estimation of distribution algorithms are evolutionary algorithms using probabilistic techniques instead of traditional genetic operators. Recently, the application of probabilistic techniques to program and function evolution has received increasing attention, and this approach promises to provide a strong alternative to the traditional genetic programming techniques. Although a probabilistic context-free grammar (PCFG) is a widely used model for probabilistic program evolution, a conventional PCFG is not suitable for estimating interactions among nodes because of the context freedom assumption. In this paper, we have proposed a new evolutionary algorithm named programming with annotated grammar estimation based on a PCFG with latent annotations, which allows this context freedom assumption to be weakened. By applying the proposed algorithm to several computational problems, it is demonstrated that our approach is markedly more effective at estimating building blocks than prior approaches.**

Index Terms—**EM algorithm, estimation of distribution algorithm, genetic programming, probabilistic context-free grammar, variational Bayes.**

Nomenclature

ACO	Ant colony optimization.
BAP	Bayesian automatic programming.
BOA	Bayesian optimization algorithm.
BOAP	BOA programming.
CFG	Context-free grammar.
CNF	Chomsky normal form.
DMAX	Deceptive MAX.
EA	Evolutionary algorithm.
ECGA	Extended compact genetic algorithm.
ECGP	Extended compact genetic programming.
EDA	Estimation of distribution algorithm.
EDP	Estimation of distribution programming.
EM	Expectation-maximization.
GA	Genetic algorithm.
GA-EDA	Genetic algorithm-based EDA.
GGGP	Grammar guided genetic programming.
GMPE	Grammar model-based program evolution.
GNF	Greibach normal form.
GP	Genetic programming.
GP-EDA	Genetic programming-based EDA.
GT-EDA	Grammar transformation in an EDA.
hBOA	Hierarchical-Bayesian optimization algorithm.
MAP	Maximum *a posteriori*.
MDL	Minimum description length.
MOSES	Meta-optimizing semantic evolutionary search.
NLP	Natural language processing.
PAGE	Programming with annotated grammar estimation.
PBIL	Population-based incremental learning.
PCFG	Probabilistic context-free grammar.
PCFG-LA	PCFG with latent annotations.
PEEL	Program evolution with explicit learning.
PIPE	Probabilistic incremental program evolution.
PLS	Probabilistic logic sampling.
POLE	Program optimization with linkage estimation.
SG-GP	Stochastic grammar-based genetic programming.
VB	Variational Bayes.

Manuscript received March 3, 2008; revised September 11, 2008 and December 29, 2008; accepted December 29, 2008. First version published July 28, 2009; current version published August 14, 2009.

Y. Hasegawa is with the Department of Computational Biology, Graduate School of Frontier Sciences, University of Tokyo, Tokyo 277-8561, Japan (e-mail: hasegawa@cb.k.u-tokyo.ac.jp).

H. Iba is with the Graduate School of Engineering, University of Tokyo, Tokyo 113-8656, Japan (e-mail: iba@iba.t.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TEVC.2009.2015574

I. Introduction

Evolutionary algorithms (EAs) are optimization algorithms based on the concept of natural evolution. Because EAs do not require strong modeling assumptions, they have been applied to a variety of problems and are considered to be a powerful solution technique. The main operators in EAs are crossover and mutation. Optimal solutions can be obtained by randomly applying these operators to promising solutions, which is an approach based on the concept of natural evolution. Recently, EAs have been considered from the viewpoint of distribution estimation, with estimation of distribution algorithms (EDAs) attracting increasing attention [16].

EAs can be considered as algorithms that sample from a distribution of promising solutions. Because the distribution is unknown, an approximation is required to perform the sampling. Genetic algorithms (GAs) [12] and genetic programmings (GPs) [13] sample from the distribution by randomly changing the promising solutions using genetic operators. On the other hand, EDAs assume a set of parametric models and sample from the predictive posterior distribution according to

$$P(Y|\mathcal{D}) = \sum_{m\in\mathcal{M}} \int d\theta \, P(Y|\theta, m) P(\theta|\mathcal{D}, m) P(m|\mathcal{D}) \quad (1)$$

where $\mathcal{M}$ is a set of assumed models used in the EDA, θ represents parameters, $\mathcal{D}$ represents learning data (the set of promising solutions), and Y is an individual. GA- and GP-type sampling are valid only for the case where two structurally similar individuals have similar fitness values. For problems not satisfying this condition, such as the deceptive problems [8], GA and GP do not perform very well. It has been reported that EDAs are highly effective for some problems that GA and GP do not handle well. Usually, EDAs use fixed-length linear arrays for chromosomes (in this paper, these EDAs are referred to as GA-EDAs). In recent years, tree-structure-based EDAs have also been proposed (GP-EDAs). Similar to GPs, GP-EDAs can handle programs and functions.

The GP-EDA algorithms can be broadly classified into two groups: prototype-tree based methods and probabilistic context-free grammar (PCFG) based methods. For algorithms in the first group, variable length tree-structures are translated into fixed length structures (mostly linear arrays). Algorithms in the second group utilize a context-free grammar (CFG) for expressing programs and functions. Algorithms based on PCFG are considered to be well suited for expressing functions in GP, and as a consequence many algorithms based on this latter approach have been proposed.

The conventional PCFG adopts the context freedom assumption that the probabilities of production rules do not depend on the parent nodes or sibling nodes. The basic GP-EDA based on the conventional PCFG estimates parameters (the production rule probabilities) and generates programs using these parameters. Although the independence assumption makes the statistical estimation easier, the conventional PCFG is essentially unable to take into account the interactions among nodes. Since functions and programs in GP often exhibit strong dependencies between nodes, the conventional PCFG cannot in principle estimate the building blocks of GP. In order to weaken the independence assumption for PCFG, annotations have been proposed in natural language processing (NLP). For example, vertical Markovization annotates the symbols with their ancestor symbols. Prior GP-EDAs, such as vectorial stochastic grammar-based GP or grammar transformation in an EDA (see Section II), used annotated PCFGs as base line grammars. In these algorithms, the depth of the production rules is taken into account.

Matsuzaki *et al.* [20] proposed the PCFG with latent annotations (PCFG-LA), which assumes that the annotations are latent and that the production rules with annotations can be estimated by the expectation-maximization (EM) algorithm. It may be expected that PCFG-LA may more precisely grasp the interactions in GP than previous methods and that it may be more flexible compared with methods based on heuristics-based annotations. As mentioned above, it is important to carefully select the models used in EDAs [i.e., $\mathcal{M}$ in (1)]. For the case of GA-EDA, EDAs using a Bayesian network are richer than EDAs using a univariate model and are considered to be more powerful algorithms. Similarly, because PCFG-LA is a richer model than PCFG using heuristics, we consider GP-EDA using PCFG-LA to be potentially more powerful than GP-EDA using PCFG with heuristics-based annotations.

The contributions of this paper may be summarized as follows.

1) We have proposed a GP-EDA named programing with annotated grammar estimation (PAGE), based on PCFG-LA, which allows the context freedom assumption to be weakened.
2) The effectiveness of PAGE is shown by several computational experiments.
3) We have derived the variational Bayes (VB) learning for PCFG-LA. By using VB learning, the number of annotations can be inferred from learning data.

This paper is structured as follows: In the next section, we briefly introduce program evolution methods based on probabilistic techniques. In Section III, we explain the PCFG-LA that is used in our proposed method to update parameters. We describe a model learning algorithm for PCFG-LA in Section IV. In this section, both the EM algorithm and the VB-based learning algorithm are considered. We describe the algorithm for sampling the predictive posterior distribution in Section V. PAGE-VB estimates the optimum annotation size during the search. A method to select the optimum annotation size is discussed in Section VI. Section VII shows flowcharts of PAGE. In Section VIII, we compare the performance of our method to other existing methods, selecting four benchmark tests for experiments. We discuss the results obtained in these experiments in Section IX. Finally, we conclude this paper in Section X.

II. Related Work

EAs based on probabilistic models were first applied using GA-style linear chromosomes. Since GP uses tree structures that are more complex than the linear arrays used in GA, progress in developing GP-EDAs has followed that of GA-EDAs. Many GP-EDAs have hitherto been proposed, and these can be broadly classified into two groups: 1) prototype-tree-based methods and 2) grammar model-based methods. Those not familiar with GP-EDAs may wish to see [33, ch. 6] which contains a concise survey. In this book, together with the basic concepts of GP-EDA, prototype-tree-based methods (PIPE to EDP, see following paragraphs), and PCFG-based methods (SG-GP to GT-EDA) are briefly introduced. The characteristics of GP building blocks are also addressed in [33].

Methods of type 1) take advantage of the techniques devised in GA-EDA. This type of algorithm translates tree structures into the fixed-length chromosomes used in GA and applies probabilistic models. Probabilistic incremental program evolution (PIPE) [29] adopts the univariate model, which can be considered to be a combination of population-based incremental learning (PBIL) [3] and GP. Estimation of distribution programming (EDP) [40], [41] considers the parent–children relationships in the tree structure. EDP considers the conditional probability distribution of children, given parent nodes. Extended compact GP (ECGP) [30] combines the extended compact GA (ECGA) [9] with GP to take into account the relationships among nodes. This algorithm estimates the group of marginal distributions using the minimum description length (MDL) principle. Bayesian optimization algorithm (BOA) programming (BOAP) [17] extends BOA [24] so that it can handle programs and functions. BOAP uses a zig-zag tree, which is based on the lambda function. Program optimization with linkage estimation (POLE) [10], [11] estimates the interactions among nodes by estimating the Bayesian network. In this algorithm, a special chromosome called an *expanded parse tree* [39] is used to convert GP programs into linear arrays. Meta-optimizing semantic evolutionary search (MOSES) [18] is an extension of a well-known GA-EDA called the hierarchical Bayesian optimization algorithm (hBOA) [23] to program evolution. MOSES evolves demes of programs, and hBOA is applied to each demes to grasp dependencies across variables. Additionally, MOSES simplifies the redundant structures into simpler structures by means of heuristics-based rules.

Methods of type 2), which are based on PCFG, have a strong connection with grammar-guided genetic programming (GGGP) [37]. In GGGP, individuals are expressed using derivation trees. Whigham also indicated the connection between PCFG and GP [38]. Although this paper was not written in terms of GP-EDAs, probability table learning in this algorithm can be considered as an EDA with local search. Stochastic grammar-based GP (SG-GP) [27] uses simple PCFG and integrates PBIL with GGGP. In this algorithm, the parameters are updated using an incremental learning strategy. These authors also proposed vectorial SG-GP, which considers depth in its grammar (simple SG-GP is then called scalar SG-GP). Program evolution with explicit learning (PEEL) [31] is an algorithm which is an extension of SG-GP that takes into account the positions (arguments) and depths of symbols. Grammar model-based program evolution (GMPE) [32] merges the production rules to establish general production rules. This method starts from specialized production rules and merges non-terminals to yield more general production rules using the MDL principle. Tanev proposed GP based on a probabilistic context-sensitive grammar [35], [36]. He employed sibling nodes and a parent node to contain context information, and probabilities are represented by conditional probabilities, given these contexts. His approach incorporated context information in a PCFG-based GP. Grammar transformation in an EDA (GT-EDA) [5] takes advantage of the MDL principle to estimate expanded production rules (represented by a tree structure) and extract good subroutines. Bayesian automatic programming (BAP) [28] uses a Bayesian network to consider relations among production rules.

Recently, a new type of GP-EDA has been proposed, called an N-gram GP [25]. An N-gram is a method proposed in NLP. An N-gram considers N consecutive sub-sequences of a given sequence, with reference to the conditional probability of a given sequence. The N-gram GP makes use of these concepts and applies the N-gram to the linear GP [22], which is the assembly language of a register based-CPU.

AntTAG [1] employed the ant colony optimization (ACO) method to evolve programs. Since the pheromone matrix in ACO can be interpreted as a probability distribution, AntTAG is capable of estimating dependencies in tree structures.

Unlike vectorial SG-GP, GT-EDA, and Tanev's work, which use fixed annotations, our proposed algorithm uses annotations that are estimated from promising solutions using the EM algorithm or the VB technique. Our method is more flexible than the fixed annotation model, and we expect our method to be more effective than methods based on fixed annotations. In the next section, we describe the details of PCFG-LA.

III. PCFG with Latent Annotations

A conventional PCFG takes advantage of the context freedom assumption that all the production rules are independent of their contexts. Because, given the context freedom assumption, a PCFG cannot take into account dependencies among nodes, recent GP-EDAs overcome this problem by using more advanced PCFGs. GMPE and GT-EDA use more specific grammars which are estimated from learning data. GMPE learns production rules from the most specialized rules and GT-EDA inversely from the most general rules. Another approach which can incorporate dependencies in a PCFG is to use annotations. In NLP, many approaches have been proposed to weaken the context freedom assumption by annotating non-terminal symbols with many features. Tanev's method can be considered as a PCFG with annotations. His method makes use of information of parent and sibling nodes. PEEL and GT-EDA also use annotations, and these methods make use of depth information. However, when using a PCFG with these fixed annotations, probabilities depend on parents or depth. For the case of the parent annotation, the same distribution is applied to nodes which have the same parent nodes. Matsuzaki *et al.* [20] have proposed a PCFG with latent annotations (PCFG-LA) which automatically learns the annotations using learning samples. They also derived a parameter-update formula employing the EM algorithm [6]. Because this model is richer in comparison with one utilizing fixed annotations based on depth and parent information and requires fewer assumptions than fixed annotated models, PCFG-LA would thus appear to be well suited for application to GP-EDA. In this paper, we propose a new PCFG-based GP-EDA named PAGE, which employs PCFG-LA as a base grammar.

The PCFG-LA used in PAGE has been developed specifically for the present application, but it is essentially identical to the conventional PCFG-LA. In this section, we describe the specialized version of PCFG-LA. For further details on the PCFG-LA technique, see [20].

In PCFG-LA, every non-terminal is labeled with annotations. In the complete form, non-terminals are represented by $A[x]$, where A is the non-terminal symbol, $x(\in H)$ is an annotation (which is not observed), and H is a set of annotations (in this paper, we take H to be the set of natural numbers, $H = \{0, 1, 2, 3, \ldots, h-1\}$, and h is the annotation size). Fig. 1 shows an example of a tree with (a) annotations and (b) the corresponding observed tree. A likelihood of an annotated tree can be represented as

$$\begin{aligned} &P(T_i, X_i|\boldsymbol{\beta}, \boldsymbol{\pi}, h) \\ &\quad = \prod_{x\in H} \pi(\mathcal{S}[x])^{\delta(x;T_i,X_i)} \prod_{r\in\mathcal{R}[H]} \beta(r)^{c(r;T_i,X_i)}. \end{aligned} \tag{2}$$

In this equation, T_i denotes the ith derivation tree, X_i is the set of latent annotations of T_i represented by $X_i \equiv \{x_i^1, x_i^2, \ldots\}$ (x_i^j is the jth annotation of T_i); $\pi(\mathcal{S}[x])$ is the

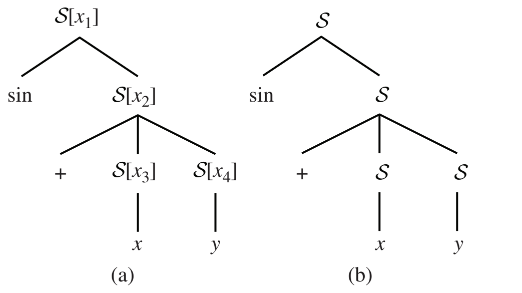

Fig. 1. Complete tree with (a) annotations and (b) its observed tree.

probability of $\mathcal{S}[x]$ at the root position, $\beta(r)$ is the probability of the annotated production rule $r \in \mathcal{R}[H]$; $\delta(x; T_i, X_i)$ is 1 if the annotation at the root node is x in the complete tree T_i, X_i and is 0 otherwise; $c(r; T_i, X_i)$ is the number of occurrences of rule $r \in \mathcal{R}[H]$ in the complete tree T_i, X_i; h is the annotation size that is specified in advance as a parameter in PAGE-EM and is estimated during the search in PAGE-VB; $\boldsymbol{\beta} \equiv \{\beta(\mathcal{S}[x] \to \alpha)|\boldsymbol{\beta} \equiv \{\beta(r)|r \in \mathcal{R}[H]\}\mathcal{S}[x] \to \alpha \in \mathcal{R}[H]\}$; and $\boldsymbol{\pi} \equiv \{\pi(\mathcal{S}[x])|x \in H\}$. The set of annotated rules $\mathcal{R}[H]$ is given in (7).

A likelihood of an observed tree can be calculated by summing over annotations

$$P(T_i|\boldsymbol{\beta}, \boldsymbol{\pi}, h) = \sum_{X_i} P(T_i, X_i|\boldsymbol{\beta}, \boldsymbol{\pi}, h). \tag{3}$$

The model learning of PCFG-LA is to infer $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$.

