We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 3

Programming with Annotated Grammar Estimation

Yoshihiko Hasegawa

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51662

1. Introduction

Evolutionary algorithms (EAs) mimic natural evolution to solve optimization problems. Because EAs do not require detailed assumptions, they can be applied to many real-world problems. In EAs, solution candidates are evolved using genetic operators such as crossover and mutation which are analogs to natural evolution. In recent years, EAs have been considered from the viewpoint of distribution estimation, with estimation of distribution algorithms (EDAs) attracting much attention ([14]). Although genetic operators in EAs are inspired by natural evolution, EAs can also be considered as algorithms that sample solution candidates from distributions of promising solutions. Since these distributions are generally unknown, approximation schemes are applied to perform the sampling. Genetic algorithms (GAs) and genetic programmings (GPs) approximate the sampling by randomly changing the promising solutions via genetic operators (mutation and crossover). In contrast, EDAs assume that the distributions of promising solutions can be expressed by parametric models, and they perform model learning and sampling from the learnt models repeatedly. Although GA-type sampling (mutation or crossover) is easy to perform, it has the disadvantage that GA-type sampling is valid only for the case where two structurally similar individuals have similar fitness values (e.g. the one-max problem). GA and GP have shown poor search performance in deceptive problems ([6]) where the condition above is not satisfied. However, EDAs have been reported to show much better search performance for some problems that GA and GP do not handle well. As in GAs, EDAs usually employ fixed length linear arrays to represent solution candidates (these EDAs are referred to as GA-EDAs in the present chapter). This decade, EDAs have been extended so as to handle programs and functions having tree structures (we refer to these as GP-EDAs in the present chapter). Since tree structures have different node number, the model learning is much more difficult than that of GA-EDAs. From the viewpoint of modeling types, GP-EDAs can be broadly classified into two groups: probabilistic proto-type tree (PPT) based methods and probabilistic context-free grammar (PCFG) based methods. PPT-based methods employ techniques devised in GA-EDAs by transforming variable length tree structures into fixed length linear arrays. PCFG-based methods employ

PCFG to model tree structures. PCFG-based methods are more advantageous than PPT-based methods in the sense that PCFG-based methods can estimate position-independent building blocks.

The conventional PCFG adopts the context freedom assumption that the probabilities of production rules do not depend on their contexts, namely parent or sibling nodes. Although the context freedom assumption makes parameter estimation easier, it cannot in principle consider interaction among nodes. In general, programs and functions have dependencies among nodes, and as a consequence, the conventional PCFG is not suitable as a baseline model of GP-EDAs. In the field of natural language processing (NLP), many approaches have been proposed in order to weaken the content freedom assumption of PCFG. For instance, the vertical Markovization annotates symbols with their ancestor symbols and has been adopted as a baseline grammar of vectorial stochastic grammar based GP (vectorial SG-GP) or grammar transformation in an EDA (GT-EDA) ([4]) (see Section 2). Matsuzaki *et al.* ([17]) proposed the PCFG with latent annotations (PCFG-LA), which assumes that all annotations are latent and the annotations are estimated from learning data. Because the latent annotation models are much richer than fixed annotation models, it is expected that GP-EDAs using PCFG-LA may more precisely grasp the interactions among nodes than other fixed annotation based GP-EDAs. In GA-EDAs, EDAs with Bayesian networks or Markov networks exhibited better search performance than simpler models such as a univariate model. In a similar way, it is generally expected that GP-EDAs using PCFG-LA are more powerful than GP-EDAs with PCFG with heuristics-based annotations because the model flexibility of PCFG-LA is much richer. We have proposed a GP-EDA named programming with annotated grammar estimation (PAGE) which adopts PCFG-LA as a baseline grammar ([9, 12]). In Section 4 of the present chapter, we explain the details of PAGE, including the parameter update formula.

As explained above, EDAs model promising solutions with parametric distributions. For the case in multimodal problems, it is not sufficient to express promising solutions with only one model, because dependencies for each optimal solution are different in general. When considering tree structures, this problem arises even in unimodal optimization problems due to diversity of tree expression. These problems can be tackled by considering global contexts in each individual, which represents which optima (e.g. multiple solutions in multimodal problems) it derives from. Consequently, we have proposed the PCFG-LA mixture model (PCFG-LAMM) which extends PCFG-LA into a mixture model, and have also proposed a new GP-EDA named unsupervised PAGE (UPAGE) which employs PCFG-LAMM as a baseline grammar ([11]). By using PCFG-LAMM, not only local dependencies but also global contexts behind individuals can be taken into account.

The main objectives of proposed algorithms may be summarized as follows:

1. PAGE employs PCFG-LA to consider local dependencies among nodes.
2. UPAGE employs PCFG-LAMM to take into account global contexts behind individuals in addition to the local dependencies.

This chapter is structured as follows: Following a section on related work, we briefly introduce the basics of PCFG. We explain PAGE in Section. 4, where details of PCFG-LA, forward–backward probabilities and a parameter update formula are provided. In Section 5, we propose UPAGE, which is a mixture model extension of PAGE. We describe PCFG-LAMM and also derive a parameter update formula for UPAGE. We compare the performance of

UPAGE and PAGE using three benchmark tests selected for experiments. We discuss the results obtained in these experiments in Section 6. Finally, we conclude the present chapter in Section 7.

2. Related work

Many GP-EDAs have been proposed, and these methods can be broadly classified into two groups: (i) PPT based methods and (ii) grammar model based methods.

Methods of type (i) employ techniques developed in GA-EDAs. This type of algorithm converts tree structures into the fixed-length chromosomes used in GA and applies probabilistic models of GA-EDAs. Probabilistic incremental program evolution (PIPE) ([25]) is a univariate model, which can be considered to be a combination of population-based incremental learning (PBIL) ([3]) and GP. Because tree structures have explicit edges between parent and children nodes, estimation of distribution programming (EDP) ([37, 38]) considers the parent–children relationships in the tree structures. Extended compact GP (ECGP) ([26]) is an extension of the extended compact GA (ECGA) ([7]) to GP and ECGP can take into account the interactions among nodes. ECGP infers the group of marginal distribution using the minimum description length (MDL) principle. BOA programming (BOAP) ([15]) uses Bayesian networks for grasping dependencies among nodes and is a GP extension of the Bayesian optimization algorithm (BOA) ([20]). Program optimization with linkage estimation (POLE) ([8, 10]) estimates the interactions among nodes by estimating the Bayesian network. POLE uses a special chromosome called an *expanded parse tree* ([36]) to convert GP programs into linear arrays, and several extended algorithms of POLE have been proposed ([27, 39]). Meta-optimizing semantic evolutionary search (MOSES) ([16]) extends the hierarchical Bayesian optimization algorithm (hBOA) ([19]) to program evolution.

Methods of type (ii) are based on Whigham's grammar-guided genetic programming (GGGP) ([33]). GGGP expresses individuals using derivation trees (see Section 3), which is in contrast with the conventional GP. Whigham indicated the connection between PCFG and GP ([35]), and actually, the probability table learning in GGGP can be viewed as an EDA with local search. Stochastic grammar based GP (SG-GP) ([23]) applied the concept of PBIL to GGGP. The authors of SG-GP 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) ([28]) takes into account the positions (arguments) and depths of symbols. Unlike SG-GP and PEEL, which employ predefined grammars, grammar model based program evolution (GMPE) ([29]) learns not only parameters but also the grammar itself from promising solutions. GMPE starts from specialized production rules which exclusively generate learning data and merges non-terminals to yield more general production rules using the MDL principle. Grammar transformation in an EDA (GT-EDA) ([4]) extracts good subroutines using the MDL principle. GT-EDA starts from general rules and expands non-terminals to yield more specialized production rules. Although the concept of GT-EDA is similar to that of GMPE, the learning procedure is opposite to GMPE [specialized to general (GMPE) versus general to specialized (GT-EDA)]. Tanev proposed GP based on a probabilistic context sensitive grammar ([31, 32]). He used sibling nodes and a parent node as context information, and production rule probabilities are expressed by conditional probabilities of these context information. Bayesian automatic programming (BAP) ([24]) uses a Bayesian network to consider relations among production rules in PCFG.

There are other GP-EDAs not belonging to either of the groups presented above. N-gram GP ([21]) is based on the linear GP ([18]), which is the assembly language of a register-based CPU, and learns the sub-sequences using an N-gram model. The N-gram model is very popular in NLP which considers N consecutive sub-sequences for calculating the probabilities of symbols. AntTAG ([1]) also shares similar concepts with GP-EDAs, although AntTAG does not employ a statistical inference method for probability learning; instead, AntTAG employs the ant colony optimization method (ACO), where the pheromone matrix in ACO can be interpreted as a probability distribution.

3. Basics of PCFG

In this section, we explain basic concepts of PCFG.

The context-free grammar (CFG) G is defined by four variables $G = \{\mathcal{N}, \mathcal{T}, \mathcal{R}, \mathcal{B}\}$, where the meanings of these variables are listed below.