In this paper, production rules are not of the Chomsky normal form (CNF), as is assumed in the original PCFG-LA paper, because of understandability of GP programs. Any functions that can be handled with traditional GP can be represented by

$$\mathcal{S} \to g\,\mathcal{S} \ldots \mathcal{S} \tag{4}$$

which is a subset of Greibach normal form (GNF). Here, $\mathcal{S} \in \mathcal{N}$, and $g \in \mathcal{T}$ ($\mathcal{N}$ and $\mathcal{T}$ are the sets of non-terminal and terminal symbols in CFG, respectively). A terminal symbol g in CFG is a function node $(+, -, \sin, \cos)$ or a terminal (x, y) in GP. In the program evolution, only $\mathcal{S}$ is used for non-terminal nodes ($\mathcal{N} = \{\mathcal{S}\}$). On the right-hand side of (4), the number of $\mathcal{S}$ symbols is identical to the arity of g. If $g = +$, then $\mathcal{S} \to +\mathcal{S}\,\mathcal{S}$, and if $g = x$, then $\mathcal{S} \to x$. $\mathcal{S} \to g\,\mathcal{S}\,\mathcal{S}$ corresponds to the expression $g(\mathcal{S}, \mathcal{S})$. This conversion implies that any S-expression can be expressed with production rules represented by (4). Annotated production rules can be expressed by (5)

$$\mathcal{S}[x] \to g\,\mathcal{S}[z_1] \ldots \mathcal{S}[z_{a_{\max}}] \tag{5}$$

where $x, z_m \in H$ and $a_{\max}$ is the arity of g in GP. However, we have made an additional assumption in our algorithm due to the problem of the number of parameters rapidly increasing with increasing arity of g. Specifically, if g has $a_{\max}$ arity, the number of parameters for the production rule $\mathcal{S} \to g\,\mathcal{S} \ldots \mathcal{S}$ with annotations is given by $h^{a_{\max}+1}$. Estimating a large number of parameters may cause an overfitting problem and requires much more computational power. In order to reduce

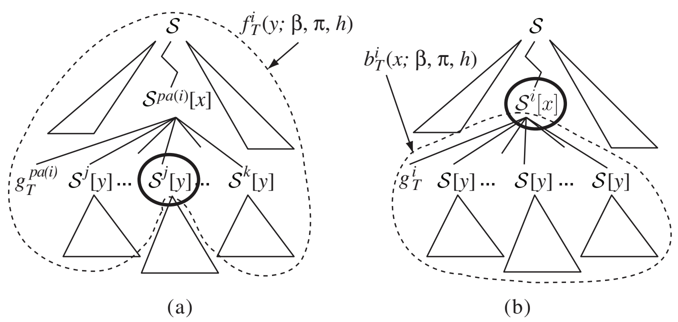

Fig. 2. (a) Forward and (b) backward probabilities. The superscripts denote the indices of non-terminals (i in $\mathcal{S}^i[y]$, for example).

the number of parameters, we assume that all the right-hand side non-terminal symbols have the same annotation, that is

$$\mathcal{S}[x] \to g\,\mathcal{S}[y]\,\mathcal{S}[y] \ldots \mathcal{S}[y]. \tag{6}$$

With this assumption, the number of parameters can be reduced to h^2, which is tractable. Henceforth, we let $\mathcal{R}[H]$ be the set of annotated rules expressed by

$$\mathcal{R}[H] \equiv \{\mathcal{S}[x] \to g\,\mathcal{S}[y]\,\mathcal{S}[y] \ldots \mathcal{S}[y]|x, y, \in H, g \in \mathcal{T}\}. \tag{7}$$

We also define a set $\mathcal{R}_r[H]$, which is composed of right-hand sides in $\mathcal{R}[H]$, i.e.,

$$\mathcal{R}_r[H] \equiv \{\alpha|\mathcal{S}[x] \to \alpha \in \mathcal{R}[H]\}.$$

A. Forward–Backward Probability

Before explaining the details of a model learning using PCFG-LA, we describe forward and backward probabilities. In [20], forward and backward probabilities are used to perform the EM algorithm. The backward probability $b_T^i(x; \boldsymbol{\beta}, \boldsymbol{\pi}, h)$ represents the probability that the tree beneath the ith non-terminal $\mathcal{S}[x]$ is generated [$\boldsymbol{\beta}, \boldsymbol{\pi}, h$ are parameters; see Fig. 2(b)]. The forward probability $f_T^i(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h)$ represents the probability that the tree above the ith non-terminal $\mathcal{S}[y]$ is generated [see Fig. 2(a)]. These probabilities can be recursively calculated [see (8)–(10)]; thus

$$b_T^i(x; \boldsymbol{\beta}, \boldsymbol{\pi}, h) = \sum_{y \in H} \beta(\mathcal{S}[x] \to g_T^i\,\mathcal{S}[y] \ldots \mathcal{S}[y]) \prod_{j \in \mathrm{ch}(i,T)} b_T^j(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \tag{8}$$

$$f_T^i(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h) = \sum_{x \in H} f_T^{\mathrm{pa}(i,T)}(x; \boldsymbol{\beta}, \boldsymbol{\pi}, h)\beta(\mathcal{S}[x] \to g_T^{\mathrm{pa}(i,T)}\,\mathcal{S}[y] \ldots \mathcal{S}[y]) \times \prod_{j \in \mathrm{ch}(\mathrm{pa}(i,T),T), j \neq i} b_T^j(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h), \quad (i \neq 1) \tag{9}$$

$$f_T^i(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h) = \pi(\mathcal{S}[y]), \quad (i = 1). \tag{10}$$

In these equations, $\mathrm{ch}(i, T)$ is a function that returns the set of non-terminal children indices of the ith non-terminal in T, $\mathrm{pa}(i, T)$ returns the parent index of the ith non-terminal in T, and g_T^i is a terminal symbol in CFG and is connected to the

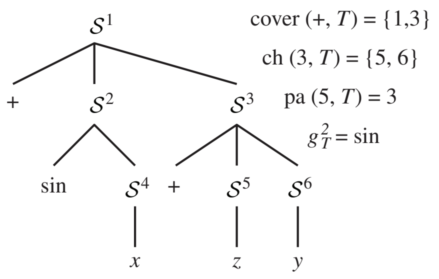

Fig. 3. Example of a derivation tree and values of the specific functions. The superscripts denote the indices of non-terminals.

ith non-terminal symbol in T. For example, for the tree shown in Fig. 3, $\mathrm{ch}(3, T) = \{5, 6\}$, $\mathrm{pa}(5, T) = 3$, and $g_T^2 = \sin$.

Using the forward–backward probability, $P(T; \boldsymbol{\beta}, \boldsymbol{\pi}, h)$ can be expressed by the following two equations:

$$P(T; \boldsymbol{\beta}, \boldsymbol{\pi}, h) = \sum_{x \in H} \pi(\mathcal{S}[x]) b_T^1(x; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \tag{11}$$

$$\begin{aligned} P(T; \boldsymbol{\beta}, \boldsymbol{\pi}, h) &= \sum_{x,y \in H} \Big\{ \beta(\mathcal{S}[x] \to g\, \mathcal{S}[y] \ldots \mathcal{S}[y]) f_T^i(x; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \\ &\qquad \times \prod_{j \in \mathrm{ch}(i,T)} b_T^j(y; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \Big\} \quad (i \in \mathrm{cover}(g, T)). \end{aligned} \tag{12}$$

In this equation, $\mathrm{cover}(g, T_i)$ represents a function that returns a set of non-terminal indices at which the production rule generating g without annotations is rooted in T_i. For example, if $g = +$ and T is the tree represented in Fig. 3, then $\mathrm{cover}(+, T) = \{1, 3\}$.

IV. Model Learning in PCFG-LA

Because PCFG-LA is a latent variable model, the model learning of PCFG-LA is not trivial. In the original paper proposing PCFG-LA, the EM algorithm was used for model learning. If the annotation size is available in advance, the EM algorithm performs well. However, when we apply the proposed method to unknown problems, the annotation size is unknown. It is thus desirable to estimate the optimum annotation size automatically from promising solutions. Because the EM algorithm is a point-estimation method, this algorithm cannot estimate the optimum annotation size. For the case of models that do not include latent variables, a model-selection method such as the Akaike information criteria (AIC) or the Bayesian information criteria (BIC) is often used. However, these methods take advantage of the asymptotic normality of estimators, which is not satisfied in models that include latent variables. Trial and error is therefore required to estimate the optimum annotation size for the EM algorithm, which is computationally expensive. In order to overcome this problem, we also use the VB learning [2] in PCFG-LA and apply PCFG-LA with VB learning to program evolution. VB learning is based on Bayes estimation. In this paper, we investigate two distinct approaches: one using the EM algorithm and another using VB learning. To distinguish these two approaches, we label the former and latter approaches by PAGE-EM (PAGE using EM algorithm) and PAGE-VB (PAGE using VB), respectively (see Table I). If we refer simply to "PAGE," then the discussion refers to both PAGE-EM and PAGE-VB.

TABLE I
Comparison of the Features of Two Proposed Methods: PAGE-EM and PAGE-VB

	Estimation method	Estimation target	Annotation size selection
PAGE-EM	EM algorithm	Parameter	Impossible
PAGE-VB	VB	Distribution	Possible

In the following sections, we describe PAGE-EM and PAGE-VB, respectively, in detail.

A. PAGE-EM: EM Algorithm-Based Estimation

In this section, we describe the EM algorithm used for model learning in PCFG-LA. The study [20] that originally proposed PCFG-LA also used the EM algorithm for model learning, and so many of the details of the present application of the EM algorithm are identical to those described in this previous study.

The EM algorithm maximizes the lower bound of the log likelihood difference and optimizes the parameters iteratively. Let $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ be current parameters and $\overline{\boldsymbol{\beta}}$ and $\overline{\boldsymbol{\pi}}$ be next-step parameters. The difference of the log-likelihood between $\log P(T; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, h)$ and $\log P(T; \boldsymbol{\beta}, \boldsymbol{\pi}, h)$ can be calculated as

$$\begin{aligned} &\log P(T; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, h) - \log P(T; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \\ &= \log \frac{P(T; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, h)}{P(T; \boldsymbol{\beta}, \boldsymbol{\pi}, h)} \\ &= \sum_X \Big\{ P(X|T; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \\ &\qquad \times \log \frac{P(T, X; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, h)}{P(T, X; \boldsymbol{\beta}, \boldsymbol{\pi}, h)} \frac{P(X|T; \boldsymbol{\beta}, \boldsymbol{\pi}, h)}{P(X|T; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, h)} \Big\} \\ &\geq \sum_X P(X|T; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \log \frac{P(T, X; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, h)}{P(T, X; \boldsymbol{\beta}, \boldsymbol{\pi}, h)}. \end{aligned} \tag{13}$$

Jensen's inequality is used to obtain the final inequality of (13). According to (13), the $\mathcal{Q}$ function to optimize in the EM algorithm can be expressed as follows:

$$\begin{aligned} &\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}} | \boldsymbol{\beta}, \boldsymbol{\pi}) \\ &\quad \equiv \sum_{i=1}^{N} \sum_{X_i} P(X_i | T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h) \log P(T_i, X_i; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, h) \end{aligned} \tag{14}$$

where N is the number of learning data (promising solutions in EDA). In this paper, a set of learning data is represented by $\mathcal{D} \equiv \{T_1, T_2, \ldots, T_N\}$. Using the forward–backward probability and optimizing $\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}} | \boldsymbol{\beta}, \boldsymbol{\pi})$, the update formula can be derived. The derivation of the update formula for PAGE-EM is essentially identical to that of the original PCFG-LA. However, because our approach is not based on chomsky normal form (CNF) and we limit the right-side annotations in production rules, the update formulae are different from those of the original paper (see Appendix II).

Finally, we obtain the following parameter update formulae:

$$\overline{\pi}(\mathcal{S}[x]) \propto \pi(\mathcal{S}[x]) \sum_{i=1}^{N} \frac{b_{T_i}^1(x;\boldsymbol{\beta},\boldsymbol{\pi},h)}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)} \tag{15}$$

$$\begin{aligned}\overline{\beta}(\mathcal{S}[x] \to g\,\mathcal{S}[y]\ldots\mathcal{S}[y]) &\propto \beta(\mathcal{S}[x] \to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])\\ &\times \sum_{i=1}^{N}\left[\frac{1}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)} \sum_{j\in \mathrm{cover}(g,T_i)} \left\{ f_{T_i}^j(x;\boldsymbol{\beta},\boldsymbol{\pi},h) \right.\right.\\ &\left.\left.\times \prod_{k\in \mathrm{ch}(j,T_i)} b_{T_i}^k(y;\boldsymbol{\beta},\boldsymbol{\pi},h)\right\}\right]. \end{aligned}\tag{16}$$

The EM algorithm maximizes the log-likelihood given by

$$\begin{aligned} L(\Theta;\mathcal{D}) &= \log P(\mathcal{D};\boldsymbol{\beta},\boldsymbol{\pi},h)\\ &= \sum_{i=1}^{N} \log P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h) \end{aligned}\tag{17}$$

monotonically from the initial parameters. The initial parameters were determined as follows:

$$\beta(\mathcal{S}[x] \to g\,\mathcal{S}[y]\ldots\mathcal{S}[y]) \propto \begin{cases} \dfrac{e^{-\kappa}}{h-1}\beta(\mathcal{S} \to g\,\mathcal{S}\ldots\mathcal{S}), & y \neq x\\ 0, & y = x \end{cases} \tag{18}$$

$$\beta(\mathcal{S}[x] \to g) \propto e^{-\kappa}\beta(\mathcal{S}[x] \to g). \tag{19}$$

In (18) and (19), κ is a random value which is uniformly distributed over $[-\log 3, \log 3]$, and $\beta(\mathcal{S} \to g\,\mathcal{S}\ldots\mathcal{S})$ is the probability of an observed production rule (without annotations). We set 0 for the case of $x = y$ in (18), because we only consider size trees of finite size (recursive production rules are not required).

B. PAGE-VB: VB-Based Estimation

The VB learning is an extension of the EM algorithm. Since the EM algorithm is a method for point estimation, this algorithm cannot estimate the optimum annotation size. In contrast, because the VB is based on Bayes estimation, the VB is able to estimate the optimum annotation size. We applied VB to PCFG-LA.

VB has been applied to the hidden Markov model (HMM) [19] and PCFG [14]. Since PCFG-LA, HMM, and PCFG share similar features, the derivation of VB for PCFG-LA can be established in a similar way to those of HMM and PCFG. In this section, we explain the application of VB to PCFG-LA. For details of VB, see [4], in which the basic concepts and many applications of VB, including HMM, are described.

We assume that a prior distribution over $\boldsymbol{\beta}$ is given by a product of Dirichlet distributions[1]

$$P(\boldsymbol{\beta}|h) = \prod_{x\in H} \mathrm{Dir}(\boldsymbol{\beta}_{\mathcal{S}[x]};\mathbf{b}_{\mathcal{S}[x]}) \tag{20}$$

[1]Dirichlet priors are used for computational reasons. Because the Dirichlet distribution is a conjugate prior in this case, the posterior distribution can be calculated in closed form [4].