- $\mathcal{N}$: Finite set of non-terminal symbols
- $\mathcal{T}$: Finite set of terminal symbols
- $\mathcal{R}$: Finite set of production rules
- $\mathcal{B}$: Start symbol

It is important to note that the terms "non-terminal" and "terminal" in CFG are different from those in GP (for example in symbolic regression problems, not only variables x, y but also $\sin, +$ are treated as terminals in CFG). In CFG, sentences are generated by applying production rules to non-terminal symbols, which are generally given by

$$A \to \alpha \quad (A \in \mathcal{N}, \alpha \in (\mathcal{N} \cup \mathcal{T})^*). \tag{1}$$

In Equation 1, $(\mathcal{N} \cup \mathcal{T})^*$ represents a set of possible elements composed of $(\mathcal{N} \cup \mathcal{T})$. By applying production rules to the start symbol $\mathcal{B}$, grammar G generates sentences. A language generated by grammar G is represented by $L(G)$. If $W \in L(G)$, then $W \in \mathcal{T}^*$.

By applying production rules, non-terminal A is replaced by another symbol. For instance, application of the production rule represented by Equation 1 to $\alpha_1 A \alpha_2 (\alpha_1, \alpha_2 \in (\mathcal{N} \cup \mathcal{T})^*, A \in \mathcal{N})$ yields $\alpha_1 \alpha \alpha_2$. In this case, it is said that "$\alpha_1 A \alpha_2$ derived $\alpha_1 \alpha \alpha_2$", and this process is represented as follows:

$$\alpha_1 A \alpha_2 \underset{G}{\Rightarrow} \alpha_1 \alpha \alpha_2.$$

Furthermore, if we have the following consecutive applications

$$\alpha_1 \underset{G}{\Rightarrow} \alpha_2 \cdots \underset{G}{\Rightarrow} \alpha_n (\alpha_i \in (\mathcal{N} \cup \mathcal{T})^*),$$

α_n is derived from α_1 and is described by $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_n$. This derivation process can be represented by a tree structure, which is known as a *derivation tree*. Derivation trees of grammar G are defined as follows.

1. Node is an element of $(\mathcal{N} \cup \mathcal{T})$
2. Root is $\mathcal{B}$

3. Branch node is an element of $\mathcal{N}$
4. If children of $A \in \mathcal{N}$ are $\alpha_1\alpha_2\cdots\alpha_k$ ($\alpha_i \in (\mathcal{N} \cup \mathcal{T})$) from left, production rule $A \to \alpha_1\alpha_2\cdots\alpha_k$ is an element of $\mathcal{R}$

We next explain CFG with an example. We now consider a univariate function $f(x)$ composed of sin, cos, exp, log and arithmetic operators ($+$, $-$, $\times$ and $\div$). A grammar G_{reg} can be

$$\begin{aligned}\mathcal{B} &= \{\langle expr\rangle\},\\ \mathcal{N} &= \{\langle expr\rangle, \langle op2\rangle, \langle op1\rangle, \langle var\rangle, \langle const\rangle\},\\ \mathcal{T} &= \{+, -, \times, \div, \sin, \cos, \exp, \log, x, C\}.\end{aligned}$$

We define the following production rules.

#	Production rule
0	$\langle expr\rangle \to \langle op2\rangle\ \langle expr\rangle\ \langle expr\rangle$
1	$\langle expr\rangle \to \langle op1\rangle\ \langle expr\rangle$
2	$\langle expr\rangle \to \langle var\rangle$
3	$\langle expr\rangle \to \langle const\rangle$
4	$\langle op2\rangle \to +$
5	$\langle op2\rangle \to -$
6	$\langle op2\rangle \to \times$
7	$\langle op2\rangle \to \div$
8	$\langle op1\rangle \to \sin$
9	$\langle op1\rangle \to \cos$
10	$\langle op1\rangle \to \exp$
11	$\langle op1\rangle \to \log$
12	$\langle var\rangle \to x$
13	$\langle const\rangle \to C$ (constant)

G_{reg} derives univariate functions by applying the production rules. Suppose we have the following derivation:

$$\begin{aligned}\langle expr\rangle &\to \langle op2\rangle\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ \langle op2\rangle\ \langle expr\rangle\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ +\ \langle expr\rangle\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ +\ \langle op1\rangle\ \langle expr\rangle\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ +\ \log\ \langle expr\rangle\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ +\ \log\ \langle var\rangle\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ +\ \log\ x\ \langle expr\rangle\ \langle expr\rangle\\ &\to +\ +\ \log\ x\ \langle var\rangle\ \langle expr\rangle\\ &\to +\ +\ \log\ x\ x\ \langle expr\rangle\\ &\to +\ +\ \log\ x\ x\ \langle const\rangle\\ &\to +\ +\ \log\ x\ x\ C.\end{aligned}$$

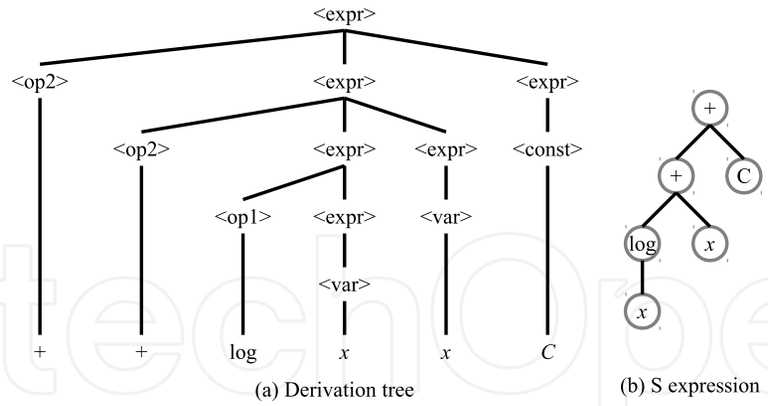


Figure 1. (a) Derivation tree for $\log x + x + C$ and (b) its corresponding S-expression in GP.

In this case, the derived function is

$$f(x) = \log x + x + C,$$

and its derivation process is represented by the derivation tree in Figure 1(a).

Although functions and programs are represented with standard tree representations (S-expression) in the conventional GP (Figure 1(b)), derivation trees can express the same functions and programs. Consequently, derivation trees can be used in program evolution, and GGGP ([33, 34]) adopted derivation trees for its chromosome.

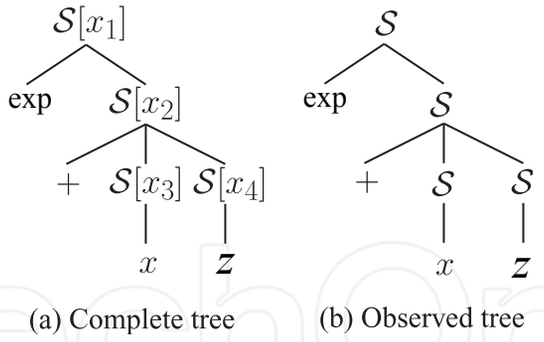
We next proceed to PCFG, which extends CFG by adding probabilities to each production rule. For example, the likelihood (probability) of the derivation tree in Fig. 1(a) is

$$\begin{aligned} P(W,T) = {} & \pi(\langle expr\rangle)\beta(\langle expr\rangle \to \langle op2\rangle\ \langle expr\rangle\ \langle expr\rangle)^2\beta(\langle op2\rangle \to +)^2 \\ & \times \beta(\langle expr\rangle \to \langle op1\rangle\ \langle expr\rangle)\beta(\langle op1\rangle \to \log) \\ & \times \beta(\langle expr\rangle \to \langle const\rangle)\beta(\langle expr\rangle \to \langle var\rangle)^2\beta(\langle const\rangle \to C)\beta(\langle var\rangle \to x)^2, \end{aligned}$$

where $W \in \mathcal{T}^*$ is a sentence (i.e. W corresponds to $\log x + x + C$ in G_{reg}), T is a derivation tree, $\pi(\langle expr\rangle)$ is the probability of $\langle expr\rangle$ and $\beta(A \to \alpha)$ is the probability of a production rule $A \to \alpha$. Furthermore, the probability $P(W)$ of sentence W is given by calculating the marginal probability in terms of $T \in \Phi(W)$:

$$P(W) = \sum_{T \in \Phi(W)} P(W,T), \tag{2}$$

where $\Phi(W)$ is the set of all possible derivation trees which derive W. In NLP, inference of the production rule parameters $\beta(A \to \alpha)$ is carried out with learning data $\mathbf{W} = \{W_1, W_2, \cdots\}$, which is a set of sentences. The learning data does not have information about derivation processes. Because there are many possible derivations $\Phi(W)$ for large sentences, directly calculating $P(W)$ with marginalization in terms of $\Phi(W)$ (Equation 2) is computationally intractable. Consequently, a computationally efficient method called the *inside–outside algorithm* is used to estimate the parameters. The inside–outside algorithm takes advantage of dynamic programming to reduce the computational cost. However, in contrast to the case of NLP, the derivation trees are observed in GP-EDAs, and the parameter estimation of production rules in GP-EDAs with PCFG is very easy. However, when using



(a) Complete tree (b) Observed tree

Figure 2. (a) Complete tree with annotations and (b) its observed tree.

more complicated grammars such as PCFG-LA, more advanced estimation methods (i.e. the expectation maximization (EM) algorithm ([5])) have to be used even when derivation trees are given.

4. PAGE

Our proposed algorithm PAGE is based on PCFG-LA. In PCFG-LA, latent annotations are estimated from promising solutions using the EM algorithm, and PCFG-LA takes advantage of forward–backward probabilities for computationally efficient estimation. In this section, we describe the details of PCFG-LA, forward-backward probabilities and a parameter update formula derived from the EM algorithm.

4.1. PCFG-LA

Although the PCFG-LA used in PAGE has been developed specifically for the present application, it is essentially identical to the conventional PCFG-LA. In this section, we describe the specialized version of PCFG-LA. For further details on PCFG-LA, the reader may refer to Ref. ([17]).

PCFG-LA assumes that 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 latent), and H is a set of annotations (in this paper, we take $H = \{0, 1, 2, 3, \cdots, h-1\}$, where h is the annotation size). Fig. 2 shows an example of a tree with annotations (a), and the corresponding observed tree (b). The likelihood of an annotated tree (complete data) is given by

$$P(T_i, X_i; \boldsymbol{\beta}, \boldsymbol{\pi}) = \prod_{x \in H} \pi(\mathcal{S}[x])^{\delta(x; T_i, X_i)} \prod_{r \in \mathcal{R}[H]} \beta(r)^{c(r; T_i, X_i)}, \tag{3}$$

where T_i denotes the ith derivation tree; X_i is the set of latent annotations of T_i represented by $X_i = \{x_i^1, x_i^2, \cdots\}$ (x_i^j is the jth annotation of T_i); $\pi(\mathcal{S}[x])$ is the 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; $c(\mathcal{S}[x] \to \alpha; T_i, X_i)$ is the number of occurrences of rule $\mathcal{S}[x] \to \alpha$ in the complete tree T_i, X_i; h is the annotation size that is specified in advance as a parameter; $\boldsymbol{\beta} = \{\beta(\mathcal{S}[x] \to \alpha) | \mathcal{S}[x] \to \alpha \in \mathcal{R}[H]\}$; and $\boldsymbol{\pi} = \{\pi(\mathcal{S}[x]) | x \in H\}$. The set of annotated rules $\mathcal{R}[H]$ is given in Equation 8. We summarized variables in Appendix B.

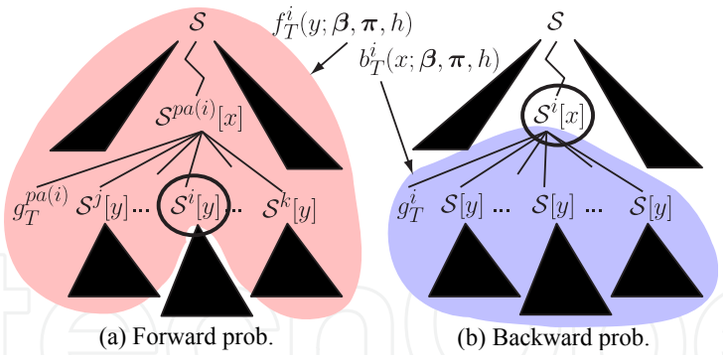


Figure 3. (a) Forward and (b) backward probabilities. The superscripts denote the indices of non-terminals (i in $\mathcal{S}^i[y]$, for example).

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

$$P(T_i;\boldsymbol{\beta},\boldsymbol{\pi}) = \sum_{X_i} P(T_i, X_i;\boldsymbol{\beta},\boldsymbol{\pi}). \tag{4}$$

PCFG-LA estimates $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ using the EM algorithm. Before explaining the estimation procedure, we should note the form of production rules. In PAGE, production rules are not Chomsky normal form (CNF), as is assumed in the original PCFG-LA, because of the understandability of GP programs. Any function which can be handled with traditional GP can be represented by

$$\mathcal{S} \rightarrow g\,\mathcal{S}\ldots\mathcal{S}, \tag{5}$$

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; see Section 3). A terminal symbol g in CFG is a function node ($+, -, \sin, \cos \in \mathfrak{F}$) or a terminal ($v, w \in \mathfrak{T}$) in GP ($\mathfrak{F}$ and $\mathfrak{T}$ denote set of GP functions and terminals, respectively). Annotated production rules are

$$\mathcal{S}[x] \rightarrow g\,\mathcal{S}[z_1]\ldots\mathcal{S}[z_{a_{\max}}], \tag{6}$$

where $x, z_m \in H$ and $a_{\max}$ is the arity of g in GP. If g has $a_{\max}$ arity, the number of parameters for the production rule $\mathcal{S} \rightarrow g\,\mathcal{S}\ldots\mathcal{S}$ with annotations is $h^{a_{\max}+1}$, which increases exponentially as the arity number increases. In order to reduce the number of parameters, we assume that all the right-hand side non-terminal symbols have the same annotation, that is

$$\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]\,\mathcal{S}[y]\ldots\mathcal{S}[y]. \tag{7}$$

With this assumption, the number of parameters can be reduced to h^2, which is tractable. Let $\mathcal{R}[H]$ be the set of annotated rules expressed by Equation 8. $\mathcal{R}[H]$ is defined by

$$\mathcal{R}[H] = \{\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]\,\mathcal{S}[y]\ldots\mathcal{S}[y] | x, y, \in H, g \in \mathcal{T}\}. \tag{8}$$

4.2. Forward–backward probability

We explain forward and backward probabilities for PCFG-LA in this section. PCFG-LA ([17]) adopted forward and backward probabilities to apply the EM algorithm ([5]). The backward probability $b_T^i(x;\boldsymbol{\beta},\boldsymbol{\pi})$ represents the probability that the tree beneath the ith non-terminal $\mathcal{S}[x]$ is generated ($\boldsymbol{\beta}$ and $\boldsymbol{\pi}$ are parameters, Fig. 3 (b)), and the forward probability $f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi})$

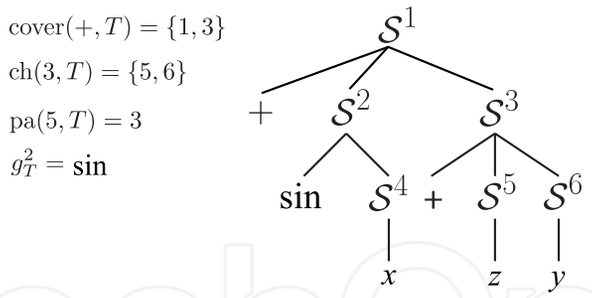


Figure 4. Example of a derivation tree and values of the specific functions. The superscripts denote the indices of non-terminals.

represents the probability that the tree above the ith non-terminal $\mathcal{S}[y]$ is generated (Fig. 3 (a)). Forward and backward probabilities can be recursively calculated as follows:

$$b_T^i(x;\boldsymbol{\beta},\boldsymbol{\pi}) = \sum_{y\in H} \beta(\mathcal{S}[x] \to g_T^i\, \mathcal{S}[y]...\mathcal{S}[y]) \prod_{j\in \mathrm{ch}(i,T)} b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi}), \tag{9}$$

$$\begin{aligned} f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi}) = &\sum_{x\in H} f_T^{\mathrm{pa}(i,T)}(x;\boldsymbol{\beta},\boldsymbol{\pi})\beta(\mathcal{S}[x] \to g_T^{\mathrm{pa}(i,T)}\, \mathcal{S}[y]...\mathcal{S}[y]) \\ &\times \prod_{j\in \mathrm{ch}(\mathrm{pa}(i,T),T), j\neq i} b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi}) \qquad (i \neq 1), \end{aligned} \tag{10}$$

$$f_T^i(y;\boldsymbol{\beta},\boldsymbol{\pi}) = \pi(\mathcal{S}[y]) \qquad (i = 1), \tag{11}$$

where $\mathrm{ch}(i,T)$ 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 ith non-terminal symbol in T. For example, for the tree shown in Fig. 4, $\mathrm{ch}(3,T) = \{5,6\}$, $\mathrm{pa}(5,T) = 3$, and $g_T^2 = \sin$.

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

$$P(T;\boldsymbol{\beta},\boldsymbol{\pi}) = \sum_{x\in H} \pi(\mathcal{S}[x]) b_T^1(x;\boldsymbol{\beta},\boldsymbol{\pi}), \tag{12}$$

$$\begin{aligned} P(T;\boldsymbol{\beta},\boldsymbol{\pi}) = &\sum_{x,y\in H} \Big\{ \beta(\mathcal{S}[x] \to g\, \mathcal{S}[y]...\mathcal{S}[y]) f_T^i(x;\boldsymbol{\beta},\boldsymbol{\pi}) \\ &\times \prod_{j\in \mathrm{ch}(i,T)} b_T^j(y;\boldsymbol{\beta},\boldsymbol{\pi}) \Big\}. \quad (i \in \mathrm{cover}(g,T)) \end{aligned} \tag{13}$$

Here, $\mathrm{cover}(g,T_i)$ 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. 4, then $\mathrm{cover}(+,T) = \{1,3\}$.