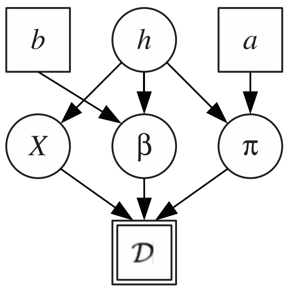

Fig. 4. Graphical model of PCFG-LA using VB learning.

$$\begin{aligned}\mathrm{Dir}(\boldsymbol{\beta}_{\mathcal{S}[x]};\mathbf{b}_{\mathcal{S}[x]}) &= \frac{\Gamma\left(\sum_{\alpha\in\mathcal{R}_r[H]} b_{\mathcal{S}[x]\to\alpha}\right)}{\prod_{\alpha\in\mathcal{R}_r[H]}\Gamma(b_{\mathcal{S}[x]\to\alpha})}\\ &\times \prod_{\alpha\in\mathcal{R}_r[H]} \beta(\mathcal{S}[x]\to\alpha)^{b_{\mathcal{S}[x]\to\alpha}-1}\end{aligned}\tag{21}$$

where $\Gamma(\cdots)$ is the gamma function, $\mathrm{Dir}(\cdots)$ is the Dirichlet distribution, $\boldsymbol{\beta}_{\mathcal{S}[x]} \equiv \{\beta(\mathcal{S}[x] \to \alpha)|\alpha \in \mathcal{R}_r[H]\}$, and $\boldsymbol{\beta} \equiv \{\beta_{\mathcal{S}[x]}|x \in H\}$. The quantity $\mathbf{b}_{\mathcal{S}[x]}$ denotes a set of hyper parameters of $\boldsymbol{\beta}_{\mathcal{S}[x]}$, which can be expressed as

$$\begin{aligned}\mathbf{b}_{\mathcal{S}[x]} &\equiv \{b_{\mathcal{S}[x]\to\alpha}|\alpha \in \mathcal{R}_r[H]\}\\ \mathbf{b} &\equiv \{\mathbf{b}_{\mathcal{S}[x]}|x \in H\}.\end{aligned}\tag{22}$$

Similarly, we assumed that a prior distribution over $\boldsymbol{\pi}$ is given by a Dirichlet distribution

$$\begin{aligned}P(\boldsymbol{\pi}|h) &= \mathrm{Dir}(\boldsymbol{\pi};\mathbf{a})\\ &= \frac{\Gamma\left(\sum_{x\in H} a_{\mathcal{S}[x]}\right)}{\prod_{x\in H}\Gamma(a_{\mathcal{S}[x]})} \prod_{x\in H} \pi(\mathcal{S}[x])^{a_{\mathcal{S}[x]}-1}\end{aligned}\tag{23}$$

where $\boldsymbol{\pi} \equiv \{\pi(\mathcal{S}[x])|x \in H\}$, and $\mathbf{a}$ denotes the hyper parameters of $\boldsymbol{\pi}$ and is expressed by

$$\mathbf{a} \equiv \{a_{\mathcal{S}[x]}|x \in H\}. \tag{24}$$

Although Bayes estimation learns the posterior distribution, Bayes estimation in models including latent variables is computationally intractable due to integrals. VB instead estimates the approximate posterior distribution $Q(\cdots|\mathcal{D})$ with a variational approximation by using the decomposable assumption (this corresponds to the mean field approximation). This is the assumption that the posterior distribution can be decomposed as

$$Q(\mathbf{X},\boldsymbol{\beta},\boldsymbol{\pi},h|\mathcal{D}) = Q(\mathbf{X}|\mathcal{D},h)Q(\boldsymbol{\beta}|\mathcal{D},h)Q(\boldsymbol{\pi}|\mathcal{D},h)Q(h|\mathcal{D}). \tag{25}$$

We describe a graphical model for PCFG-LA with VB learning in Fig. 4. By using the decomposable assumption, the approximate posterior of $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ is represented by (26) and (27) using the variational method. The detailed derivation of these posteriors are explained in Appendix I.

$$\begin{aligned}Q(\boldsymbol{\beta}_{\mathcal{S}[x]}|\mathcal{D},h) &\propto P(\boldsymbol{\beta}_{\mathcal{S}[x]}|h)\\ &\times \exp\langle \log P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)\rangle_{Q(\mathbf{X}|\mathcal{D},h),Q(\boldsymbol{\beta}_{-\mathcal{S}[x]}|\mathcal{D},h),Q(\boldsymbol{\pi}|\mathcal{D},h)}\end{aligned}\tag{26}$$

$$\begin{aligned}Q(\boldsymbol{\pi}|\mathcal{D},h) &\propto P(\boldsymbol{\pi}|h)\\ &\times \exp\langle \log P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)\rangle_{Q(\mathbf{X}|\mathcal{D},h),Q(\boldsymbol{\beta}|\mathcal{D},h)}\,.\end{aligned}\tag{27}$$

In (26) and (27), $\langle \cdots \rangle_{Q(\cdots)}$ represents the expected value with respect to $Q(\cdots)$, while $\langle \cdots \rangle_{Q(\boldsymbol{\beta}_{-\mathcal{S}[x]}|\mathcal{D},h)}$ denotes the calculation of the expected value with

$$\langle \cdots \rangle_{Q(\boldsymbol{\beta}_{-\mathcal{S}[x]}|\mathcal{D},h)} \equiv \int \prod_{w \in H, w \neq x} \left\{ d\boldsymbol{\beta}_{\mathcal{S}[w]} Q(\boldsymbol{\beta}_{\mathcal{S}[w]}|\mathcal{D},h) \right\} (\cdots).$$

From these equations, we obtain $Q(\boldsymbol{\beta}_{\mathcal{S}[x]}|\mathcal{D},h)$

$$Q(\boldsymbol{\beta}_{\mathcal{S}[x]}|\mathcal{D},h) = \mathrm{Dir}(\boldsymbol{\beta}_{\mathcal{S}[x]}; \breve{\mathbf{b}}_{\mathcal{S}[x]}) \tag{28}$$

$$\breve{\mathbf{b}}_{\mathcal{S}[x]} \equiv \{ \breve{b}_{\mathcal{S}[x]\to\alpha} | \alpha \in \mathcal{R}_r[H] \} \tag{29}$$

$$\breve{b}_{\mathcal{S}[x]\to\alpha} = \sum_{i=1}^{N} \sum_{X_i} Q(X_i|T_i,h) \cdot c(\mathcal{S}[x] \to \alpha; T_i, X_i) + b_{\mathcal{S}[x]\to\alpha} \tag{30}$$

and the approximate posterior distribution $Q(\boldsymbol{\pi}|\mathcal{D},h)$

$$Q(\boldsymbol{\pi}|\mathcal{D},h) = \mathrm{Dir}(\boldsymbol{\pi}; \breve{\mathbf{a}}) \tag{31}$$

$$\breve{\mathbf{a}} \equiv \{ \breve{a}_{\mathcal{S}[x]} | x \in H \} \tag{32}$$

$$\breve{a}_{\mathcal{S}[x]} = \sum_{i=1}^{N} \sum_{X_i} Q(X_i|T_i,h) \cdot \delta(x; T_i, X_i) + a_{\mathcal{S}[x]}. \tag{33}$$

The posterior distribution of latent variables $Q(\mathbf{X}|\mathcal{D},h) = \prod_{i=1}^{N} Q(X_i|T_i,h)$ can be calculated in a similar way

$$Q(X_i|T_i,h) \propto \exp \langle \log P(T_i, X_i|\boldsymbol{\beta},\boldsymbol{\pi},h) \rangle_{Q(\boldsymbol{\pi}|\mathcal{D},h), Q(\boldsymbol{\beta}|\mathcal{D},h)}. \tag{34}$$

From the above relation, the approximate posterior distribution of latent variables is derived as

$$Q(X_i|T_i,h) = \frac{1}{\mathcal{Z}(T_i)} \left\{ \prod_{x \in H} \widetilde{\pi}(\mathcal{S}[x])^{\delta(x;T_i,X_i)} \right\} \times \left\{ \prod_{r \in \mathcal{R}[H]} \widetilde{\beta}(r)^{c(r;T_i,X_i)} \right\} \tag{35}$$

$$\widetilde{\beta}(\mathcal{S}[x] \to \alpha) \equiv \exp \left[\psi(\breve{b}_{\mathcal{S}[x]\to\alpha}) - \psi\left(\sum_{\alpha \in \mathcal{R}_r[H]} \breve{b}_{\mathcal{S}[x]\to\alpha} \right) \right] \tag{36}$$

$$\widetilde{\pi}(\mathcal{S}[x]) \equiv \exp \left[\psi(\breve{a}_{\mathcal{S}[x]}) - \psi\left(\sum_{x \in H} \breve{a}_{\mathcal{S}[x]} \right) \right]. \tag{37}$$

In these equations, $\psi(\cdot)$ is the digamma function, and $\mathcal{Z}(T_i)$ is a normalizing constant (for the normalizing condition $\sum_{X_i} Q(X_i|T_i,h) = 1$). We can see that (35) is very similar to (2): If we substitute $\widetilde{\boldsymbol{\beta}}$ and $\widetilde{\boldsymbol{\pi}}$ to $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$, respectively, in the expression for $P(X_i|T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)$ in (2), we obtain the expression for $Q(X_i|T_i,h)$ given in (35).[2] Thus $\mathcal{Z}(T_i)$ corresponds to $P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)$ in (2) and can be calculated from (11)

$$\mathcal{Z}(T_i) = \sum_{x \in H} \widetilde{\pi}(\mathcal{S}[x]) b^1_{T_i}(x; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h). \tag{38}$$

[2]Note that parameters $\widetilde{\beta}$ and $\widetilde{\pi}$ are not normalized.

In (30) and (33), direct calculations of $\sum_{X_i} Q(X_i|T_i,h) \cdot c(\mathcal{S}[x] \to \alpha; X_i, T_i)$ and $\sum_{X_i} Q(X_i|T_i,h) \cdot \delta(x; T_i, X_i)$ are intractable when the number of nodes becomes large. These terms can instead be calculated using forward–backward probabilities (see Appendix III). We set

$$\widetilde{c}(\mathcal{S}[x] \to \alpha; T_i) \equiv \sum_{X_i} Q(X_i|T_i,h) \cdot c(\mathcal{S}[x] \to \alpha; T_i, X_i)$$

$$\widetilde{\delta}(x; T_i) \equiv \sum_{X_i} Q(X_i|T_i,h) \cdot \delta(x; T_i, X_i)$$

and finally, we have

$$\widetilde{c}(\mathcal{S}[x] \to g\,\mathcal{S}[y] \ldots \mathcal{S}[y]; T_i) = \frac{\widetilde{\beta}(\mathcal{S}[x] \to g\,\mathcal{S}[y] \ldots \mathcal{S}[y])}{\mathcal{Z}(T_i)} \times \sum_{j \in \mathrm{cover}(g,T_i)} f^j_{T_i}(x; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h) \times \prod_{k \in \mathrm{ch}(j,T_i)} b^k_{T_i}(y; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h) \tag{39}$$

$$\widetilde{\delta}(x; T_i) = \frac{\widetilde{\pi}(\mathcal{S}[x])}{\mathcal{Z}(T_i)} b^1_{T_i}(x; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h). \tag{40}$$

We can see similarities between (39) and (40) and (15) and (16).

V. Generation of New Individuals

Let $\mathcal{P}_g$ be the population of generation g and $\mathcal{D}_g$ be the set of promising individuals at generation g (learning data, which is represented by $\mathcal{D}$ in the previous sections). EDA generally generates $\mathcal{P}_{g+1}$ by sampling from the predictive posterior distribution $P(T, X|\mathcal{D}_g)$.

For the case of the EM algorithm, the predictive posterior distribution is represented by point values [see (41) below]. Because the EM algorithm is a point estimation method, the predictive posterior distribution can be represented by the point estimation values $\boldsymbol{\beta}^*_{\mathrm{EM}}$ and $\boldsymbol{\pi}^*_{\mathrm{EM}}$ as

$$P(T, X|\mathcal{D}_g) = P(T, X|\boldsymbol{\beta}^*_{\mathrm{EM}}, \boldsymbol{\pi}^*_{\mathrm{EM}}, h). \tag{41}$$

In this equation, $\boldsymbol{\beta}^*_{\mathrm{EM}}$ and $\boldsymbol{\pi}^*_{\mathrm{EM}}$ are limit values of the parameters obtained by updating (15) and (16), and h is the annotation size provided in advance. The population $\mathcal{P}_{g+1}$ obeys the posterior distribution expressed by (41) and can be generated using probabilistic logic sampling (PLS), which is computationally cheap.

For the case of VB, the predictive posterior distribution is calculated in an ensemble fashion. Let $\mathcal{H}$ be the set of possible annotation sizes h. The predictive posterior distribution can be expressed as

$$\begin{aligned} P(T, X|\mathcal{D}_g) &= \sum_{h \in \mathcal{H}} \int d\boldsymbol{\beta} d\boldsymbol{\pi}\, P(T, X|\boldsymbol{\beta},\boldsymbol{\pi},h) P(\boldsymbol{\beta},\boldsymbol{\pi},h|\mathcal{D}_g) \\ &= \sum_{h \in \mathcal{H}} \int d\boldsymbol{\beta} d\boldsymbol{\pi}\, P(T, X|\boldsymbol{\beta},\boldsymbol{\pi},h) P(\boldsymbol{\beta},\boldsymbol{\pi}|\mathcal{D}_g,h) P(h|\mathcal{D}_g). \end{aligned} \tag{42}$$

Because it is not possible to sum over all possible models $\mathcal{H}$, maximum *a posteriori* (MAP) approximation is adopted to evaluate (42). That is, only the value of h that yields the highest $P(h|\mathcal{D}_g)$ is used

$$h_g^{\mathrm{opt}} \equiv \operatorname*{argmax}_h P(h|\mathcal{D}_g). \tag{43}$$

Applying (43) and substituting the approximate posterior distribution $Q(\cdots|\mathcal{D}_g)$ to the true posterior distributions $P(\cdots|\mathcal{D}_g)$, we finally obtain

$$\begin{aligned} P\left(T, X|\mathcal{D}_g\right) &\simeq \int d\boldsymbol{\beta} d\boldsymbol{\pi}\, P\left(T, X|\boldsymbol{\beta}, \boldsymbol{\pi}, h_g^{\mathrm{opt}}\right) P\left(\boldsymbol{\beta}, \boldsymbol{\pi}|\mathcal{D}_g, h_g^{\mathrm{opt}}\right) \\ &\simeq \int d\boldsymbol{\beta} d\boldsymbol{\pi}\, P\left(T, X|\boldsymbol{\beta}, \boldsymbol{\pi}\right) Q^*\left(\boldsymbol{\beta}|\mathcal{D}, h_g^{\mathrm{opt}}\right) Q^*\left(\boldsymbol{\pi}|\mathcal{D}, h_g^{\mathrm{opt}}\right) \end{aligned} \tag{44}$$

where $Q^*(\cdots|\mathcal{D})$ is the limit distribution obtained using hyper parameter update formulae. From (44), we have further

$$\begin{aligned} &P(T, X|\mathcal{D}_g) \\ &\propto \frac{\prod_{x\in H}\Gamma\left(\delta(x;T,X)+\check{a}^*_{\mathcal{S}[x]}\right)}{\Gamma\left(\sum_{x\in H}\left(\delta(x;T,X)+\check{a}^*_{\mathcal{S}[x]}\right)\right)} \\ &\times \prod_{x\in H}\frac{\prod_{\alpha\in\mathcal{R}_r[H]}\Gamma\left(c(\mathcal{S}[x]\to\alpha;T,X)+\check{b}^*_{\mathcal{S}[x]\to\alpha}\right)}{\Gamma\left(\sum_{\alpha\in\mathcal{R}_r[H]}\left(c(\mathcal{S}[x]\to\alpha;T,X)+\check{b}^*_{\mathcal{S}[x]\to\alpha}\right)\right)}. \end{aligned} \tag{45}$$

However, because (45) is not decomposed, Gibbs sampling is required to sample this equation. Since Gibbs sampling is a highly computationally expensive method, we approximate the predictive posterior distribution with the MAP point value. That is

$$\begin{aligned} &\int d\boldsymbol{\beta} d\boldsymbol{\pi}\, P\left(T, X|\boldsymbol{\beta}, \boldsymbol{\pi}, h_g^{\mathrm{opt}}\right) Q^*\left(\boldsymbol{\beta}|\mathcal{D}, h_g^{\mathrm{opt}}\right) Q^*\left(\boldsymbol{\pi}|\mathcal{D}, h_g^{\mathrm{opt}}\right) \\ &\simeq P\left(T, X|\boldsymbol{\beta}^*_{\mathrm{MAP}}, \boldsymbol{\pi}^*_{\mathrm{MAP}}, h_g^{\mathrm{opt}}\right) \end{aligned} \tag{46}$$

where $\boldsymbol{\beta}^*_{\mathrm{MAP}}$ and $\boldsymbol{\pi}^*_{\mathrm{MAP}}$ are MAP values of $Q^*(\boldsymbol{\beta}|\mathcal{D}, h_g^{\mathrm{opt}})$ and $Q^*(\boldsymbol{\pi}|\mathcal{D}, h_g^{\mathrm{opt}})$, respectively. By applying Lagrange's method, the MAP values in the Dirichlet distribution $Q^*(\cdots|\mathcal{D})$ can be expressed as

$$\pi^*_{\mathrm{MAP}}(\mathcal{S}[x]) = \frac{\check{a}^*_{\mathcal{S}[x]}-1}{\sum_{x\in H}\left(\check{a}^*_{\mathcal{S}[x]}-1\right)} \tag{47}$$

$$\beta^*_{\mathrm{MAP}}(\mathcal{S}[x]\to\alpha) = \frac{\check{b}^*_{\mathcal{S}[x]\to\alpha}-1}{\sum_{\alpha\in\mathcal{R}_r[H]}\left(\check{b}^*_{\mathcal{S}[x]\to\alpha}-1\right)}. \tag{48}$$

Although the use of MAP significantly reduces computational costs, this is at the expense of losing one of the advantages of Bayes estimation, namely that of minimization of the generalization error.

VI. Estimation of Optimal Annotation Size

In Section V, we approximated the predictive posterior distribution of VB using the optimum annotation size h_g^{opt}.

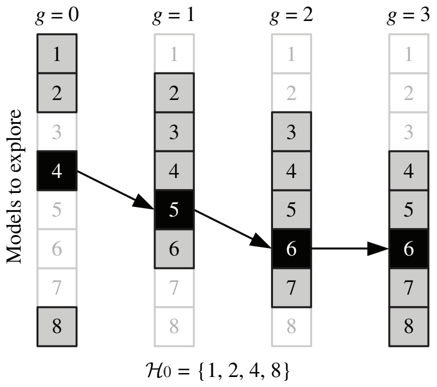

Fig. 5. Model search. The symbol g denotes the generation number, and cells shaded in gray are those whose corresponding values of $\mathcal{F}_h^*[Q]$ will be computed, while the cells shaded in black denote the optimum annotation size h_g^{opt} at each generation ($h_{\mathrm{range}} = 2$).