4.3. Parameter update formula

We describe the parameter estimation in PCFG-LA. Because PCFG-LA contains latent variables X, the parameter estimation is carried out with the EM algorithm. Let $\boldsymbol{\beta}$ and $\boldsymbol{\pi}$

be current parameters $\overline{\boldsymbol{\beta}}$ and $\overline{\boldsymbol{\pi}}$ be nextstep parameters. The $\mathcal{Q}$ function to optimize in the EM algorithm can be expressed as follows:

$$\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}} | \boldsymbol{\beta}, \boldsymbol{\pi}) = \sum_{i=1}^{N} \sum_{X_i} P(X_i | T_i; \boldsymbol{\beta}, \boldsymbol{\pi}) \log P(T_i, X_i; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}), \tag{14}$$

where N is the number of learning data (promising solutions in EDA). A set of learning data is represented by $\mathcal{D} \equiv \{T_1, T_2, \cdots, T_N\}$. Using the forward–backward probabilities and maximizing $\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}} | \boldsymbol{\beta}, \boldsymbol{\pi})$ under constraints $\sum_{\alpha} \beta(\mathcal{S}[x] \rightarrow \alpha) = 1$ and $\sum_{x} \pi(\mathcal{S}[x]) = 1$, we obtain the following update formula:

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

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

The EM algorithm maximizes the log-likelihood given by

$$\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\pi}; \mathcal{D}) = \sum_{i=1}^{N} \log P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}). \tag{17}$$

By iteratively performing Equations 15–16, the log-likelihood monotonically increases and we obtain locally maximum likelihood estimation parameters. For the case of the EM algorithm, the annotation size h has to be given in advance. 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 Akaike information criteria (AIC) or 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. In Ref. ([12]), we derived variational Bayesian (VB) ([2]) based inference for PCFG-LA, which can estimate the optimal annotation size. Because the derivation of the VB-based algorithm is much more complicated than that of the EM algorithm and because such explanation is outside the scope of this chapter, we do not explain the details of the VB-based algorithm. For details of VB-based PAGE, please read Ref. ([12]).

The procedures of PAGE are listed below.

1. Generate initial population
   Initial population $\mathcal{P}_0$ is generated by randomly creating M individuals.
2. Select promising solutions
   N individuals $\mathcal{D}_g$ are selected from a population of gth generation $\mathcal{P}_g$. In our implementation, we use the truncation selection.
3. Parameter estimation
   Using a parameter update formula (Equations 15–16), converged parameters $(\boldsymbol{\beta}_*, \boldsymbol{\pi}_*)$ are estimated with learning data $\mathcal{D}_g$.

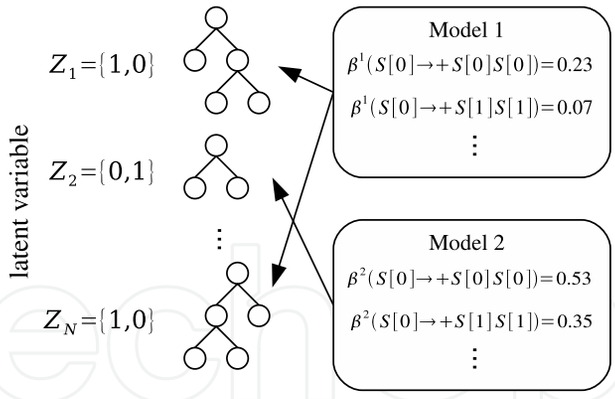


Figure 5. Illustrative description of PCFG-LAMM used in UPAGE.

4. Generation of new individuals
   EDA generates new individuals by sampling from the predictive posterior distributions, namely

$$P(T, X|\mathcal{D}_g) = P(T, X; \boldsymbol{\beta}_*, \boldsymbol{\pi}_*).$$

   Since the EM algorithm is a point estimation method, new individuals can be generated with probabilistic logic sampling which is computationally efficient. The details of the sampling procedures are summarized below (note, when at the maximum depth limitation, select terminal nodes unconditionally).

   (a) A root node is selected following probability distribution $\boldsymbol{\pi}_* = \{\pi_*(\mathcal{S}[x])|x \in H\}$.

   (b) If there are non-terminal symbols $\mathcal{S}[x]$ $(x \in H)$ in a derivation tree, select a production rule according to the probability distribution

$$\beta_*(\mathcal{S}[x]) = \{\beta_*(\mathcal{S}[x] \rightarrow \alpha)|\mathcal{S}[x] \rightarrow \alpha \in \mathcal{R}[H]\}.$$

   Repeat (b) until there are no non-terminal symbols left in the derivation tree.

5. Unsupervised PAGE

In this section, we introduce UPAGE ([11]) which is a mixture model extension of PAGE. UPAGE uses PCFG-LAMM as a baseline grammar, and we explain details of PCFG-LAMM and a parameter update formula in this section.

5.1. PCFG-LAMM

Although PCFG-LA is suitable for estimating local dependencies among nodes, it cannot consider global contexts behind individuals. Suppose there are two optimal solutions represented by $F_1(x)$ and $F_2(x)$. In this case, a population includes solution candidates for $F_1(x)$ and $F_2(x)$ at the same time. Since building blocks for two optimal solutions are different, model and parameter learning with one model results in slow convergence due to the mixed learning data. Furthermore in GP, there are multiple optimal structures even if the problems to be solved are not multimodal. For instance, if an optimum includes a substructure represented by $\sin(2x)$, $\sin(2x)$ as well as $2\sin(x)\cos(x)$ which are mathematically equivalent can be building blocks, where their tree representations are different. When modeling such a mixed population, it is very difficult for PCFG-LA to estimate these multiple structures separately

as in the multimodal case. We have proposed a PCFG-LAMM which is a mixture model extension of PCFG-LA and have also proposed UPAGE based on PCFG-LAMM.

PCFG-LAMM assumes that the probability distributions are a mixture of more than two PCFG-LA models. In PCFG-LAMM, each solution is considered to be sampled from either of the PCFG-LA models (Figure 5). We introduce a latent variable z_i^k, where z_i^k is 1 when the ith derivation tree is generated from the kth model and 0 otherwise ($Z_i = \{z_i^1, z_i^2, \cdots, z_i^\mu\}$). We summarized variables in Appendix B. As a consequence, PCFG-LAMM handles X_i and Z_i as latent variables. The likelihood of complete data is given by

$$\begin{aligned} P(T_i, X_i, Z_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) &= \prod_{k=1}^{\mu} \left\{ \zeta^k P(T_i, X_i; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \right\}^{z_i^k} \\ &= \prod_{k=1}^{\mu} \left\{ \zeta^k \prod_{x \in H} \pi^k(\mathcal{S}[x])^{\delta(x;T_i,X_i)} \prod_{r \in \mathcal{R}[H]} \beta^k(r)^{c(r;T_i,X_i)} \right\}^{z_i^k}, \end{aligned} \tag{18}$$

where ζ^k is the mixture ratio of the kth model ($\boldsymbol{\zeta} = \{\zeta^1, \zeta^2, \cdots, \zeta^\mu\}$ where $\sum_k \zeta^k = 1$). $\beta^k(r)$ and $\pi^k(\mathcal{S}[x])$ denote the probabilities of production rule r and root $\mathcal{S}[x]$ of the kth model, respectively. By calculating the marginal of Equation 18 with respect to X_i and Z_i, the likelihood of observed tree T_i is calculated as

$$\begin{aligned} P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) &= \sum_{k=1}^{\mu} \left\{ \zeta^k P(T_i; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \right\} \\ &= \sum_{k=1}^{\mu} \left\{ \zeta^k \sum_{x \in H} \pi^k(\mathcal{S}[x]) b_{T_i}^1(x; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \right\}. \end{aligned} \tag{19}$$

5.2. Parameter update formula

As in PCFG-LA, the parameter inference of PCFG-LAMM is carried out via the EM algorithm because PCFG-LAMM contains latent variables X_i and Z_i. Let $\boldsymbol{\beta}$, $\boldsymbol{\pi}$ and $\boldsymbol{\zeta}$ be current parameters $\overline{\boldsymbol{\beta}}$, $\overline{\boldsymbol{\pi}}$ and $\overline{\boldsymbol{\zeta}}$ be nextstep parameters. The $\mathcal{Q}$ function of the EM algorithm is given by

$$\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, \overline{\boldsymbol{\zeta}} | \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) = \sum_{i=1}^{N} \sum_{X_i} \sum_{Z_i} P(X_i, Z_i | T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) \log P(T_i, X_i, Z_i; \overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, \overline{\boldsymbol{\zeta}}). \tag{20}$$

By maximizing $\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, \overline{\boldsymbol{\zeta}} | \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})$ under constraints ($\sum_k \zeta^k = 1$, $\sum_\alpha \beta^k(\mathcal{S}[x] \to \alpha) = 1$ and $\sum_x \pi^k(\mathcal{S}[x]) = 1$), a parameter update formula can be obtained as follows (see Appendix B):

$$\begin{aligned} \overline{\beta}^k(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y]) \propto \sum_{i=1}^{N} \Bigg\{ & \frac{\beta^k(\mathcal{S}[x] \to g\, \mathcal{S}[y] \cdots \mathcal{S}[y])}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})} \zeta^k \\ & \times \sum_{\ell \in \mathrm{cover}(g, T_i)} f_{T_i}^{\ell}(x; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \prod_{j \in \mathrm{ch}(\ell, T_i)} b_{T_i}^{j}(y; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \Bigg\}, \end{aligned} \tag{21}$$

$$\overline{\pi}^k \propto \sum_{i=1}^{N} \left\{ \frac{\pi^k(\mathcal{S}[x])}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})} \zeta^k b_{T_i}^1(x; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \right\}, \tag{22}$$

$$\overline{\zeta}^k \propto \sum_{i=1}^{N} \left\{ \frac{\zeta^k P(T_i; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k)}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})} \right\}. \tag{23}$$

The parameter inference starts from some initial values and converges to a local optimum using Equations 21–23. A log-likelihood is given by

$$\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}; \mathcal{D}) = \sum_{i=1}^{N} \log P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}). \tag{24}$$

The procedures of UPAGE are listed below.

1. Generate initial population
   Initial population $\mathcal{P}_0$ is generated by randomly creating M individuals. In our implementation, the ratio between production rules of function nodes (e.g. $\mathcal{S}[x] \rightarrow +\,\mathcal{S}[y]\,\mathcal{S}[y]$) and those of terminal nodes (e.g. $\mathcal{S}[x] \rightarrow +\,\mathcal{S}[y]\,\mathcal{S}[y]$) are set to 4 : 1.
2. Select promising solutions
   N individuals $\mathcal{D}_g$ are selected from a population of gth generation $\mathcal{P}_g$. In our implementation, we used the truncation selection.
3. Parameter estimation
   Using a parameter update formula (Equations 21–23), converged parameters $(\boldsymbol{\beta}_*, \boldsymbol{\pi}_*, \boldsymbol{\zeta}_*)$ are estimated with learning data $\mathcal{D}_g$.
4. Generation of new individuals
   EDA generates new individuals by sampling from the predictive posterior distributions, namely
   $$P(T, X, Z|\mathcal{D}_g) = P(T, X, Z; \boldsymbol{\beta}_*, \boldsymbol{\pi}_*, \boldsymbol{\zeta}_*).$$
   Since the EM algorithm is a point estimation method, new individuals can be generated with probabilistic logic sampling, which is computationally cheap. The details of the sampling procedures are summarized below (note, when at the maximum depth limitation, select a terminal node unconditionally).
   (a) Select a model following probability distribution $\boldsymbol{\zeta}_* = \{\zeta_*^1, \zeta_*^2, \cdots, \zeta_*^\mu\}$.
   (b) Let the selected model index be ℓ. A root node is selected following probability distribution $\boldsymbol{\pi}_*^\ell = \{\pi_*^\ell(\mathcal{S}[x])|x \in H\}$.
   (c) If there are non-terminal symbols $\mathcal{S}[x]$ $(x \in H)$ in a derivation tree, select a production rule following the probability distribution
   $$\beta_*^\ell(\mathcal{S}[x]) = \{\beta_*^\ell(\mathcal{S}[x] \rightarrow \alpha)|\mathcal{S}[x] \rightarrow \alpha \in \mathcal{R}[H]\}.$$
   Repeat (c) until there are no non-terminal symbols left in the derivation tree.

5.3. Computer experiments

In order to show the effectiveness of UPAGE, we analyze UPAGE from the viewpoint of the number of fitness evaluations. We applied UPAGE to three benchmark problems: the royal tree problem (Section 5.3.1), the bipolar royal tree problem (Section 5.3.2) and the deceptive MAX (DMAX) problem (Section 5.3.3). Because we want to study the effectiveness of the mixture model versus PCFG-LA, we specifically compared UPAGE with PAGE. In each benchmark test, we employed the parameter settings shown in Table 1, where UPAGE and

PAGE and UPAGE				
	Meaning	Royal Tree	Bipolar Royal Tree	DMAX
M	Population size	1000	3000	3000
P_s	Selection rate	0.1	0.1	0.1
P_e	Elite rate	0.01	0.01	0.01

UPAGE				
	Meaning	Royal Tree	Bipolar Royal Tree	DMAX
h	Annotation size	11	22	22
μ	The number of mixtures	2	2	2

PAGE				
	Meaning	Royal Tree	Bipolar Royal Tree	DMAX
h	Annotation size	16	32	32

Table 1. Main parameter settings of UPAGE and PAGE.

PAGE used the same population size, elite rate and selection rate. For the method-specific parameters of PAGE and UPAGE, we determined h and μ so that the number of parameters to be estimated is almost the same in UPAGE and PAGE. In the three benchmark problems, we carried out UPAGE and PAGE 30 times to compare the number of fitness evaluations and also performed the Welch t-test (two-tailed) to determine the statistical significance.

5.3.1. Royal tree problem

We apply UPAGE to the royal tree problem ([22]), which has only one optimal solution. The royal tree problem is a popular benchmark problem in GP. The royal tree problem is suitable for analyzing GP because the optimal structure of the royal tree is composed of smaller substructures (building blocks), and hence it well reflects the behavior of GP.

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 one from a root to leaves in a tree. A function a has a terminal x. The fitness function of the royal tree problem is given by

$$Score(\mathcal{X}_i) = wb_i \sum_j (wa_{ij} \times Score(\mathcal{X}_{ij})), \tag{25}$$

where $\mathcal{X}_i$ is the ith node in tree structures, and $\mathcal{X}_{ij}$ denotes the jth child of $\mathcal{X}_i$. The fitness value of the royal tree problem is calculated recursively from a root node. In Equation 25, wb_i and wa_{ij} are weights which are defined as follows:

- wa_{ij}
  - $Full\,Bonus = 2$
    If a subtree rooted at $\mathcal{X}_{ij}$ has a correct root and is a perfect tree.

	Average number of fitness evaluations	Standard deviation
UPAGE	6171	28
PAGE	6237	18

P-value of t-test (Welch, two-tailed)
0.74

Table 2. The number of fitness evaluations, standard deviation and P-value of t-test in the royal tree problem.

  - $Partial\ Bonus = 1$
    If a subtree rooted at $\mathcal{X}_{ij}$ has a correct root but is not a perfect tree.
  - $Penalty = 1/3$
    If $\mathcal{X}_{ij}$ is not a correct root.
- wb_i
  - $Complete\ Bonus = 2$
    If a subtree rooted at $\mathcal{X}_i$ is a perfect tree.
  - $Otherwise = 1$

In the present chapter, we employ the following GP functions and terminals:

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

Here, $\mathfrak{F}$ and $\mathfrak{T}$ denote function and terminal sets, respectively, of GP. For details of the royal tree problem, please see Ref. ([22]).

Table 2 shows the average number of fitness evaluations (along with their standard deviation) and the P-value of a t-test (Welch, two-tailed). As can been seen with Table 2, there is no noticeable difference between UPAGE and PAGE in the average number of fitness evaluations, which is confirmed by the P-value of t-test. The royal tree problem is not multimodal, and hence the optimal solution has only one tree expression. Consequently, we do not have to consider global contexts behind optimal solutions, which is an advantage of UPAGE over PAGE.

5.3.2. Bipolar royal tree problem

We next apply UPAGE to the bipolar royal tree problem. In the field of GA-EDAs, a mixture model based method UEBNA was proposed, and it was reported that UEBNA is especially effective in multimodal problems such as two-max problem. Consequently, we apply UPAGE to a bipolar problem having two optimal solutions, which is a multimodal extension of the royal tree problem. In order to make the royal tree problem multimodal, we set $\mathfrak{T} = \{x, y\}$ and $Score(x) = Score(y) = 1$. With this setting, the royal tree problem has two optimal solutions of x (Fig. 7(a)) and y (Fig. 7(b)). PAGE and UPAGE stop when either of the two optimal solutions is obtained.

Table 3 shows the average number of fitness evaluations along with their standard deviation. We see that UPAGE can obtain an optimal solution with a smaller number of fitness

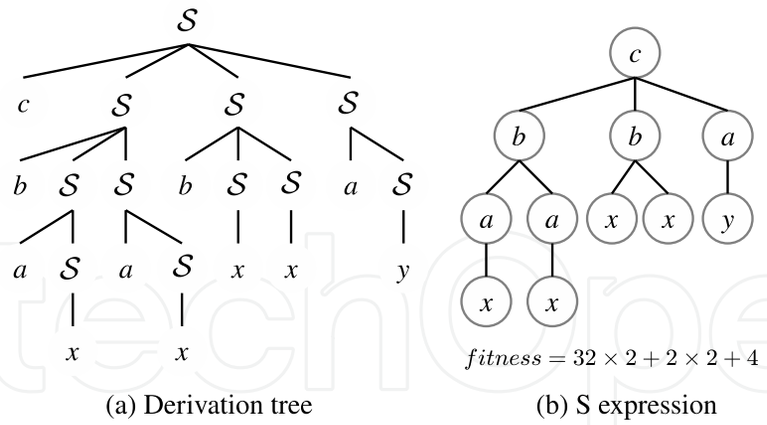


Figure 6. Example of fitness calculation in the bipolar royal tree problem. (a) Derivation tree and (b) S-expression.

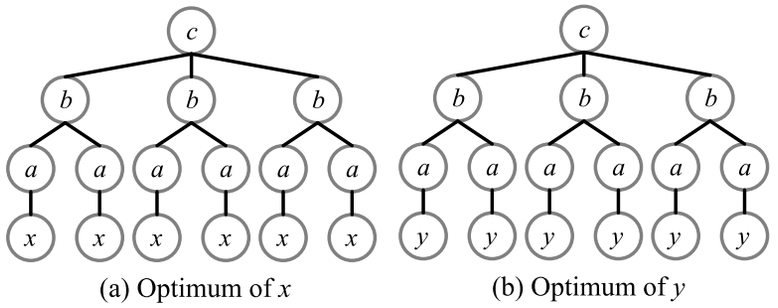


Figure 7. (a) Optimum structure of x and (b) that of y in the bipolar royal tree problem. These two structures have the same fitness value.

evaluations than PAGE. Table 3 gives the P-value of a t-test (Welch, two-tailed), which allows us to say that the difference between UPAGE and PAGE is statistically significant.