By applying Lagrange method to (54) under a constaint $\sum Q(h|\mathcal{D}) = 1$, we obtain

$$Q^*(h|\mathcal{D}) = \frac{P(h)\exp\left(\mathcal{F}_h^*[Q]\right)}{\sum_\ell P(\ell)\exp\left(\mathcal{F}_\ell^*[Q]\right)}. \tag{49}$$

If the prior distribution on the annotation size $P(h)$ is the uniform distribution, the best annotation size yields the maximum $\mathcal{F}_h^*[Q]$. The optimum annotation size is obtained as

$$\begin{aligned} h_g^{\mathrm{opt}} &= \operatorname*{argmax}_h Q^*(h|\mathcal{D}) \\ &= \operatorname*{argmax}_h \mathcal{F}_h^*[Q] \end{aligned}$$

where $\mathcal{F}_h^*[Q]$ is calculated in Appendix IV. We have to search over all possible annotation sizes. This means that at each generation we have to calculate $\mathcal{F}_1^*[Q]$, $\mathcal{F}_2^*[Q]$, $\mathcal{F}_3^*[Q]\ldots$ and select the value of h that yields the maximum $\mathcal{F}_h^*[Q]$. However, selecting the optimum annotation size in this way is a prohibitively expensive computational task. Thus we adopted the following two heuristics to reduce the computational cost. We let $\mathcal{H}_g$ be a set of annotation sizes to explore for generation g.

1) **Sparse Exploration**
At the initial generation, we calculate the lower bound $\mathcal{F}_h^*[Q]$ only sparsely. Thus, if $\mathcal{H}_0 = \{1, 2, 4, 8\}$, then PAGE-VB calculates $\mathcal{F}_1^*[Q]$, $\mathcal{F}_2^*[Q]$, $\mathcal{F}_4^*[Q]$, $\mathcal{F}_8^*[Q]$ only for the initial generation and selects the optimum annotation size as that which yields the highest $\mathcal{F}_h^*[Q]$.
2) **Exploration Restriction**
We assume that the models of neighboring generations have similar models and that the optimum annotation size at $g + 1$ is similar to that at g. At generation $g + 1$, PAGE-VB calculates only the h_{range} neighboring annotation sizes of h_g^{opt} obtained for generation g (see Fig. 5)

$$\mathcal{H}_{g+1} = \left\{h_g^{\mathrm{opt}} - h_{\mathrm{range}}, \ldots, h_g^{\mathrm{opt}}, \ldots, h_g^{\mathrm{opt}} + h_{\mathrm{range}}\right\}.$$

For example, if the optimum annotation size at generation g is $h_g^{\mathrm{opt}} = 5$ and $h_{\mathrm{range}} = 2$, PAGE-VB

Algorithm 1: Parameter update formula for PAGE-EM.

```
1) Initialize parameters $\boldsymbol{\beta}^{(0)}, \boldsymbol{\pi}^{(0)}$
   $\beta(\mathcal{S}[x] \to g\,\mathcal{S}[y]\ldots\mathcal{S}[y]) \propto \frac{e^{-\kappa}}{h-1}\beta(\mathcal{S} \to g\,\mathcal{S}\ldots\mathcal{S})$,
   $\beta(\mathcal{S}[x] \to g) \propto e^{-\kappa}\beta(\mathcal{S}[x] \to g)\ \ (\kappa \in [-\log 3, \log 3])$,
   $\pi(\mathcal{S}[x]) \propto \upsilon \in [0,1]$.
2) Update parameters using the following equations:
   $\pi^{(t+1)}(\mathcal{S}[x]) \propto \pi^{(t)}(\mathcal{S}[x]) \sum_{i=1}^{N} \frac{b^1_{T_i}(x; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)}{P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)}$
   $\beta^{(t+1)}(\mathcal{S}[x] \to g\,\mathcal{S}[y]\ldots\mathcal{S}[y]) \propto \beta^{(t)}(\mathcal{S}[x] \to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])$
   $\times \sum_{i=1}^{N}\Bigg[\frac{1}{P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)} \times \sum_{j \in \mathrm{cover}(g,T_i)} \Bigg\{ f^j_{T_i}(x; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)$
   $\times \prod_{k \in \mathrm{ch}(j,T_i)} b^k_{T_i}(y; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)\Bigg\}\Bigg]$.
3) If the update termination criteria are not met, return to 2.
   Otherwise stop updating.
Termination criteria are listed below
($L^{(t)}(\Theta; \mathcal{D}) = \sum_{i=1}^{N} \log P(T_i; \boldsymbol{\beta}^{(t)}, \boldsymbol{\pi}^{(t)}, h)$).
1) $t \geq 10$
2) $|L^{(t+1)}(\Theta; \mathcal{D}) - L^{(t)}(\Theta; \mathcal{D})| < 0.005 \times |L^{(t)}(\Theta; \mathcal{D})|$
```

Algorithm 2: Distribution (hyper-parameters) update formula for PAGE-VB.

```
1) Initialize hyper parameters $\mathbf{a}^{(0)} = \mathbf{a}, \mathbf{b}^{(0)} = \mathbf{b}$
   $a_{\mathcal{S}[x]} \in [1,2]$
   $b_{\mathcal{S}[x] \to g\,\mathcal{S}[y]\ldots\mathcal{S}[y]} \in [1,2]\ \ (x \neq y)$
   $b_{\mathcal{S}[x] \to g\,\mathcal{S}[x]\ldots\mathcal{S}[x]} = 1$
1) Update distribution (hyper-parameters) using the following equations:
   $Q^{(t+1)}\left(\boldsymbol{\beta}_{\mathcal{S}[x]}|\mathcal{D}, h\right) = \mathrm{Dir}\left(\boldsymbol{\beta}_{\mathcal{S}[x]}; \mathbf{b}^{(t+1)}_{\mathcal{S}[x]}\right)$
   $\mathbf{b}^{(t+1)}_{\mathcal{S}[x]} \equiv \left\{\mathbf{b}^{(t+1)}_{\mathcal{S}[x] \to \alpha} | \mathcal{S}[x] \to \alpha \in \mathcal{R}[H]\right\}$
   $b^{(t+1)}_{\mathcal{S}[x] \to \alpha} = \sum_{i=1}^{N} \sum_{X_i} \left\{ Q^{(t)}(X_i|T_i, h) \cdot c(\mathcal{S}[x] \to \alpha; T_i, X_i)\right\} + b_{\mathcal{S}[x] \to \alpha}$,
   $Q^{(t+1)}(\boldsymbol{\pi}|\mathcal{D}, h) = \mathrm{Dir}\left(\boldsymbol{\pi}; \mathbf{a}^{(t+1)}\right)$
   $\mathbf{a}^{(t+1)} \equiv \left\{a^{(t+1)}_{\mathcal{S}[x]} | x \in H\right\}$
   $a^{(t+1)}_{\mathcal{S}[x]} = \sum_{i=1}^{N} \sum_{X_i} \left\{ Q^{(t)}(X_i|T_i, h) \cdot \delta(x; T_i, X_i)\right\} + a_{\mathcal{S}[x]}$.
2) If the update termination criteria are not met, return
   to 2. Otherwise stop updating.
Termination criteria are listed below ($\mathcal{F}^{(t)}_h[Q]$ represents $\mathcal{F}_h[Q]$ at $t$th iteration).
1) $t \geq 20$
2) $|\mathcal{F}^{(t+1)}_h[Q] - \mathcal{F}^{(t)}_h[Q]| < 0.002 \times |\mathcal{F}^{(t)}_h[Q]|$
```

calculates only $\mathcal{F}^*_3[Q]$, $\mathcal{F}^*_4[Q]$, $\mathcal{F}^*_5[Q]$, $\mathcal{F}^*_6[Q]$, $\mathcal{F}^*_7[Q]$ at generation $g+1$.

Fig. 5 is a visualization of the above assumption. In this figure, the numbers in the cells denote the values of h. Gray cells express the values of h for which $\mathcal{F}^*_h[Q]$ is calculated, while the black cells denote the optimum annotation size (i.e. the value of h yielding the highest $\mathcal{F}^*_h[Q]$).

Algorithm 3: PAGE-EM

```
1) $\boldsymbol{\beta}_0, \boldsymbol{\pi}_0$ ←Initialize parameters
   $g \leftarrow 0$
2) $\mathcal{P}_0$ ←Generate $M$ initial individuals from $P(T, X|\boldsymbol{\beta}_0, \boldsymbol{\pi}_0, h)$
3) $\mathcal{D}_g$ ←Select $N \leq M$ promising solutions using a truncation
   selection
4) $\boldsymbol{\beta}^*_{\mathrm{EM}}, \boldsymbol{\pi}^*_{\mathrm{EM}}$ ←Estimate parameters with $\mathcal{D}_g$ using the EM
   algorithm (Algorithm 1)
5) $\mathcal{P}_{g+1}$ ←Generate $M$ new individuals from $P(T, X|\boldsymbol{\beta}^*_{\mathrm{EM}}, \boldsymbol{\pi}^*_{\mathrm{EM}}, h)$
   $g \leftarrow g+1$
Steps from 3 to 5 are repeated until termination criteria are met.
```

Algorithm 4: PAGE-VB

```
1) $\boldsymbol{\beta}_0, \boldsymbol{\pi}_0$ ←Initialize parameters
   $g \leftarrow 0$
2) $\mathcal{P}_0$ ←Generate $M$ initial individuals from $P(T, X|\boldsymbol{\beta}_0, \boldsymbol{\pi}_0, h=1)$
3) $\mathcal{D}_g$ ←Select $N \leq M$ promising solutions using a truncation
   selection
4) $\Pi_h \equiv \{\breve{\mathbf{a}}^*, \breve{\mathbf{b}}^*\}$ ←Calculate hyper-parameters for each $h \in \mathcal{H}_g$
   using $\mathcal{D}_g$ (Algorithm 2)
5) $\mathcal{F}^*_h[Q]$ ←Calculate lower bound for each $h \in \mathcal{H}_g$ using $\Pi_h$
6) $h^{\mathrm{opt}}_g$ ←Select the best annotation size by $\mathrm{argmax}_h\, \mathcal{F}^*_h[Q]$
7) $\boldsymbol{\beta}^*_{\mathrm{MAP}}, \boldsymbol{\pi}^*_{\mathrm{MAP}}$ ←Calculate MAP parameters with $\Pi_{h^{\mathrm{opt}}_g}$
8) $\mathcal{P}_{g+1}$ ←Generate $M$ new individuals from
       $P(T, X|\boldsymbol{\beta}^*_{\mathrm{MAP}}, \boldsymbol{\pi}^*_{\mathrm{MAP}}, h^{\mathrm{opt}}_g)$
   $g \leftarrow g+1$
Steps from 3 to 8 are repeated until termination criteria are met.
```

VII. Flowcharts

Algorithms 1 and 2 describe procedures for EM algorithm and VB, respectively. High-level pseudo codes for PAGE-EM and PAGE-VB are shown in Algorithms 3 and 4. For both algorithms, a truncation selection is used to select promising solutions. Thus, the best $M \times P_s$ individuals are selected as learning data for the EM algorithm or the VB learning.

VIII. Experiments

In this section, we examine the behavior and search performance of the proposed methods PAGE-EM and PAGE-VB. We used the royal tree problem at level d, the royal tree problem at level e, the DMAX problem with weak deceptiveness, and the DMAX problem with strong deceptiveness. We performed 50 runs for each experiment with each method. We used a t-test (Welch, two-tailed) to evaluate each experiment for any statistically significant differences between the results of the different approaches. Main parameter setups for PAGE-EM and PAGE-VB are listed in Table II.

A. Royal Tree Problem at Level d

The aim of this experiment is to investigate the effect of annotation size on search performance and to analyze the search mechanism. In this experiment, the two proposed methods PAGE-EM and PAGE-VB are applied along with PCFG-GP.

1) **PAGE-EM**

   PAGE-EM uses PCFG-LA together with the EM algorithm. As shown in the previous section, h has to be

TABLE II
MAIN PARAMETERS FOR PAGE-EM AND PAGE-VB. DMAX WEAK AND DMAX STRONG DENOTE DMAX PROBLEM WITH WEAK DECEPTIVENESS AND DMAX PROBLEM WITH STRONG DECEPTIVENESS, RESPECTIVELY

PAGE-EM		
Variable	Parameter	Value
M	Population size	1000 (Royal Tree at level d, DMAX weak, DMAX strong) 2500 (Royal Tree at level e)
h	Annotation size	$2, 4, 8, 16$ (Royal Tree at level d) 16 (Royal Tree at level e, DMAX weak) 8 (DMAX strong)
D_P	Maximum depth	5 (Royal Tree at level d, DMAX weak) 6 (Royal Tree at level e) 4 (DMAX strong)
P_s	Selection rate	0.1 (Royal Tree at level d, DMAX weak, DMAX strong) 0.05 (Royal Tree at level e)
P_e	Elite rate	0.1 (Royal Tree at level d, DMAX weak, DMAX strong) 0.05 (Royal Tree at level e)

PAGE-VB		
Variable	Parameter	Value
M	Population size	1000 (Royal Tree level d, DMAX weak, DMAX strong) 4000 (Royal Tree level e)
$\mathcal{H}_0$	Annotation to explore	$\{1, 2, 4, 8, 16\}$
h_{range}	Annotation exploration range	2
D_P	Maximum depth	5 (Royal Tree level d, DMAX weak) 6 (Royal Tree level e) 4 (DMAX strong)
P_s	Selection rate	0.1 (Royal Tree level d, DMAX weak, DMAX strong) 0.05 (Royal Tree level e)
P_e	Elite rate	0.1 (Royal Tree level d, DMAX weak, DMAX strong) 0.05 (Royal Tree level e)

specified in advance. In this experiment, we considered four values ($h = 2, 4, 8, 16$).

2) **PAGE-VB**
PAGE-VB uses PCFG-LA with VB learning. This algorithm automatically estimates the annotation size during the search. We set the possible annotation sizes for the initial generation to be $\mathcal{H}_0 = \{1, 2, 4, 8, 16\}$.
3) **PCFG-GP**
GP-EDA based on the conventional PCFG. No annotations are used in this model, which is practically identical to the scalar SG-GP.

The royal tree problem [26] is an extension of the royal road function [21], which is a celebrated benchmark test for GA methods. The royal tree problem uses a set of functions with alphabetical symbols which have increasing arity (a has arity 1, b has arity 2, and so on) and terminal nodes. Denoting the set of GP functions by $\mathscr{F}$ and the set of terminal nodes by $\mathscr{T}$, we thus have

$$\mathscr{F} = \{a, b, c, d, \ldots\}$$
$$\mathscr{T} = \{x\}.$$

The royal tree problem defines the state *perfect tree* at each level. The perfect tree at a given level is composed of the perfect tree that is one level smaller than the given level. Thus, the perfect tree of level c is composed of the perfect tree of level b. In perfect trees, alphabets of functions descend by 1 from a root to leaves in a tree. A function a has a terminal x. For more details, see [26].

For the present test problem we consider a royal tree problem at level d. The production rules for the royal tree problem at level d are shown in the list below, where symbols in lower case denote terminal symbols in CFG. Symbols $a-d$ denote function nodes, and x is a terminal node in GP.

$$\mathcal{S} \to a\,\mathcal{S}$$
$$\mathcal{S} \to b\,\mathcal{S}\,\mathcal{S}$$
$$\mathcal{S} \to c\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}$$
$$\mathcal{S} \to d\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}$$
$$\mathcal{S} \to x.$$

1) Results: Fig. 6 plots the number of fitness evaluations versus the cumulative frequency of successful runs. In this experiment, PCFG-GP failed to return the optimum value. The reason for this failure is straightforward. Since leaf nodes occur with the highest frequency in the tree structures, among all the production rules, the rule $\mathcal{S} \to x$ has the highest probability. Thus, using the conventional PCFG, the size of generated individuals tends to be small. On the other hand, our approach, i.e., PAGE-EM, uses production rules with latent annotations. In Table III, the production rules with annotations are described for $h = 8$ and $h = 16$. Note that we have not presented production rules that have small probabilities (smaller than 0.01) in the table due to limitations of space.

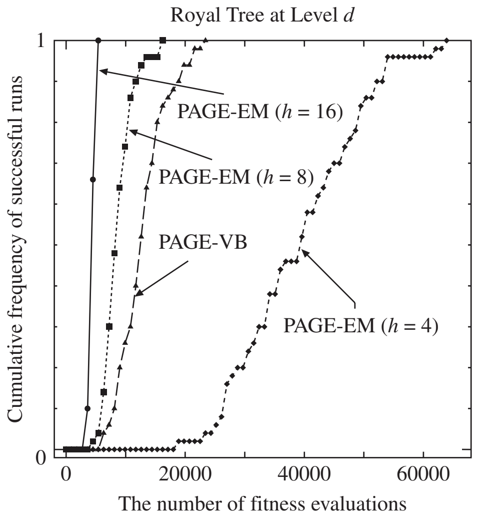

Fig. 6. Number of fitness evaluations versus cumulative frequency of successful runs for the royal tree problem at level d. The number attached to PAGE in parentheses denotes the annotation size h.

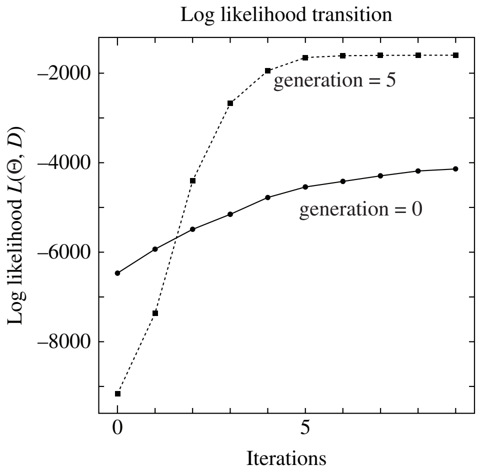

Fig. 7. Transition of log-likelihood for the royal tree problem at level d solved by PAGE-EM ($h = 16$). We show the transition for generations 0 and 5 (end of the search).