Because the bipolar royal tree problem has two optimal solutions (x and y), PAGE learns the production rule probabilities with learning data containing solution candidates of both x and y optima. Let us consider the annotation size required to express optimal solutions of the bipolar royal tree problem of depth 5. For the case of PAGE, the minimum annotation size to be able to learn the two optimal solutions separately is 10. In contrast, UPAGE can express the two optimal solutions with mixture size 2 and annotation size 5, which results in a smaller number of parameters. This consideration shows that a mixture model is more suitable for this class of problems.

Figure 8 shows the increase in the log-likelihood for the bipolar royal tree problem, in particular, the transitions at generation 0 and generation 5. As can been seen from the figure, the log-likelihood converges after about 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.

5.3.3. DMAX Problem

We apply UPAGE to the DMAX problem ([8, 10]), which has deceptiveness when it is solved with GP. 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

	Average number of fitness evaluations	Standard deviation
UPAGE	25839	4737
PAGE	31878	4333

P-value of t-test (Welch, two-tailed)
4.49×10^{-6}

Table 3. The number of fitness evaluations, standard deviation and P-value of t-test in the bipolar royal tree problem.

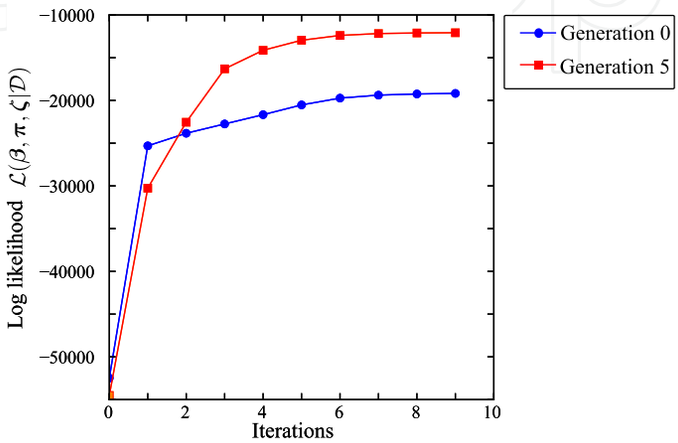


Figure 8. Transitions of loglikelihood of UPAGE in the bipolar royal tree problem.

maximum tree depth. However, the symbols used in the DMAX problem are different from those used in the MAX problem. The DMAX problem has three parameters, and the difficulty of the problem can be tuned using these three parameters. For the problem of interest in the present chapter, we selected $m = 3$ and $r = 2$, whose deceptiveness is of medium degree. In this setting, the GP terminals and functions are

$$\mathfrak{F} = \{+_3, \times_3\},$$
$$\mathfrak{T} = \{0.95, -1\},$$

where $+_3$ and $\times_3$ are 3 arity addition and multiplication operators, respectively. The optimal solution in the present setting is given by

$$(-1 \times 3)^{26}(0.95 \times 3) \fallingdotseq 7.24 \times 10^{12}. \tag{26}$$

Table 4 shows the average number of fitness evaluations along with their standard deviation for the DMAX problem. We can see that UPAGE obtained the optimal solution with a smaller number of fitness evaluations compared to PAGE. Table 4 gives the P-value of a t-test (Welch and two-tailed) and allows us to say that the difference in the averages of UPAGE and PAGE is statistically significant.

In the bipolar royal tree problem, expressions of the two optimal solutions (x or y) are different, and thus building blocks of the optima are also different. In contrast, the DMAX problem has mathematically only one optimal solution, which are represented by Equation 26. Although the DMAX problem is a unimodal problem, the DMAX problem has different expressions for the optimal solution due to commutative operators such as $+_3$ and $\times_3$. From this experiment, we see that UPAGE is superior to PAGE for this class of benchmark problems.

	Average number of fitness evaluations	Standard deviation
UPAGE	36729	3794
PAGE	38709	2233

P-value of t-test (Welch, two-tailed)
1.94×10^{-2}

Table 4. The number of fitness evaluations, standard deviation and P-value of t-test in the DMAX problem.

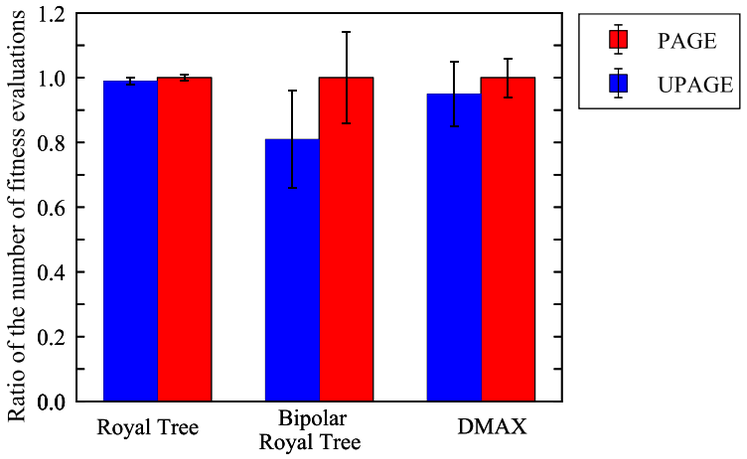


Figure 9. The average number of fitness evaluations (smaller is better) in royal tree problem, bipolar royal tree problem and DMAX problem relative to those of PAGE (i.e. the PAGE results are normalized to 1).

Common parameters in PAGE and UPAGE		
	Meaning	Bipolar Royal Tree
M	Population size	6000
P_s	Selection rate	0.3
P_e	Elite rate	0.1

UPAGE		
	Meaning	Bipolar Royal Tree
h	Annotation size	16
μ	The number of mixtures	4

PAGE		
	Meaning	Bipolar Royal Tree
h	Annotation size	32

Table 5. Parameter settings for a multimodal problem.

5.4. Multimodal problem

In the preceding section, we evaluated the performance of UPAGE from the viewpoint of the average number of fitness evaluations. In this section, we show the effectiveness of UPAGE in terms of its capability for obtaining multiple solutions of a multimodal problem. Because there are two optimal solutions in the bipolar royal tree problem (see Fig. 7(a) and (b)), we

	Successful runs / Total runs
UPAGE	10/15
PAGE	0/15

Table 6. The number of runs which could obtain both optimal solutions. We carried out 15 runs in total.

show that UPAGE can obtain both optimal solutions in a single run. Parameter settings are shown in Table 5.

Table 6 shows the number of successful runs in which both optimal solutions are obtained in a single run. As can been seen in Table 6, UPAGE succeeded in obtaining both optimal solutions in 10 out of 15 runs, whereas PAGE could not obtain them at all.

Table 7 shows production rule probabilities of UPAGE in a successful run. Although the mixture size is $\mu = 4$, we have only presented probabilities of Model = 0 and Model = 3, which are related to optimal solutions of y (Fig. 7(b)) and x (Fig. 7(a)), respectively (i.e. Model = 1 and Model = 2 are not shown). Because we see in Model = 0 that the probabilities generating y are very high, we consider that the optimal solution of y was generated by Model = 0. On the other hand, it is estimated that the optimal solution of x was generated by Model = 3. From this probability table, we can confirm that UPAGE successfully estimated the mixed population separately, because Model = 3 and 0 can generate optimal solutions of x and y with relatively high probability. It is very difficult for PAGE to estimate multiple solutions because PCFG-LA is not a mixture model and it is almost impossible to learn the distributions separately. As was shown in Section 5.3, UPAGE is superior to PAGE in terms of the number of fitness evaluations. From Table 7, it is considered that this superiority is due to UPAGE's capability of learning distributions in a separate way.

6. Discussion

In the present chapter, we have introduced PAGE and UPAGE. PAGE is based on PCFG-LA, which takes into account latent annotations to weaken the context freedom assumption. By considering latent annotations, dependencies among nodes can be considered. We reported in Ref. ([12]) that PAGE is more powerful for several benchmark tests than other GP-EDAs, including GMPE and POLE.