Referring to the probabilities corresponding to the annotations, the annotations are used to generate different symbols. For example, for $h = 16$, $\mathcal{S}[0]$, $\mathcal{S}[1]$, $\mathcal{S}[2]$, $\mathcal{S}[9]$, $\mathcal{S}[12]$, $\mathcal{S}[13]$, and $\mathcal{S}[14]$ generate x exclusively, while $\mathcal{S}[5]$ generates d exclusively. Using these production rules, we can grasp the sub-structures with a higher probability. In Table III, we have traced the highest probability rules from the root and have shaded these rules in gray. By using these gray rules, the optimum structures can be obtained with relatively high probability. The number of annotations h strongly affects the search performance of the proposed PAGE-EM algorithm. In Fig. 6, we can see that larger values for h yield better performance. In the case of smaller annotation sizes, there are many overlapping symbols. For example in Table III ($h = 8$), we can see that $\mathcal{S}[2]$ mainly generates $a\,\mathcal{S}[4]$. However, $\mathcal{S}[2]$ also generates b and c. This overlap makes the probability $P(T; \boldsymbol{\beta}, \boldsymbol{\pi}, h)$ of the optimum smaller and it is more difficult to find the optimum. By contrast, for larger values of h there are many redundant symbols, and so, there are fewer symbols that overlap in the manner of $\mathcal{S}[2]$ for the case of $h = 8$.

The search performance of PAGE-VB is poorer than that of PAGE-EM ($h = 16$). However, in PAGE-EM, trial and error are required to determine the optimum annotation size. By contrast, PAGE-VB automatically determines the annotation size during the search. For this problem, it is clear that the number of annotations greatly affects the search performance, with a better performance for larger annotation sizes. However, larger annotation sizes require significantly more computational time ($\sim O(h^2)$) and may cause an overfitting problem. PAGE-VB automatically estimates the annotation size by calculating $\mathcal{F}_h^*[Q]$. Fig. 9 shows $\mathcal{F}_h^*[Q]$ for each annotation size for generations 0 and 17. These values are obtained for a run without exploration restriction ($\mathcal{F}_h^*[Q]$ is calculated at every annotation size up to 16). As can been seen, $\mathcal{F}_h^*[Q]$ has peaks at $h = 5$ (generation 0) and $h = 6$ (generation 17). Accordingly, PAGE-VB selected these annotation sizes as the best annotation sizes. In this experiment, the search performance of PAGE-VB is inferior to that of PAGE-EM ($h = 16$). However, we can say that PAGE-VB is an effective approach if we consider the cost of determining the optimum annotation size. In Fig. 6, we can see a large difference between the values of $\mathcal{F}_h^*[Q]$ obtained for $h = 8$ and $h = 4$. The optimum annotation size estimated by VB method lies between these two annotation sizes. PAGE-VB balanced the search performance against the cost of model learning.

PAGE-EM estimates the parameters by means of the EM algorithm, which monotonically increases the likelihood. Fig. 7 describes the increase of log-likelihood [see (17)] for the royal tree problem at level d using PAGE-EM, in particular, the transition of generation 0 and generation 5. As can been seen from the figure, values for the parameters converge after around 10 iterations. The log-likelihood improvement at generation 5 is larger than that at generation 0, because the tree structures have converged toward the end of the search.

PAGE-VB also maximizes the lower bound using VB. Fig. 8 shows the transition of $\mathcal{F}_h[Q]$ (at generation 17 for $h = 5$ and at generation 0 for $h = 8$). These transitions resemble those of the log likelihood for the EM algorithm (see Fig. 7). Similar to the improvement in the log-likelihood for the later generation using the EM algorithm, the increase in the lower bound $\mathcal{F}_h[Q]$ for the later generation is due to the tree structures approaching convergence as the search progresses.

B. Royal Tree Problem at Level e

We next apply PAGE to the royal tree problem at level e. The aim of this experiment is to compare the search performance of PAGE with that of GMPE [32], which is a powerful GP-EDA inferring production rules from the learning data. We used the following four algorithms for comparison.

1) **PAGE-EM**
In this experiment, we ran PAGE-EM with $h = 16$. The population size was $M = 2500$, for which PAGE-EM

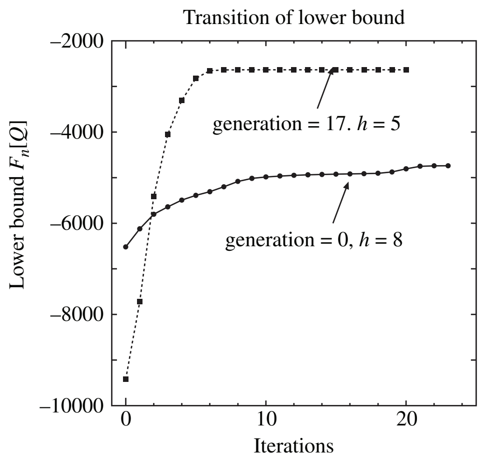

Fig. 8. Transition of the lower bound $\mathcal{F}_h[Q]$ for the royal tree problem at level d solved by PAGE-VB.

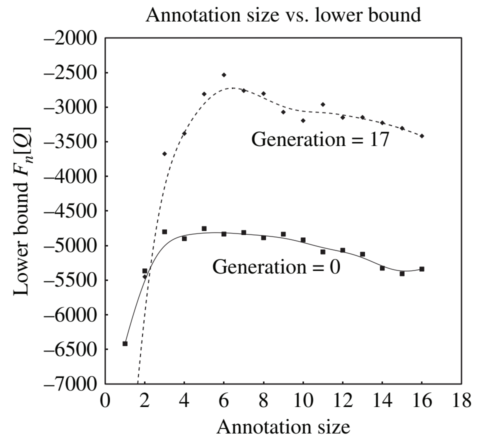

Fig. 9. Limit of the lower bound $\mathcal{F}_h^*[Q]$ plotted against annotation size for the royal tree problem at level d solved with PAGE-VB. Interpolations are carried out with splines.

was able to obtain the optimum of the royal tree problem at level e with a probability larger than 0.9.

2) **PAGE-VB**
We fixed the set of possible annotation sizes for the initial generation as $\mathcal{H}_0 = \{1, 2, 4, 8, 16\}$. The population size was $M = 4000$, for which PAGE-VB was able to obtain the optimum of the royal tree problem at level e with a probability larger than 0.9.

3) **GMPE**
GMPE is a GP-EDA capable of taking account of interactions among nodes. GMPE starts from the most specialized grammar which exclusively generates learning data. GMPE gradually merges two non-terminals to make the grammar more general. For further details on GMPE, see [32]. The result obtained by GMPE was scanned from Fig. 2 in [32]. In [32], values of $M = 60$ (population size) and $P_s = 0.5$ (selection rate, a truncate selection) were used.

4) **PCFG-GP**
GP-EDA is based on the conventional PCFG. No annotations are used in this model, which is practically identical to the scalar SG-GP.

TABLE III
PROBABILITIES OBTAINED FOR THE ROYAL TREE PROBLEM (LEVEL d) BY PAGE WITH $h = 8$ AND $h = 16$

Production rule (PAGE-EM, $h = 8$)	π, β
$\mathcal{S}[3]$	1.000
$\mathcal{S}[0] \to b\,\mathcal{S}[6]\,\mathcal{S}[6]$	0.018
$\mathcal{S}[0] \to c\,\mathcal{S}[6]\,\mathcal{S}[6]\,\mathcal{S}[6]$	0.972
$\mathcal{S}[1] \to x$	1.000
$\mathcal{S}[2] \to a\,\mathcal{S}[1]$	0.090
$\mathcal{S}[2] \to a\,\mathcal{S}[4]$	0.538
$\mathcal{S}[2] \to a\,\mathcal{S}[7]$	0.075
$\mathcal{S}[2] \to b\,\mathcal{S}[1]\,\mathcal{S}[1]$	0.014
$\mathcal{S}[2] \to b\,\mathcal{S}[4]\,\mathcal{S}[4]$	0.022
$\mathcal{S}[2] \to b\,\mathcal{S}[7]\,\mathcal{S}[7]$	0.025
$\mathcal{S}[2] \to c\,\mathcal{S}[4]\,\mathcal{S}[4]\,\mathcal{S}[4]$	0.059
$\mathcal{S}[2] \to c\,\mathcal{S}[7]\,\mathcal{S}[7]\,\mathcal{S}[7]$	0.032
$\mathcal{S}[2] \to d\,\mathcal{S}[4]\,\mathcal{S}[4]\,\mathcal{S}[4]\,\mathcal{S}[4]$	0.068
$\mathcal{S}[2] \to x$	0.061
$\mathcal{S}[3] \to d\,\mathcal{S}[0]\,\mathcal{S}[0]\,\mathcal{S}[0]\,\mathcal{S}[0]$	1.000
$\mathcal{S}[4] \to x$	1.000
$\mathcal{S}[5] \to a\,\mathcal{S}[1]$	0.161
$\mathcal{S}[5] \to a\,\mathcal{S}[4]$	0.468
$\mathcal{S}[5] \to a\,\mathcal{S}[7]$	0.356
$\mathcal{S}[6] \to a\,\mathcal{S}[2]$	0.012
$\mathcal{S}[6] \to b\,\mathcal{S}[2]\,\mathcal{S}[2]$	0.467
$\mathcal{S}[6] \to b\,\mathcal{S}[5]\,\mathcal{S}[5]$	0.399
$\mathcal{S}[6] \to c\,\mathcal{S}[2]\,\mathcal{S}[2]\,\mathcal{S}[2]$	0.030
$\mathcal{S}[6] \to d\,\mathcal{S}[2]\,\mathcal{S}[2]\,\mathcal{S}[2]\,\mathcal{S}[2]$	0.055
$\mathcal{S}[6] \to x$	0.029
$\mathcal{S}[7] \to x$	1.000

Production rule (PAGE-EM, $h = 16$)	π, β
$\mathcal{S}[5]$	1.000
$\mathcal{S}[0] \to x$	1.000
$\mathcal{S}[1] \to x$	1.000
$\mathcal{S}[2] \to x$	1.000
$\mathcal{S}[3] \to c\,\mathcal{S}[13]\,\mathcal{S}[13]\,\mathcal{S}[13]$	0.063
$\mathcal{S}[3] \to c\,\mathcal{S}[14]\,\mathcal{S}[14]\,\mathcal{S}[14]$	0.131
$\mathcal{S}[3] \to c\,\mathcal{S}[9]\,\mathcal{S}[9]\,\mathcal{S}[9]$	0.093
$\mathcal{S}[3] \to d\,\mathcal{S}[14]\,\mathcal{S}[14]\,\mathcal{S}[14]\,\mathcal{S}[14]$	0.054
$\mathcal{S}[3] \to d\,\mathcal{S}[9]\,\mathcal{S}[9]\,\mathcal{S}[9]\,\mathcal{S}[9]$	0.060
$\mathcal{S}[3] \to x$	0.329
$\mathcal{S}[4] \to a\,\mathcal{S}[1]$	0.154
$\mathcal{S}[4] \to a\,\mathcal{S}[14]$	0.257
$\mathcal{S}[4] \to a\,\mathcal{S}[2]$	0.189
$\mathcal{S}[4] \to a\,\mathcal{S}[9]$	0.238
$\mathcal{S}[5] \to d\,\mathcal{S}[10]\,\mathcal{S}[10]\,\mathcal{S}[10]\,\mathcal{S}[10]$	1.000
$\mathcal{S}[6] \to a\,\mathcal{S}[14]$	0.144
$\mathcal{S}[6] \to a\,\mathcal{S}[9]$	0.092
$\mathcal{S}[6] \to b\,\mathcal{S}[1]\,\mathcal{S}[1]$	0.070
$\mathcal{S}[6] \to b\,\mathcal{S}[2]\,\mathcal{S}[2]$	0.079
$\mathcal{S}[6] \to d\,\mathcal{S}[14]\,\mathcal{S}[14]\,\mathcal{S}[14]\,\mathcal{S}[14]$	0.167
$\mathcal{S}[7] \to a\,\mathcal{S}[14]$	0.152
$\mathcal{S}[7] \to a\,\mathcal{S}[9]$	0.184
$\mathcal{S}[7] \to c\,\mathcal{S}[9]\,\mathcal{S}[9]\,\mathcal{S}[9]$	0.071
$\mathcal{S}[7] \to x$	0.319
$\mathcal{S}[8] \to b\,\mathcal{S}[11]\,\mathcal{S}[11]$	0.188
$\mathcal{S}[8] \to b\,\mathcal{S}[4]\,\mathcal{S}[4]$	0.397
$\mathcal{S}[8] \to b\,\mathcal{S}[6]\,\mathcal{S}[6]$	0.079
$\mathcal{S}[8] \to c\,\mathcal{S}[3]\,\mathcal{S}[3]\,\mathcal{S}[3]$	0.053
$\mathcal{S}[9] \to x$	1.000
$\mathcal{S}[10] \to c\,\mathcal{S}[15]\,\mathcal{S}[15]\,\mathcal{S}[15]$	0.900
$\mathcal{S}[10] \to c\,\mathcal{S}[8]\,\mathcal{S}[8]\,\mathcal{S}[8]$	0.093
$\mathcal{S}[11] \to a\,\mathcal{S}[1]$	0.167
$\mathcal{S}[11] \to a\,\mathcal{S}[12]$	0.080
$\mathcal{S}[11] \to a\,\mathcal{S}[13]$	0.181
$\mathcal{S}[11] \to a\,\mathcal{S}[14]$	0.209
$\mathcal{S}[11] \to a\,\mathcal{S}[2]$	0.143
$\mathcal{S}[11] \to a\,\mathcal{S}[9]$	0.208
$\mathcal{S}[12] \to x$	0.993
$\mathcal{S}[13] \to x$	1.000
$\mathcal{S}[14] \to x$	1.000
$\mathcal{S}[15] \to b\,\mathcal{S}[11]\,\mathcal{S}[11]$	0.648
$\mathcal{S}[15] \to b\,\mathcal{S}[4]\,\mathcal{S}[4]$	0.239

TABLE IV
NUMBER OF AVERAGE FITNESS EVALUATIONS REQUIRED TO OBTAIN THE OPTIMUM IN ROYAL TREE PROBLEM AT LEVEL e (THE TOP 90% OF RESULTS WERE USED). SINCE THE RESULTS OF GMPE WERE INFERRED FROM A SCANNED IMAGE, THEY WERE SUBJECT TO ERRORS

Algorithm	Number of fitness evaluations	Sample standard deviation
PAGE-EM	61809	15341
PAGE-VB	263720	63528
GMPE	90525	79774
PCFG-GP	–	–

The fitness calculation for this problem is identical to that in level d. The royal tree problem at level e is more difficult than that at level d. The number of nodes contained in the optimum structure at level e is 326 which is about five times larger than that at level d (65 in level d). The production rules for the royal tree problem at level e are shown in the list below, where symbols in lower case denote terminal symbols in CFG. Symbols a–e denote function nodes, and x is a terminal node in GP

$$\begin{aligned}
\mathcal{S} &\to a\,\mathcal{S}\\
\mathcal{S} &\to b\,\mathcal{S}\,\mathcal{S}\\
\mathcal{S} &\to c\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\\
\mathcal{S} &\to d\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\\
\mathcal{S} &\to e\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\\
\mathcal{S} &\to x.
\end{aligned}$$

1) Results: We present the number of fitness evaluations required to obtain the optimum in Table IV. In this table, the top 90% of the runs are used for calculating the average. Fig. 10 plots the number of fitness evaluations versus the cumulative frequency of successful runs. We can see from these results that PAGE-EM yielded the best search performance. Fig. 10 shows that half the runs in GMPE obtained the optimum with a relatively small number of fitness evaluations. We can see that with 50% probability, GMPE is comparable with PAGE-EM ($h = 16$). However, 20% of the runs in GMPE required more fitness evaluations. PAGE-EM and PAGE-VB searched for the optimum, given larger population sizes ($M = 2500$ and 4000) and a small number of generations. On the other hand, GMPE searched with a smaller population ($M = 60$) and more generations. GMPE utilizes a highly flexible model which is inspired by the model merging method [34] proposed in NLP, which infers not only parameters but production rules from learning data as well. The search performance of GMPE is affected by the size of the initial population. Because GMPE learns production rules in a greedy fashion for each generation, this search is highly computationally expensive calculation. PAGE can handle a larger population size due to a certain approximation that it utilizes [see (6)]. With this assumption, the computational cost of PAGE is proportional to h^2. At the same time, PAGE learns the model in a hill-climbing fashion, because the EM algorithm and VB optimize values for the parameters by iteration. These approximations result in faster

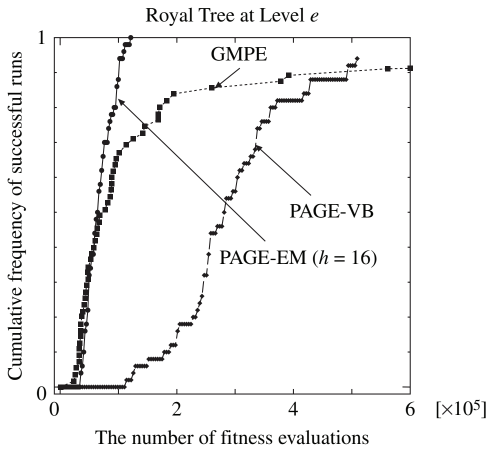

Fig. 10. Number of fitness evaluations versus the cumulative frequency of successful runs for the royal tree problem at level e.