Although PCFG-LA is suitable for estimating dependencies among local nodes, it cannot consider global contexts (contexts of entire tree structures) behind individuals. In many real-world problems, not only local dependencies but also global contexts have to be taken into account. In order to consider the global contexts, we have proposed UPAGE by extending PCFG-LA into a mixture model (PCFG-LAMM). In the bipolar royal tree problem, there are two optimal structures of x and y and the global contexts represent which optima (x or y) each tree structure comes from. From Table 7, the mixture model of UPAGE successfully worked and UPAGE could estimate mixed population separately. We have also shown that a mixture model is effective not only in multimodal problems but also in some unimodal problems, namely in the DMAX problem. Although the optimal solution of the DMAX problem is represented by mathematically one expression, the tree expressions are not unique, due to commutative operators ($\times_3$ and $+_3$). Consequently, the mixture model is also effective in the DMAX problem (see Section 5.3.3), and this situation where there exists the expression diversity often arises in real world problems. When obtaining multiple optimal solutions in a single run, UPAGE succeeded in cases for which PAGE obtained only one of the

Model = 0	Pr
ζ^0	0.11
$\mathcal{S}[1]$	1.00
$\mathcal{S}[0] \rightarrow a\,\mathcal{S}[10]$	0.20
$\mathcal{S}[0] \rightarrow a\,\mathcal{S}[2]$	0.18
$\mathcal{S}[0] \rightarrow a\,\mathcal{S}[5]$	0.28
$\mathcal{S}[1] \rightarrow d\,\mathcal{S}[4]\,\mathcal{S}[4]\,\mathcal{S}[4]\,\mathcal{S}[4]$	1.00
$\mathcal{S}[10] \rightarrow x$	0.14
$\mathcal{S}[10] \rightarrow y$	0.86
$\mathcal{S}[11] \rightarrow x$	0.14
$\mathcal{S}[11] \rightarrow y$	0.86
$\mathcal{S}[12] \rightarrow a\,\mathcal{S}[10]$	0.17
$\mathcal{S}[12] \rightarrow a\,\mathcal{S}[2]$	0.18
$\mathcal{S}[12] \rightarrow a\,\mathcal{S}[5]$	0.32
$\mathcal{S}[13] \rightarrow x$	0.21
$\mathcal{S}[13] \rightarrow y$	0.79
$\mathcal{S}[14] \rightarrow b\,\mathcal{S}[7]\,\mathcal{S}[7]$	0.10
$\mathcal{S}[14] \rightarrow c\,\mathcal{S}[10]\,\mathcal{S}[10]\,\mathcal{S}[10]$	0.15
$\mathcal{S}[15] \rightarrow x$	0.12
$\mathcal{S}[15] \rightarrow y$	0.88
$\mathcal{S}[2] \rightarrow x$	0.25
$\mathcal{S}[2] \rightarrow y$	0.75
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[10]$	0.21
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[15]$	0.18
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[2]$	0.17
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[5]$	0.22
$\mathcal{S}[4] \rightarrow c\,\mathcal{S}[8]\,\mathcal{S}[8]\,\mathcal{S}[8]$	1.00
$\mathcal{S}[5] \rightarrow y$	0.97
$\mathcal{S}[6] \rightarrow y$	1.00
$\mathcal{S}[7] \rightarrow x$	0.52
$\mathcal{S}[7] \rightarrow y$	0.48
$\mathcal{S}[8] \rightarrow b\,\mathcal{S}[0]\,\mathcal{S}[0]$	0.50
$\mathcal{S}[8] \rightarrow b\,\mathcal{S}[12]\,\mathcal{S}[12]$	0.17
$\mathcal{S}[8] \rightarrow b\,\mathcal{S}[3]\,\mathcal{S}[3]$	0.31
$\mathcal{S}[9] \rightarrow x$	0.14
$\mathcal{S}[9] \rightarrow y$	0.86

Model = 3	Pr
ζ^3	0.52
$\mathcal{S}[11]$	1.00
$\mathcal{S}[0] \rightarrow a\,\mathcal{S}[13]$	0.16
$\mathcal{S}[0] \rightarrow a\,\mathcal{S}[2]$	0.29
$\mathcal{S}[0] \rightarrow a\,\mathcal{S}[5]$	0.32
$\mathcal{S}[1] \rightarrow b\,\mathcal{S}[0]\,\mathcal{S}[0]$	0.13
$\mathcal{S}[1] \rightarrow b\,\mathcal{S}[14]\,\mathcal{S}[14]$	0.19
$\mathcal{S}[1] \rightarrow b\,\mathcal{S}[3]\,\mathcal{S}[3]$	0.15
$\mathcal{S}[1] \rightarrow b\,\mathcal{S}[7]\,\mathcal{S}[7]$	0.17
$\mathcal{S}[1] \rightarrow b\,\mathcal{S}[8]\,\mathcal{S}[8]$	0.32
$\mathcal{S}[10] \rightarrow c\,\mathcal{S}[1]\,\mathcal{S}[1]\,\mathcal{S}[1]$	1.00
$\mathcal{S}[11] \rightarrow d\,\mathcal{S}[10]\,\mathcal{S}[10]\,\mathcal{S}[10]\,\mathcal{S}[10]$	1.00
$\mathcal{S}[12] \rightarrow a\,\mathcal{S}[4]$	0.13
$\mathcal{S}[12] \rightarrow c\,\mathcal{S}[13]\,\mathcal{S}[13]\,\mathcal{S}[13]$	0.34
$\mathcal{S}[12] \rightarrow x$	0.13
$\mathcal{S}[13] \rightarrow x$	0.72
$\mathcal{S}[13] \rightarrow y$	0.28
$\mathcal{S}[14] \rightarrow a\,\mathcal{S}[15]$	0.16
$\mathcal{S}[14] \rightarrow a\,\mathcal{S}[4]$	0.10
$\mathcal{S}[14] \rightarrow a\,\mathcal{S}[5]$	0.45
$\mathcal{S}[14] \rightarrow a\,\mathcal{S}[6]$	0.13
$\mathcal{S}[15] \rightarrow x$	0.89
$\mathcal{S}[15] \rightarrow y$	0.11
$\mathcal{S}[2] \rightarrow x$	0.99
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[13]$	0.11
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[15]$	0.14
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[2]$	0.20
$\mathcal{S}[3] \rightarrow a\,\mathcal{S}[5]$	0.44
$\mathcal{S}[4] \rightarrow x$	0.68
$\mathcal{S}[4] \rightarrow y$	0.32
$\mathcal{S}[5] \rightarrow x$	0.92
$\mathcal{S}[6] \rightarrow x$	0.93
$\mathcal{S}[7] \rightarrow a\,\mathcal{S}[13]$	0.23
$\mathcal{S}[7] \rightarrow a\,\mathcal{S}[2]$	0.31
$\mathcal{S}[7] \rightarrow a\,\mathcal{S}[4]$	0.10
$\mathcal{S}[7] \rightarrow a\,\mathcal{S}[5]$	0.29
$\mathcal{S}[8] \rightarrow a\,\mathcal{S}[2]$	0.17
$\mathcal{S}[8] \rightarrow a\,\mathcal{S}[4]$	0.18
$\mathcal{S}[8] \rightarrow a\,\mathcal{S}[5]$	0.41
$\mathcal{S}[8] \rightarrow a\,\mathcal{S}[6]$	0.16
$\mathcal{S}[9] \rightarrow a\,\mathcal{S}[13]$	0.19
$\mathcal{S}[9] \rightarrow a\,\mathcal{S}[4]$	0.19
$\mathcal{S}[9] \rightarrow a\,\mathcal{S}[5]$	0.38

Table 7. Estimated parameters by UPAGE in a successful run. Although the number of mixtures is $\mu = 4$, we only show Model = 0 and Model = 3 related to optimal solutions of y and x, respectively. Due to limited space, we do not show parameters of production rules which are smaller than 0.1.

Method	Estimation of interaction among nodes	Position independent model	Consideration of global contexts
Scalar SG-GP	No	Yes	No
Vectorial SG-GP	Partially	No	No
GT-EDA	Yes	No	No
GMPE	Yes	Yes	No
PAGE	Yes	Yes	No
UPAGE	Yes	Yes	Yes

Table 8. Classification of GP-EDAs and their capabilities.

optima. This result shows that UPAGE is more effective than PAGE not only quantitatively but also qualitatively. We also note that UPAGE is more powerful than PAGE in terms of computational time. In our computer experiments, we set the number of parameters in UPAGE and PAGE to be approximately the same. Figure 10 shows the relative computational time per generation of UPAGE and PAGE (the computational time of PAGE is normalized to 1) and we see that UPAGE required only sixty percent of the time required by PAGE. Although we have shown in Section 5.3.1 that UPAGE and PAGE required approximately the same number of fitness evaluations to obtain the optimal solution in the royal tree problem, UPAGE is more effective even for the royal tree problem if the actual computational time is considered.

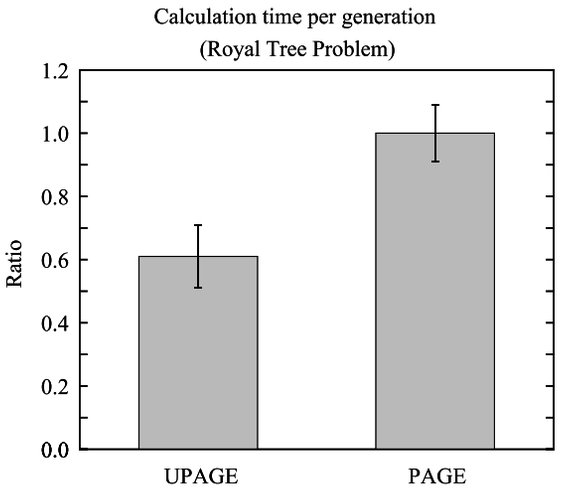


Figure 10. The computational time per generation of UPAGE and PAGE (smaller is better). The time of PAGE is normalized to 1.