TABLE V
RESULTS OF A t-TEST FOR THE ROYAL TREE PROBLEM AT LEVEL e. THE VALUE IN THE TABLE IS A p-VALUE

＼	PAGE-EM	PAGE-VB
PAGE-EM	＼	5.90×10^{-23}
PAGE-VB	–	＼

parameter learning by PAGE-EM, and as a result, PAGE-EM can handle more individuals than GMPE. From the viewpoint of the average number of fitness evaluations, PAGE-VB is inferior to GMPE. However, the probability of success for PAGE-VB is better than for GMPE.

Table V details the results of the t-tests. The top 90% of the results are used for t-test. Since we did not have raw data available for GMPE, we were unable to calculate its associated p-value.

C. DMAX Problem With Weak Deceptiveness

We applied our approach to the DMAX problem [10] to demonstrate the superiority of our proposal over other algorithms, including the conventional GP and POLE. The DMAX problem is an extended version of the MAX problem. Because the original MAX problem does not have deceptiveness, it is very easy to solve by GP-EDAs that do not consider interactions among nodes. Here, we instead consider the MAX problem extended by adding deceptiveness, which is referred to as the *deceptive MAX problem* (DMAX problem). We employed the GP function set $\mathscr{F}$ and terminal set $\mathscr{T}$ given by

$$\begin{aligned}
\mathscr{F} &= \{+_m, \times_m\}\\
\mathscr{T} &= \{\lambda, 0.95\}
\end{aligned} \tag{50}$$

with

$$\lambda^r = 1, \quad \lambda \in \mathbb{C}, \quad r \in \mathbb{N} \tag{51}$$

where $+_m$ and $\times_m$ are arity m versions of $+$ and $\times$, respectively, and λ is generally a complex value and $\lambda \neq 1$.

The main objective of the DMAX problem is identical to that of the original MAX problem: to find the functions that 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 (50). A fitness value of an individual is the real part of its function: If the value of a function is $v + wi$ (where $v, w \in \mathbb{R}, i = \sqrt{-1}$), then its fitness value is v. The DMAX problem can be represented by three parameters: m (arity), r (power), and D_P (maximum tree depth). By changing these three DMAX problem parameters, the difficulty of this problem can be tuned. We will separately discuss the DMAX with weak and strong deceptiveness in Section VIII-C and D, respectively.

In the previous section, we compared the search performance of PAGE with PCFG-GP and GMPE. In this section, we compare the search performance of PAGE with that of other program-optimization methods, including conventional GP and POLE. Specifically, we compare the following methods.

1) **PAGE-EM**
In this experiment, we ran PAGE-EM with $h = 16$. The population size was $M = 1000$.
2) **PAGE-VB**
We fixed the set of possible annotation sizes for the initial generation as $\mathcal{H}_0 = \{1, 2, 4, 8, 16\}$. The population size was again $M = 1000$.
3) **PCFG-GP**
The population size was also set to $M = 1000$.
4) **Simple GP**
This is a simple implementation of GP. In the experiments, $P_e = 0.01$ (elite rate), $P_c = 0.9$ (crossover rate), $P_m = 0$ (mutation rate), and $P_r = 0.09$ (reproduction rate) were used. Crossover points are selected in the following way: When applying crossover to two individuals, we select the first crossover point from function nodes with a probability of 0.9 and from terminals at 0.1. The second crossover point is selected under the condition that the depth of both individuals does not exceed the depth limitation. The population size was set at $M = 1000$, for which GP can obtain the optimum solution for the DMAX problem with a probability larger than 0.9. The tournament size was $t_s = 2$.
5) **POLE**
POLE [10], [11] is a state-of-the-art prototype tree-based GP-EDA employing a Bayesian network for estimating dependencies among nodes. The population size was set to $M = 4000$, for which POLE can obtain the optimum solution for the DMAX problem with a probability larger than 0.9. In POLE, the BIC metric is used to construct the Bayesian network. We set $R_P = 2$ (parent range), $P_e = 0.005$, and $P_s = 0.1$, and a truncate selection was used. For definitions of these parameters, see [11].

In this section, we first consider a case with $m = 3$, $r = 2$, and $D_P = 5$, which has relatively weak deceptiveness. In this paper, we denote this setting as the DMAX problem with weak deceptiveness.

For this problem, obtaining the optimum value is more complicated than obtaining it for the MAX problem [7], [15], which does not have deceptiveness. For the case of $m = 3, r = 2$, the DMAX problem has weak deceptiveness. For clarity, we describe the case with $D_P = 3$. In this parameter setup, $\lambda = -1$. First, we add λ with $+_3$ to make -3. Then we multiply this value with $\times_3$ to yield $(-3)^3 = -27$. However, -27 is negative, and it is not an optimum. Using 3×0.95 as a substitute for -3, then the maximum value is $(-3)^2(0.95 \times 3) = 25.65$. When $D_P = 5$, the maximum value is 7.24×10^{12}.

As described above, the DMAX problem with weak deceptiveness uses the following production rules:

$$\mathcal{S} \to +_3 \mathcal{S}\mathcal{S}\mathcal{S}$$
$$\mathcal{S} \to \times_3 \mathcal{S}\mathcal{S}\mathcal{S}$$
$$\mathcal{S} \to 0.95$$
$$\mathcal{S} \to -1.$$

1) Results: Table VI depicts the number of fitness evaluations carried out by the various methods. Fig. 11 describes the number of fitness evaluations versus the cumulative frequency of successful runs for the DMAX problem with weak deceptiveness. Results for PCFG-GP are not shown because this method was unable to solve this problem. Clearly, PAGE-EM and PAGE-VB perform better than GP and POLE. The DMAX problem used in this section has a deceptiveness when using the crossover operator in GP. In the introduction, we have argued that GA- and GP-type samplings are valid only for the case where two structurally similar individuals have similar fitness values. Because the DMAX problem with weak deceptiveness violates this assumption, GP-type sampling did

TABLE VI
NUMBER OF AVERAGE FITNESS EVALUATIONS REQUIRED TO OBTAIN THE OPTIMUM IN THE DMAX PROBLEM WITH WEAK DECEPTIVENESS

Algorithm	Number of fitness evaluations	Sample standard deviation
PAGE-EM	14 922	1287
PAGE-VB	25 542	5229
PCFG-GP	–	–
GP	82 332	20 874
POLE	72 516	4381

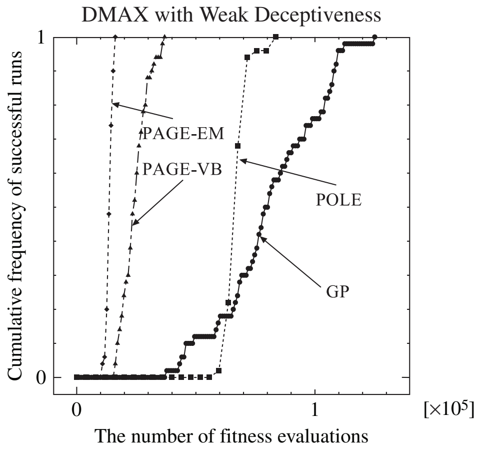

Fig. 11. Number of fitness evaluations versus cumulative frequency for successful runs with the DMAX problem with weak deceptiveness.

TABLE VII
RESULTS OF A t-TEST FOR DMAX PROBLEM WITH WEAK DECEPTIVENESS. THE VALUE IN THE TABLE IS A p-VALUE

＼	PAGE-EM	PAGE-VB	GP	POLE
PAGE-EM	＼	1.68×10^{-19}	1.16×10^{-27}	5.28×10^{-63}
PAGE-VB	–	＼	2.79×10^{-25}	1.27×10^{-68}
GP	–	–	＼	2.17×10^{-3}
POLE	–	–	–	＼

not perform very well. PAGE effectively estimates building blocks, and so the optimum solution for DMAX with weak deceptiveness could be obtained efficiently.

POLE is a state-of-the-art GP-EDA based on the prototype-tree approach. POLE required fewer fitness evaluations than GP. POLE employs a Bayesian network and can thus estimate interactions among nodes. POLE was able to identify the building blocks of this problem and, hence, was able to obtain the optimum solution of the DMAX problem with fewer fitness evaluations. However, prototype tree-based approaches cannot estimate building blocks that do not depend on their positions. In the DMAX problem, the building blocks $((-1) + (-1) + (-1))$ and $(0.95+0.95+0.95)$ do not depend on their absolute positions. Because PAGE-EM and PAGE-VB can estimate building blocks that are independent of their positions, PAGE-EM and PAGE-VB can reuse the estimated building blocks efficiently. This difference between POLE and PAGE explains the superiority of PAGE against POLE in this problem.

Table VII details the results of the t-tests. According to this table, the p-values for the differences in means between PAGE-EM, PAGE-VB, and GP, POLE are extremely small, indicating that the differences between these means are extremely statistically significant.

2) Effect of Selection Pressure: In this experiment, we used the parameter value $P_s = 0.1$ in the PAGE-EM and PAGE-VB. This means that the best $0.1 \times M$ individuals are used as learning data. In order to investigate the effect of P_s on a search performance, we repeated the experiment varying P_s while keeping all other parameters unchanged. Specifically, we considered values $P_s = 0.1, 0.2, 0.5$ and observed the probability of success at each generation to quantify the effect of P_s.

Table VIII depicts the number of fitness evaluations carried out by various selection rates. Fig. 12 shows the number of fitness evaluations versus cumulative frequency of successful runs for each value of P_s. As can been seen from these results, smaller values of P_s yield better performance. On the other hand, even at $P_s = 0.5$, PAGE-EM succeeded in obtaining the optimum. This result indicates that PAGE-EM is robust with respect to the selection rate.

For the case of PAGE-VB, the selection rate greatly affected the search performance. Half of the runs failed to obtain the optimum in $P_s = 0.5$. PAGE-EM only estimates parameters of production rules and does not learn annotation size. On the other hand, PAGE-VB estimates both annotation size and parameters. Since learning data with larger selection rate contain more noise, these noisy data affect the Bayes estimation of annotation size.

TABLE VIII
NUMBER OF AVERAGE FITNESS EVALUATIONS REQUIRED TO OBTAIN THE OPTIMUM FOR $P_s = 0.1, 0.2, 0.5$ IN THE DMAX PROBLEM WITH WEAK DECEPTIVENESS

Algorithm (P_s)	Number of fitness evaluations	Sample standard deviation
PAGE-EM (0.1)	14922	1287
PAGE-EM (0.2)	21024	2352
PAGE-EM (0.5)	56610	4804
PAGE-VB (0.1)	25542	5229
PAGE-VB (0.2)	34902	9364
PAGE-VB (0.5)	–	–

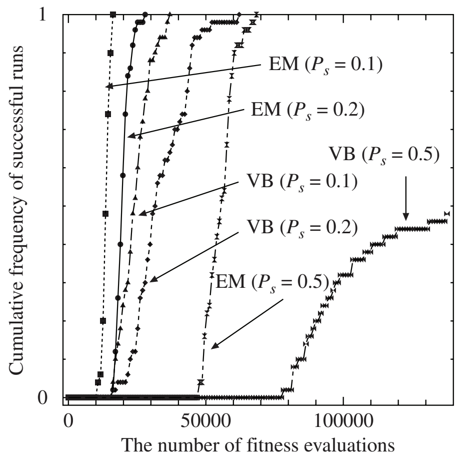

Fig. 12. Cumulative frequency of successful runs for $P_s = 0.1, 0.2, 0.5$ for the DMAX problem with weak deceptiveness. All parameters other than P_s are unchanged between experiments. EM and VB represent PAGE-EM and PAGE-VB, respectively.

D. DMAX Problem With Strong Deceptiveness

In this section, we compare the search performance of PAGE-EM and PAGE-VB with other program optimization methods, including conventional GP using the DMAX problem with strong deceptiveness. This problem shows strong deceptiveness for conventional GPs. Specifically, we compare the following methods.

1) **PAGE-EM**
In this experiment, we ran PAGE-EM with $h = 8$. The population size was $M = 1000$.
2) **PAGE-VB**
We fixed the set of possible annotation sizes for the initial generation as $\mathcal{H}_0 = \{1, 2, 4, 8, 16\}$. The population size was again $M = 1000$.
3) **PCFG-GP**
The population size was again set to $M = 1000$.
4) **Simple GP**
This GP is identical to that used in the previous section. In this experiment, $P_e = 0.005$, $P_c = 0.995$, $P_m = 0$, and $P_r = 0$ were used. The population size was set to $M = 16\,000$, with which GP can obtain the optimum of the DMAX problem with a probability larger than 0.9. The tournament size is $t_s = 2$.
5) **POLE**
POLE used in this problem is identical to that used in the previous section. The population size was set to

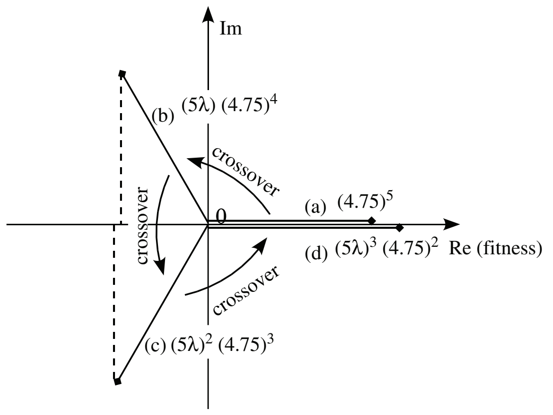

Fig. 13. Optimum of the DMAX problem with strong deceptiveness in the complex plane (see text).

TABLE IX
NUMBER OF AVERAGE FITNESS EVALUATIONS REQUIRED TO OBTAIN THE OPTIMUM WITH A PROBABILITY OF 90% IN DMAX PROBLEM WITH STRONG DECEPTIVENESS

	Number of fitness evaluations	Sample standard deviation
PAGE-EM	16043	1189
PAGE-VB	18293	1688
PCFG-GP	–	–
POLE	119572	4295
GP	1477376	293890

$M = 6300$, and with this setting POLE was able to obtain the optimum solution for the DMAX problem with a probability larger than 0.9.

As mentioned in the previous section, the strength of deceptiveness in the DMAX problem can be tuned with three parameters: m, r, and D_P. In this section, we use $m = 5$, $r = 3$, and $D_P = 4$, which correspond to strong deceptiveness. We show the optimum in this problem. For clarity, we describe the case with $D_P = 3$. In this parameter setup, $\lambda = \cos(2\pi/3) + i\sin(2\pi/3)$. First, we add λ with $+_5$ to make 5λ. Then, we multiply this value with $\times_5$ to yield $(5\lambda)^5 = 5^5\lambda^5 = 5^5\lambda^2$. However, $\mathrm{Re}(5^5\lambda^2)$ is negative and, therefore, is not a good solution. Thus two of the five values to be multiplied have to be real values. Using 5×0.95 as a substitute for 5λ, the maximum value is $(5\lambda)^3(0.95 \times 5)^2 = 2820.3125$. For $D_P = 4$, the maximum value is $(5\lambda)^{24}(0.95\times5) = 2.83\times10^{17}$. Fig. 13 shows a visualization of the values in the complex plane where fitness is plotted along the horizontal (real) axis. As can been seen, the replacement of one 0.95×5 with 5λ increases the argument by 120° in the complex plane. After three such replacements, the value returns to the same phase but has a larger absolute value.

The production rules employed in PAGE-EM, PAGE-VB and PCFG-GP can be represented by the following equations:

$$\mathcal{S} \to +_5 \mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}$$
$$\mathcal{S} \to \times_5 \mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}$$
$$\mathcal{S} \to 0.95$$
$$\mathcal{S} \to \lambda.$$

TABLE X
t-TEST FOR THE DMAX PROBLEM WITH STRONG DECEPTIVENESS. THE VALUE IN THE TABLE IS A p-VALUE

╲	PAGE-EM	PAGE-VB	GP	POLE
PAGE-EM	╲	2.82×10^{-9}	4.62×10^{-29}	1.10×10^{-61}
PAGE-VB	–	╲	4.89×10^{-29}	6.58×10^{-67}
GP	–	–	╲	7.13×10^{-28}
POLE	–	–	–	╲

TABLE XI
ESTIMATED PROBABILITIES FOR THE DMAX PROBLEM WITH STRONG DECEPTIVENESS WITH $h = 8$ FOUND BY PAGE-EM. RULES WITH PROBABILITIES SMALLER THAN 0.01 ARE OMITTED DUE TO LIMITATIONS OF SPACE

Production rule (PAGE-EM, $h = 8$)	π, β
$\mathcal{S}[4]$	1.000
$\mathcal{S}[0] \to \lambda$	0.983
$\mathcal{S}[0] \to 0.95$	0.017
$\mathcal{S}[1] \to \lambda$	0.941
$\mathcal{S}[1] \to 0.95$	0.059
$\mathcal{S}[2] \to \times_5 \mathcal{S}[6]\mathcal{S}[6]\mathcal{S}[6]\mathcal{S}[6]\mathcal{S}[6]$	1.000
$\mathcal{S}[3] \to \lambda$	0.989
$\mathcal{S}[3] \to 0.95$	0.011
$\mathcal{S}[4] \to \times_5 \mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]\mathcal{S}[2]$	1.000
$\mathcal{S}[5] \to \lambda$	0.995
$\mathcal{S}[6] \to +_5 \mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]\mathcal{S}[0]$	0.399
$\mathcal{S}[6] \to +_5 \mathcal{S}[1]\mathcal{S}[1]\mathcal{S}[1]\mathcal{S}[1]\mathcal{S}[1]$	0.010
$\mathcal{S}[6] \to +_5 \mathcal{S}[3]\mathcal{S}[3]\mathcal{S}[3]\mathcal{S}[3]\mathcal{S}[3]$	0.234
$\mathcal{S}[6] \to +_5 \mathcal{S}[5]\mathcal{S}[5]\mathcal{S}[5]\mathcal{S}[5]\mathcal{S}[5]$	0.096
$\mathcal{S}[6] \to +_5 \mathcal{S}[7]\mathcal{S}[7]\mathcal{S}[7]\mathcal{S}[7]\mathcal{S}[7]$	0.261
$\mathcal{S}[7] \to \lambda$	0.025
$\mathcal{S}[7] \to 0.95$	0.975

1) Results: We present the average number of fitness evaluations in Table IX. We used the top 90% of results to calculate averages. For the DMAX problem with strong deceptiveness, two building blocks have to be used to obtain the optimum: One is composed of five 0.95's, while the other is composed of five λ's. The production rules that generate 0.95 and λ have the same left-hand side symbol ($\mathcal{S} \to 0.95$ and $\mathcal{S} \to \lambda$). Because PCFG-GP does not distinguish between these two symbols, structures composed of a mixture of 0.95 and λ are generated (e.g., $\lambda + \lambda + 0.95 + \lambda + 0.95$). Thus, this algorithm, which does not use annotations, failed to obtain the optimum. On the other hand, for PAGE-EM and PAGE-VB, 0.95 and λ are generated from different production rules. Table XI shows the production rules estimated by PAGE-EM ($h = 8$). Clearly λ and 0.95 are generated from different symbols. The symbols $\mathcal{S}[0]$, $\mathcal{S}[1]$, $\mathcal{S}[3]$, and $\mathcal{S}[5]$ generate λ, and $\mathcal{S}[7]$ generates 0.95. Thus substructures that are mixtures of λ and 0.95 are not generated. Table X details the results of the t-tests. According to Table X, the p-values for differences between the PAGE-EM, PAGE-VB, GP, and POLE means are extremely small, indicating that the differences between these means are extremely statistically significant.

Fig. 14 plots the number of fitness evaluations versus cumulative frequency of successful runs for the four methods. Since PCFG-GP could not find the optimum even once, we omit

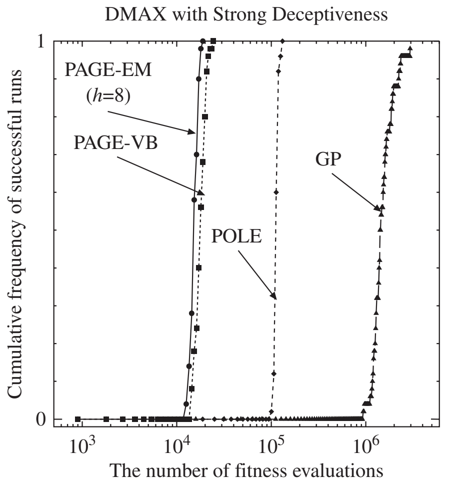

Fig. 14. Number of fitness evaluations versus cumulative frequency of successful runs for the DMAX problem with strong deceptiveness. The horizontal axis is in a log scale.

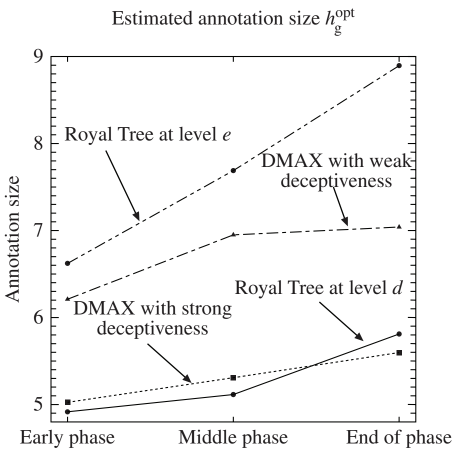

Fig. 15. Optimum annotation size estimated by PAGE-VB in three phases for each problem. Weak and strong denote weak and strong deceptiveness, respectively.

the results of PCFG-GP. Clearly, PAGE-EM and PAGE-VB perform much better in comparison with GP and POLE. The performance difference between PAGE and GP is much larger for this problem than for the DMAX with weak deceptiveness. As described in [10], this setting of the DMAX problem has very strong deceptiveness when using the crossover operator in GP, especially when using a higher selection pressure. The DMAX problem addresses the defect of the crossover operator. Our approach effectively estimates the parameters with latent annotations; therefore, the deceptiveness of the DMAX problem with strong deceptiveness can be overcome. GP required significantly more fitness evaluations than either PAGE-EM or PAGE-VB. GP cannot effectively solve the DMAX problem because of the deceptive fitness landscape of the problem. Since 0.95 is itself a positive value, GP tends to obtain an optimum containing many 0.95's. In Fig. 13, (a) is a local optimum, while (d) is the global optimum of the DMAX problem ($m = 5$, $r = 3$, and $D_P = 3$). For an individual represented by (a) to transform to (d) by a crossover operator, the individual has to traverse intermediate states represented by (b) and (c). However, the fitness values of (b) and (c) are negative, and therefore, the probability that these intermediate structures are selected in the next generation is very low. Because of these difficulties, GP performed much less successfully than DMAX with weak deceptiveness. Because structurally similar individuals have completely different fitness values in this problem, GP-type sampling did not work at all.

In the previous section, we saw that POLE performed slightly better than GP in the DMAX problem with weak deceptiveness. On the other hand, POLE required significantly fewer fitness evaluations than GP for the DMAX with strong deceptiveness. POLE also estimated the building blocks and, hence, was able to escape the deceptiveness of the DMAX problem. As mentioned in the previous experiment, POLE cannot reuse estimated building blocks because POLE is based on the prototype tree. The prototype-tree approach in general cannot estimate building blocks that are independent of position, and this is the reason that POLE performed less well than PAGE.

IX. Discussion

In this paper, we have proposed an EA named PAGE based on PCFG-LA, which allows the context freedom assumption used by conventional PCFG methods to be weakened. Two estimation methods have been used for model learning in PAGE: the EM algorithm and the VB-based algorithm. While these statistical estimation algorithms lead to higher computational costs than conventional PCFG, the performance of PAGE is markedly superior to that of the corresponding algorithm that does not use annotations (PCFG-GP).

Since PCFG-LA is a latent variable model, a sophisticated estimation is required for model learning. The original PCFG-LA adopted the EM algorithm and the present PAGE-EM used this PCFG-LA. In PAGE-EM, the number of annotations has to be specified in advance. Higher annotation sizes yield a better performance, but at the expense of significantly higher computational costs, and may also cause an over-fitting problem. Thus for PAGE-EM, trial and error are required to determine the optimum annotation size. PAGE-VB uses VB-based estimation instead of the EM algorithm. We derived the hyperparameter update formula using VB in PCFG-LA. VB learning has been shown to be able to select the best annotation size. Fig. 15 shows the average of the estimated optimum annotation sizes for each problem. In this figure, we divided each run into three phases: an early phase (the first third of generations), a middle phase (the middle third of generations), and the end phase (the last third of generations). As can been seen from Fig. 15, PAGE-VB estimated the largest annotation size for the royal-tree problem at level e. Learning data ($M \times P_s$) is the largest for the royal tree at level e. If more learning data are available, it is well-known that a more complex model can be learned. VB automatically reflects this fact by selecting a model with a maximal lower bound for the marginal log likelihood. Furthermore, the royal tree problem at level e requires more annotations because it uses more function nodes than the royal tree problem at level d. VB learning also inferred the problem complexity of each level from learning data. For the DMAX problem, DMAX with weak deceptiveness required more annotations than DMAX with strong deceptiveness. Since the maximum depth of DMAX with weak deceptiveness is larger than that of DMAX with strong deceptiveness, it follows that more annotations are required to express the tree structure of the optimum. For every problem, clearly, the estimated optimum annotation size is larger for later generations. In the later generations, promising solutions tend to converge to the optimum structures. Thus, more specialized models can be used for the later generations, and PAGE-VB makes use of this possibility. We can say from these experimental data on estimated annotation size that PAGE-VB estimated larger annotation sizes when enough learning data were available, and the structural complexity of the problem was greater. However, PAGE-VB required a larger number of fitness evaluations than PAGE-EM to obtain the optimum solutions, especially for the royal tree problem. We used the uniform prior probability distribution for the model ($P(h) \propto 1$), and this may not be well suited to the present simulations. We intend to investigate further the selection of the prior probability distribution in the near future. We can, however, say that PAGE-VB is a more effective approach than PAGE-EM in the sense that, in general, it is difficult to determine the optimum annotation size in advance. In the present study, we conducted trial-and-error runs with PAGE-EM to decide a suitable annotation size. This requires further fitness evaluations. By contrast, PAGE-VB estimated the annotation size from a single run.

In PAGE, we restricted the production rules to the form represented by (6). By using this assumption, the computational cost can be dramatically reduced. Furthermore, reduction of parameter size may avoid the overfitting problem. However, these advantages are obtained at the expense of model flexibility. Even so, we think this assumption is effective in many applications. For example, in the DMAX problem, the optimum structure is a combination of two types of building blocks and is not symmetric in general. With computer experiments, PAGE solves the DMAX problem very effectively. This result indicates that these asymmetric structures can be generated under the restricted production rules.

In Section VIII-C and VIII-D, we compared the search performances of POLE and PAGE. POLE is a state-of-the-art prototype tree-based GP-EDA employing a Bayesian network. In [11], POLE is reported to be a powerful program evolution method. The advantage of PAGE is that it can estimate building blocks which are position independent. Since building blocks of programs and functions are position independent, PAGE is considered to be more suitable for extraction of building blocks. Although POLE cannot estimate these position-independent structures, POLE is capable of estimating dependencies between any two nodes even if these two nodes are distant in the tree structure. In principle, PAGE can estimate dependencies between two distant nodes; however, many annotations are required. For such problems, we think POLE is more suitable than PAGE.

Since PAGE utilizes complicated statistical estimations such as the EM algorithm and VB, the computational cost of PAGE is greater than that of GP. In general, a complex estimation often requires more computational power. PAGE adopted the approximation that right-hand side non-terminals have the same annotation. By using this assumption, the computational complexity of PAGE is proportional to h^2. Thus, we think that the computational cost of PAGE is acceptable. Moreover, GP has recently been applied to very large datasets: biological datasets, for example. In such applications, it is the fitness evaluations that require most of the computational power. We think that PAGE will be more effective in problems where fitness evaluations are highly computationally expensive.

PAGE employed a more flexible PCFG than the PCFG methods used in prior GP-EDAs. PAGE is particularly effective at extracting building blocks from promising individuals. On the other hand, there are many GP problems where GP mutation is very effective. Thus, it can be considered that a combination of the present PAGE approach and GP mutation may prove to be a highly effective approach for improving the flexibility and power of PAGE. In this paper, we have demonstrated the effectiveness of PAGE mainly from the viewpoint of search performance. However, there may be some applications where the estimated probabilities themselves are important. For such applications, it may be that PAGE can be used not only as an optimization algorithm but also as a data-mining tool.

X. Conclusion

We have proposed a probabilistic program evolution algorithm termed PAGE. Our proposal takes advantage of latent annotations that allow the context freedom assumption for a CFG to be weakened. By applying the proposed method to four computational experiments, we have confirmed that latent annotations are highly effective for extracting the building blocks. We hope that it will be possible to apply POLE to a wide class of real-world problems, which is an intended future area of study.

Appendix I
Calculation of Posteriors in VB

In VB, the lower bound $\mathcal{F}[Q]$ of a log-likelihood $\log P(\mathcal{D})$ can be calculated using the following equations that apply Jensen's inequality:

$$\begin{aligned}
&\log P(\mathcal{D})\\
&= \log \sum_{h\in\mathcal{H}} \sum_{\mathbf{X}} \int d\boldsymbol{\beta} d\boldsymbol{\pi}\, P(\mathcal{D}, \mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\pi}, h)\\
&= \log \sum_{h\in\mathcal{H}} \sum_{\mathbf{X}} \int d\boldsymbol{\beta} d\boldsymbol{\pi}\, Q(\mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\pi}, h|\mathcal{D}) \frac{P(\mathcal{D}, \mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\pi}, h)}{Q(\mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\pi}, h|\mathcal{D})}\\
&\geq \sum_{h\in\mathcal{H}} \sum_{\mathbf{X}} \int d\boldsymbol{\beta} d\boldsymbol{\pi}\, Q(\mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\pi}, h|\mathcal{D}) \log \frac{P(\mathcal{D}, \mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\pi}, h)}{Q(\mathbf{X}, \boldsymbol{\beta}, \boldsymbol{\pi}, h|\mathcal{D})}\\
&= \sum_{h\in\mathcal{H}} \sum_{\mathbf{X}} \int d\boldsymbol{\beta} d\boldsymbol{\pi} \left\{ Q(\mathbf{X}|\mathcal{D}, h) Q(\boldsymbol{\beta}|\mathcal{D}, h) Q(\boldsymbol{\pi}|\mathcal{D}, h) \right.\\
&\qquad \left. \times\, Q(h|\mathcal{D}) \log \frac{P(\mathcal{D}, \mathbf{X}|\boldsymbol{\beta}, \boldsymbol{\pi}, h) P(\boldsymbol{\pi}|h) P(\boldsymbol{\beta}|h) P(h)}{Q(\mathbf{X}|\mathcal{D}, h) Q(\boldsymbol{\beta}|\mathcal{D}, h) Q(\boldsymbol{\pi}|\mathcal{D}, h) Q(h|\mathcal{D})} \right\} \qquad (52)\\
&\equiv \mathcal{F}[Q] \qquad (53)
\end{aligned}$$

In (52), decomposable assumption represented by (25) is used. $\mathcal{F}[Q]$ is given by

$$\begin{aligned}\mathcal{F}[Q]&\\ &=\sum_{h\in\mathcal{H}}Q(h|\mathcal{D})\log\frac{P(h)}{Q(h|\mathcal{D})}\\ &\quad+\sum_{h\in\mathcal{H}}\sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}\left\{Q(\mathbf{X}|\mathcal{D},h)Q(\boldsymbol{\beta}|\mathcal{D},h)Q(\boldsymbol{\pi}|\mathcal{D},h)\right.\\ &\quad\left.\times Q(h|\mathcal{D})\log\frac{P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)P(\boldsymbol{\pi}|h)P(\boldsymbol{\beta}|h)}{Q(\mathbf{X}|\mathcal{D},h)Q(\boldsymbol{\beta}|\mathcal{D},h)Q(\boldsymbol{\pi}|\mathcal{D},h)}\right\}\\ &=-\mathrm{KL}\left[Q(h|\mathcal{D})|P(h)\right]+\langle\mathcal{F}_h[Q]\rangle_{Q(h|\mathcal{D})}\end{aligned}\tag{54}$$

where

$$\begin{aligned}\mathcal{F}_h[Q]&\\ &\equiv\sum_{\mathbf{X}}\int d\boldsymbol{\beta}d\boldsymbol{\pi}\left\{Q(\mathbf{X}|\mathcal{D},h)Q(\boldsymbol{\beta}|\mathcal{D},h)Q(\boldsymbol{\pi}|\mathcal{D},h)\right.\\ &\qquad\left.\times\log\frac{P(\mathcal{D},\mathbf{X}|\boldsymbol{\beta},\boldsymbol{\pi},h)P(\boldsymbol{\pi}|h)P(\boldsymbol{\beta}|h)}{Q(\mathbf{X}|\mathcal{D},h)Q(\boldsymbol{\beta}|\mathcal{D},h)Q(\boldsymbol{\pi}|\mathcal{D},h)}\right\}.\end{aligned}$$

Here, $\mathrm{KL}[\cdots|\cdots]$ is the Kullback–Leibler divergence. If the annotation size h is given, the maximization of the lower bound of $\mathcal{F}_h[Q]$ is identical to the minimization of the Kullback–Leibler divergence between the true posterior distribution and the approximate posterior distribution. The approximate posterior distribution is calculated by maximizing the functional $\mathcal{F}_h[Q]$ under the normalizing constraints. The maximization of the functional with constraints can be performed using the variational method. Due to the decomposable assumption, each approximate posterior distribution can be obtained independently. For more details, see [2].

Appendix II
Parameter Update Formula for EM

$\mathcal{Q}(\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}}|\boldsymbol{\beta},\boldsymbol{\pi})$ is calculated as follows:

$$\mathcal{Q}(\overline{\boldsymbol{\beta}},\overline{\boldsymbol{\pi}}|\boldsymbol{\beta},\boldsymbol{\pi})=\mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta},\boldsymbol{\pi})+\mathcal{Q}(\overline{\boldsymbol{\pi}}|\boldsymbol{\beta},\boldsymbol{\pi})\tag{55}$$

where

$$\begin{aligned}\mathcal{Q}(\overline{\boldsymbol{\pi}}|\boldsymbol{\beta},\boldsymbol{\pi})=\sum_{i=1}^{N}\sum_{X_i}&\left\{P(X_i|T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)\right.\\ &\left.\times\sum_{x\in H}\delta(x;T_i,X_i)\log\overline{\pi}(\mathcal{S}[x])\right\}\end{aligned}\tag{56}$$

$$\begin{aligned}\mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta},\boldsymbol{\pi})=\sum_{i=1}^{N}\sum_{X_i}&\left\{P(X_i|T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)\right.\\ &\left.\times\sum_{r\in\mathcal{R}[H]}c(r;T_i,X_i)\log\overline{\beta}(r)\right\}.\end{aligned}\tag{57}$$

According to (55), $\overline{\boldsymbol{\beta}}$ and $\overline{\boldsymbol{\pi}}$ can be maximized independently. In this Appendix, we show a derivation of the update formula for $\overline{\beta}$, because $\overline{\pi}$ can be derived in a similar and easier way.

In order to maximize (57) under a constraint $\sum_\alpha\overline{\beta}(\mathcal{S}[x]\to\alpha)=1$, the Lagrange method is used as follows:

$$\frac{\partial\mathcal{L}}{\partial\overline{\beta}(\mathcal{S}[x]\to\alpha)}=0\tag{58}$$

$$\mathcal{L}=\mathcal{Q}(\overline{\boldsymbol{\beta}}|\boldsymbol{\beta},\boldsymbol{\pi})+\sum_x\xi_x\left(1-\sum_{\alpha\in\mathcal{R}_r[H]}\overline{\beta}(\mathcal{S}[x]\to\alpha)\right)\tag{59}$$

where ξ_x is a Lagrange multiplier. By solving (58), $\overline{\beta}$ is represented as

$$\begin{aligned}&\overline{\beta}(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])\\ &\quad\propto\sum_{i=1}^{N}\sum_{X_i}\big\{P(X_i|T_i;\beta,\pi,h)\\ &\quad\times c(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y];T_i,X_i)\big\}.\end{aligned}\tag{60}$$

Let us define $c(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y];T_i)$ as follows:

$$\begin{aligned}&c(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y];T_i)\\ &\equiv\sum_{X_i}P(X_i|T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)c(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y];T_i,X_i)\\ &=\sum_{X_i}\frac{P(T_i,X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}c(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y];T_i,X_i).\end{aligned}\tag{61}$$

Differentiating (2) with $\beta(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])$, we obtain

$$\begin{aligned}&\frac{\partial P(T_i,X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{\partial\beta(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])}\\ &\quad=\frac{c(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y];T_i,X_i)}{\beta(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])}P(T_i,X_i;\boldsymbol{\beta},\boldsymbol{\pi},h).\end{aligned}\tag{62}$$

Merging (61) and (62), we obtain (63)

$$\begin{aligned}&c(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y];T_i)\\ &\quad=\frac{\beta(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])}{P(T_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}\\ &\qquad\times\sum_{X_i}\frac{\partial P(T_i,X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{\partial\beta(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])}.\end{aligned}\tag{63}$$

The partial derivative part in (63) is calculated as

$$\begin{aligned}&\sum_{X_i}\frac{\partial P(T_i,X_i;\boldsymbol{\beta},\boldsymbol{\pi},h)}{\partial\beta(\mathcal{S}[x]\to g\,\mathcal{S}[y]\ldots\mathcal{S}[y])}\\ &\quad=\sum_{j\in\mathrm{cover}(g,T_i)}f^j_{T_i}(x;\boldsymbol{\beta},\boldsymbol{\pi},h)\cdot\prod_{k\in\mathrm{ch}(j,T_i)}b^k_{T_i}(y;\boldsymbol{\beta},\boldsymbol{\pi},h).\end{aligned}\tag{64}$$

By using (64), the update formula for $\overline{\beta}$ can be obtained. A similar consideration for $\overline{\pi}$ yeilds a update formula of $\overline{\pi}$.

Appendix III
Calculation of Sum Over Latent Annotations

In Appendix III, we show how to calculate

$$\widetilde{c}(\mathcal{S}[x]\to\alpha;T_i)\equiv\sum_{X_i}Q(X_i|T_i,h)\cdot c(\mathcal{S}[x]\to\alpha;T_i,X_i)\tag{65}$$

in (30). It is intractable to directly evaluate this equation because the computational cost increases exponentially with the number of latent annotations. Because $Q(X_i|T_i, h)$ is just $P(X_i|T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, h)$ with $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ replaced by $\widetilde{\boldsymbol{\beta}}$ and $\widetilde{\boldsymbol{\pi}}$, (65) can be evaluated in the same way as (61). By substituting $\widetilde{\boldsymbol{\beta}}$ and $\widetilde{\boldsymbol{\pi}}$, we finally obtain (66)

$$\begin{aligned}\widetilde{c}(\mathcal{S}[x] \to g\,\mathcal{S}[y] \ldots \mathcal{S}[y]; T_i) &= \frac{\widetilde{\beta}(\mathcal{S}[x] \to g\,\mathcal{S}[y] \ldots \mathcal{S}[y])}{\mathcal{Z}(T_i)} \\ &\times \sum_{j \in \mathrm{cover}(g, T_i)} f_{T_i}^{j}(x; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h) \prod_{k \in \mathrm{ch}(j, T_i)} b_{T_i}^{k}(y; \widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\pi}}, h). \end{aligned} \tag{66}$$

Here, $\mathcal{Z}(T_i)$ is the normalizing constant calculated using (38), which is represented by the backward probability. Thus, (66) can be evaluated using forward–backward probabilities, which are tractable.

We may calculate $\sum_{X_i} Q(X_i|T_i, h) \cdot \delta(x; T_i, X_i)$ in (40) in a similar and easier way.

Appendix IV
KL Divergence Calculation

The lower bound $\mathcal{F}_h^*[Q]$ can be expressed as

$$\begin{aligned}\mathcal{F}_h^*[Q] = \sum_{i=1}^{N} \log \mathcal{Z}(T_i) &- \mathrm{KL}\left[Q^*(\boldsymbol{\beta}|\mathcal{D}, h)|P(\boldsymbol{\beta}|h)\right] \\ &- \mathrm{KL}\left[Q^*(\boldsymbol{\pi}|\mathcal{D}, h)|P(\boldsymbol{\pi}|h)\right].\end{aligned}$$

The last two terms are calculated as follows:

$$\begin{aligned}&\mathrm{KL}\left[Q^*(\boldsymbol{\beta}|\mathcal{D}, h)|P(\boldsymbol{\beta}|h)\right] \\ &= \sum_{x \in H} \left\{ \sum_{\alpha \in \mathcal{R}_r[H]} \left(\log \Gamma(b_{\mathcal{S}[x]\to\alpha}) - \log \Gamma\left(\breve{b}^*_{\mathcal{S}[x]\to\alpha}\right)\right) \right. \\ &\quad + \log \Gamma\left(\sum_{\alpha \in \mathcal{R}_r[H]} \breve{b}^*_{\mathcal{S}[x]\to\alpha} \right) \\ &\quad \left. - \log \Gamma\left(\sum_{\alpha \in \mathcal{R}_r[H]} b_{\mathcal{S}[x]\to\alpha} \right) \right\} \\ &\quad + \sum_{x \in H} \sum_{\alpha \in \mathcal{R}_r[H]} \left(\breve{b}^*_{\mathcal{S}[x]\to\alpha} - b_{\mathcal{S}[x]\to\alpha} \right) \\ &\quad \times \left\{ \psi\left(\breve{b}^*_{\mathcal{S}[x]\to\alpha}\right) - \psi\left(\sum_{\zeta \in \mathcal{R}_r[H]} \breve{b}^*_{\mathcal{S}[x]\to\zeta} \right) \right\}\end{aligned}$$

$$\begin{aligned}&\mathrm{KL}\left[Q^*(\boldsymbol{\pi}|\mathcal{D}, h)|P(\boldsymbol{\pi}|h)\right] \\ &= \sum_{x \in H} \left(\log \Gamma(a_{\mathcal{S}[x]}) - \log \Gamma\left(\breve{a}^*_{\mathcal{S}[x]}\right)\right) \\ &\quad + \log \Gamma\left(\sum_{x \in H} \breve{a}^*_{\mathcal{S}[x]} \right) - \log \Gamma\left(\sum_{x \in H} a_{\mathcal{S}[x]} \right) \\ &\quad + \sum_{x \in H} \left(\breve{a}^*_{\mathcal{S}[x]} - a_{\mathcal{S}[x]} \right) \left\{ \psi(\breve{a}^*_{\mathcal{S}[x]}) - \psi\left(\sum_{y \in H} \breve{a}^*_{\mathcal{S}[y]} \right) \right\}.\end{aligned}$$

References

[1] H. A. Abbass, X. Hoai, and R. I. Mckay, "AntTAG: A new method to compose computer programs using colonies of ants," in *Proc. IEEE Congr. Evol. Comput.*, Honolulu, HI, 2002, pp. 1654–1659.

[2] H. Attias, "Inferring parameters and structure of latent variable models by variational bayes," in *Proc. 15th Conf. Uncertainty Artificial Intell.*, Stockholm, Sweden: Morgan Kaufmann, 1999, pp. 21–30.

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

[4] M. J. Beal, "Variational algorithms for approximate bayesian inference," Ph.D. dissertation, Gatsby Comput. Neurosci. Unit, Univ. College London, London, U.K., 2003.

[5] P. A. N. Bosman and E. D. de Jong, "Grammar transformations in an EDA for genetic programming," Inst. Inform. Comput. Sci., Utrecht University, Utrecht, The Netherlands, Tech. Rep. UU-CS-2004-047, 2004.

[6] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statistical Soc., Series B*, vol. 39, no. 1, 1977.

[7] C. Gathercole and P. Ross, "An adverse interaction between crossover and restricted tree depth in genetic programming," presented at 1st Annu. Conf. Genetic Program., Stanford Univ., Stanford, CA, Jul. 28–31, 1996.

[8] D. E. Goldberg, D. Deb, and H. Kargupta, "Rapid, accurate optimization of difficult problems using fast messy genetic algorithms," presented at 5th Int. Conf. Genetic Algorithms, San Mateo, CA, 1993.

[9] G. Harik, "Linkage learning via probabilistic modeling in the ECGA," Algorithms Lab., Dept. General Eng., Urbana, IL, Tech. Rep. 99010, Jan. 1999.

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

[11] Y. Hasegawa and H. Iba, "A Bayesian network approach for program generation," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 750–764, Dec. 2008.

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

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

[14] K. Kurihara and T. Sato, "An application of the variational Bayesian approach to probabilistic context-free grammars," in *Proc. Workshop Beyond Shallow Analyses*. Available: http://satowww.cs.titech.ac.jp/en/publication/

[15] W. B. Langdon and R. Poli, "An analysis of the MAX problem in genetic programming," in *Proc. 2nd Annu. Conf. Genetic Programm.* San Mateo, CA: Morgan Kaufmann, 1997, pp. 222–230.

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

[17] M. Looks, "Learning computer programs with the Bayesian optimization algorithm," M.S. thesis, Dept. Comput. Sci. Eng., Washington Univ. Sever Inst. Technol., St. Louis, MO, 2005.

[18] M. Looks, "Scalable estimation-of-distribution program evolution," in *Proc. 9th Annu. Conf. Genetic Evol. Comput.*, New York: ACM, pp. 539–546.

[19] D. J. C. MacKay. (1997). *Ensemble Learning for Hidden Markov Models* [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/abstracts/ensemblePaper.html

[20] T. Matsuzaki, Y. Miyao, and J. Tsujii, "Probabilistic CFG with latent annotations," in *Proc. 43rd Meeting Assoc. Comput. Linguistics*. Detroit, MI: Morgan Kaufmann, 2005, pp. 75–82.

[21] M. Mitchell, S. Forrest, and J. H. Holland, "The royal road for genetic algorithms: Fitness landscapes and GA performance," presented at 1st Eur. Conf. Artifcial Life, Toward Prac. Autonomous Syst., Paris, France, 1992.

[22] P. Nordin, "A compiling genetic programming system that directly manipulates the machine code," in *Advances Genetic Programming*, Cambridge, MA: MIT Press, 1994, ch. 14, pp. 311–331.

[23] M. Pelikan and D. E. Goldberg, "Escaping hierarchical traps with competent genetic algorithms," in *Proc. Conf. Genetic Evol. Comput.*, New York: ACM, 2001, pp. 511–518.

[24] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. Genetic Evol. Comput Conf.*, vol. 1. Orlando, FL, pp. 525–532.

[25] R. Poli and N. F. McPhee, "A linear estimation-of-distribution GP system," in *Proc. 11th Eur. Conf. Genetic Programmins EuroGP*, 2008, pp. 206–217.

[26] W. F. Punch, "How effective are multiple populations in genetic programming," in *Proc. 3rd Annu. Conf. Genetic Programming*, Madison, WI: Univ. Wisconsin, Jul. 1998, pp. 308–313.

[27] A. Ratle and M. Sebag, "Avoiding the bloat with probabilistic grammar-guided genetic programming," in *Proc. Artificial Evolution 5th Int. Conf., Evolution Artificielle*, LNCS vol. 2310. Creusot, France: Springer-Verlag, Oct. 2001, pp. 255–266.

[28] E. N. Regolin and A. T. R. Pozo, "Bayesian automatic programming," in *Proc. 8th Eur. Conf. Genetic Programming*, LNCS vol. 3447. Lausanne, Switzerland: Springer-Verlag, Mar.–Apr. 2005, pp. 38–49.

[29] R. P. Salustowicz and J. Schmidhuber, "Probabilistic incremental program evolution," *Evol. Comput.*, vol. 5, no. 2, pp. 123–141, 1997.

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

[31] 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. Congr. Evol. Comput.*, Canberra, Australia: IEEE Press, Dec. 2003, pp. 1639–1646.

[32] Y. Shan, R. I. McKay, R. Baxter, H. Abbass, D. Essam, and N. X. Hoai, "Grammar modelbased program evolution," in *Proc. IEEE Congr. Evol. Comput.*, Portland, OR: IEEE Press, Jun. 2004, pp. 478–485.

[33] Y. Shan, R. I. McKay, D. Essam, and H. A. Abbass, "A survey of probabilistic model building genetic programming," in *Scalable Optimization Via Probabilistic Modeling*, New York: Springer-Verlag, 2006, ch. 6, pp. 121–160.

[34] A. Stolcke, "Bayesian learning of probabilistic language models," Ph.D. dissertation, Univ. California, Berkeley, CA, 1994.

[35] I. Tanev, "Implications of incorporating learning probabilistic context-sensitive grammar in genetic programming on evolvability of adaptive locomotion gaits of snakebot," in *Proc. Workshop GECCO*, Seattle, WA, Jun. 2004.

[36] I. Tanev, "Incorporating learning probabilistic context-sensitive grammar in genetic programming for efficient evolution and adaptation of Snakebot," in *Proc. EuroGP*, Lausanne, Switzerland: Springer-Verlag, 2005, pp. 155–166.

[37] P. A. Whigham, "Grammatically-based genetic programming," in *Proc. Workshop Genetic Programming: From Theory to Real-World Applicat.*, Tahoe City, CA, 1995, pp. 44–41.

[38] P. A. Whigham and D. O. C. Science, "Inductive bias and genetic programming," presented at 1st Int. Conf. Genetic Algorithms Eng. Syst.: Innovations Applicat., Sheffield, U.K., 1995.

[39] M. Wineberg and F. Oppacher, "A representation scheme to perform program induction in a canonical genetic algorithm, in *Parallel Problem Solving From Nature III*, LNCS vol. 866. Jerusalem, Israel: Springer-Verlag, Oct. 1994, pp. 292–301.

[40] K. Yanai and H. Iba, "Estimation of distribution programming based on Bayesian network," in *Proc. Congr. Evol. Comput.*, Canberra, Australia: IEEE Press, Dec. 2003, pp. 1618–1625.

[41] K. Yanai and H. Iba, "Probabilistic distribution models for EDA-based GP," in *Proc. Conf. Genetic Evol. Comput.*, vol. 2. Washington, DC, ACM, Jun. 2005, pp. 1775–1776.

Yoshihiko Hasegawa received the B.Sc. degree in physics, the M.Eng. degree in electric engineering, and the Ph.D degree in information science from the University of Tokyo, Tokyo, Japan in 2003, 2005, and 2008, respectively.

Currently, he is an Assistant Professor with the Department of Computational Biology, Graduate School of Frontier Sciences, University of Tokyo. His interests include estimation of distribution algorithms, bioinformatics, and information science based on statistical physics.

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

Since 1990, he was with the Electro Technical Laboratory, University of Tokyo, before joining the Department of Electronic Engineering in 1998, where he was an Associate Professor from 1998 to 2004. He is currently a Professor with the Graduate School of Engineering. His research interests include evolutionary computation, genetic programming, bioinformatics, foundations of artificial intelligence, financial engineering machine learning, robotics, and vision.

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