Table 8 summarizes functionalities of several GP-EDAs. SG-GP employs the conventional PCFG and hence it cannot estimate dependencies among nodes. Although GT-EDA, GMPE and PAGE adopt different types of grammar models, they belong to the same class in the sense that these three methods can take into account dependencies among nodes, which is enabled by a use of specialized production rules depending on contexts. However, these methods cannot consider global contexts, and consequently, they are not suitable for estimating problems having complex distributions. In contrast, in addition to local dependencies among nodes, UPAGE can consider global contexts of tree structures. The model of UPAGE is the most flexible among these GP-EDAs, and this flexibility is reflected by the search performance.

In the present implementation of UPAGE, we had to set the mixture size μ and the annotation size h in advance because UPAGE employed the EM algorithm. However, it is desirable to

estimate μ and h, as well as β, π and ζ during search. In the case of PAGE, we proposed PAGE-VB in Ref. ([12]), which adopted VB to estimate the annotation size h. In a similar fashion, it is possible to apply VB to UPAGE to enable the inference of μ and h.

We have shown the effectiveness of PAGE and UPAGE with benchmark problems not having intron structures. However, in real-world applications, problems generally include intron structures, which make the model and parameter inference much more difficult. For such problems, we consider that intron removal algorithms ([13, 30]) are effective, and application of such algorithms to GP-EDAs is left as a topic of future study.

7. Conclusion

We have introduced a probabilistic program evolution algorithm named PAGE and its extension UPAGE. PAGE takes advantage of latent annotations that enables consideration of dependencies among nodes, and UPAGE incorporates a mixture model for taking into account global contexts. By applying UPAGE to computational experiments, we have confirmed that a mixture model is highly effective for obtaining solutions in terms of the number of fitness evaluations. At the same time, UPAGE is more advantageous than PAGE in the sense that UPAGE can obtain multiple solutions for multimodal problems. We hope that it will be possible to apply PAGE and UPAGE to a wide class of real-world problems, which is an intended future area of study.

Author details

Yoshihiko Hasegawa
The University of Tokyo, Japan

Appendix A: Parameter list

We summarized parameters used in PAGE and UPAGE in the following table.

Target model	Parameter	Meaning
PAGE and UPAGE	$\delta(x;T,X)$	Frequency of a root $\mathcal{S}[x]$ in a complete tree (0 or 1)
	$c(r;T,X)$	Frequency of a production rule r in a complete tree
	h	Annotation size
	H	Set of annotation $H=\{0,1,\cdots h-1\}$
	T_i	Observed derivation tree
	x_i^j	jth latent annotation in T_i
	$\mathcal{R}[H]$	Set of production rules
	$\mathcal{N}$	Set of non-terminals in CFG
	$\mathcal{T}$	Set of terminals in CFG
	$\mathfrak{F}$	Set of function nodes in GP
	$\mathfrak{T}$	Set of terminal nodes in GP
PAGE	$\pi(\mathcal{S}[x])$	Probability of a root $\mathcal{S}[t]$
	$\beta(r)$	Probability of a production rule r
UPAGE	ζ^k	Mixture ratio of kth model.
	$\pi^k(\mathcal{S}[x])$	Probability of a root $\mathcal{S}[t]$ in kth model.
	$\beta^k(r)$	Probability of a production rule r in kth model
	z_i^k	$z_k^k=1$, if ith individual belongs to kth model
	μ	Mixture size

Appendix B: Derivation of a parameter update formula for UPAGE

We here explain details of the parameter update formula for UPAGE (see Section 4.1). By separating $\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, \overline{\boldsymbol{\zeta}} | \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})$ into terms containing $\overline{\boldsymbol{\beta}}$, $\overline{\boldsymbol{\pi}}$ and $\overline{\boldsymbol{\zeta}}$, a parameter update formula for $\overline{\boldsymbol{\beta}}$, $\overline{\boldsymbol{\pi}}$ and $\overline{\boldsymbol{\zeta}}$ can be calculated separately.

We here derive $\overline{\boldsymbol{\beta}}$. Maximization of $\mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, \overline{\boldsymbol{\zeta}} | \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})$ under a constraint $\sum_\alpha \overline{\beta}^k(\mathcal{S}[x] \to \alpha) = 1$ can be performed by the method of Lagrange multipliers:

$$\frac{\partial \mathscr{L}}{\partial \overline{\beta}^k(\mathcal{S}[x] \to \alpha)} = 0, \tag{27}$$

with

$$\mathscr{L} = \mathcal{Q}(\overline{\boldsymbol{\beta}}, \overline{\boldsymbol{\pi}}, \overline{\boldsymbol{\zeta}} | \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) + \sum_{k,x} \xi_{k,x} \left(1 - \sum_\alpha \overline{\beta}^k(\mathcal{S}[x] \to \alpha) \right), \tag{28}$$

where $\xi_{k,x}$ denote Lagrange multipliers. By calculating Equation 27, we obtain the following update formula:

$$\begin{aligned} \overline{\beta}^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]) \propto \sum_{i=1}^N \sum_{X_i} \sum_{Z_i} \Big\{ & P(X_i, Z_i | T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) z_i^k \\ & \times c(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]; T_i, X_i) \Big\}. \end{aligned} \tag{29}$$

Because Equation 29 includes summation in terms of X_i, direct calculation is intractable due to exponential increase of computational cost. Consequently, we use forward–backward probabilities. Let $c^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]; T_i)$ be

$$\begin{aligned} & c^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]; T_i) \\ & = \sum_{X_i} \sum_{Z_i} P(X_i, Z_i | T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) z_i^k c(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]; T_i, X_i). \end{aligned}$$

By differentiating the likelihood of complete data (Equation 18) with respect to $\beta^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y])$, we have

$$\begin{aligned} & c^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y]; T_i) \\ & = \frac{\beta^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y])}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})} \sum_{X_i} \sum_{Z_i} \frac{\partial P(T_i, X_i, Z_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})}{\partial \beta^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y])}. \end{aligned}$$

The last term is calculated as

$$\begin{aligned} \sum_{X_i} \sum_{Z_i} \frac{\partial P(T_i, X_i, Z_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})}{\partial \beta^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y])} &= \zeta^k \sum_{X_i} \frac{\partial P(T_i, X_i; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k)}{\partial \beta^k(\mathcal{S}[x] \to g\,\mathcal{S}[y] \cdots \mathcal{S}[y])} \\ &= \zeta^k \sum_{\ell \in \mathrm{cover}(g, T_i)} f_{T_i}^\ell(x; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \prod_{j \in \mathrm{ch}(\ell, T_i)} b_{T_i}^j(y; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k). \end{aligned}$$

By this procedure, the update formula for β is expressed with Equation 21, and the update formula for π is calculated in a similar way (and much easier). The update formula for ζ is

given by

$$\begin{aligned}
\overline{\zeta}^k &\propto \sum_{i=1}^{N} \sum_{X_i} \sum_{Z_i} P(X_i, Z_i | T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) z_i^k \\
&= \sum_{i=1}^{N} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})} \sum_{X_i} \sum_{Z_i} \left\{ z_i^k P(T_i, X_i, Z_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta}) \right\} \\
&= \sum_{i=1}^{N} \frac{1}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})} \sum_{X_i} \left\{ \zeta^k P(T_i, X_i; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k) \right\} \\
&= \sum_{i=1}^{N} \frac{\zeta^k P(T_i; \boldsymbol{\beta}^k, \boldsymbol{\pi}^k)}{P(T_i; \boldsymbol{\beta}, \boldsymbol{\pi}, \boldsymbol{\zeta})}.
\end{aligned}$$

8. References

[1] Abbass, H. A., Hoai, X. & Mckay, R. I. [2002]. AntTAG: A new method to compose computer programs using colonies of ants, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1654–1659.

[2] Attias, H. [1999]. Inferring parameters and structure of latent variable models by variational Bayes, *the 15th Conference of Uncertainty in Artificial Intelligence*, Morgan Kaufmann, Stockholm, Sweden, pp. 21–30.

[3] Baluja, S. [1994]. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning, *Technical Report CMU-CS-94-163*, Pittsburgh, PA.
URL: *citeseer.ist.psu.edu/baluja94population.html*

[4] Bosman, P. A. N. & de Jong, E. D. [2004]. Grammar transformations in an EDA for genetic programming, *Technical Report UU-CS-2004-047*, Institute of Information and Computing Sciences, Utrecht University.

[5] Dempster, A., Laird, N. & Rubin, D. [1977]. Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B* 39(1): 1–38.

[6] Goldberg, D. E., Deb, D. & Kargupta, H. [1993]. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms, *in* S. Forrest (ed.), *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, Morgan Kaufman, San Mateo, pp. 56–64.

[7] Harik, G. [1999]. Linkage learning via probabilistic modeling in the ECGA, *IlliGAL Report* (99010).

[8] Hasegawa, Y. & Iba, H. [2006]. Estimation of Bayesian Network for Program Generation, *Proceedings of The Third Asian-Pacific Workshop on Genetic Programming*, Hanoi, Vietnam, pp. 35–46.

[9] Hasegawa, Y. & Iba, H. [2007]. Estimation of distribution algorithm based on probabilistic grammar with latent annotations, *Proceedings of IEEE Congress of Evolutionary Computation*, IEEE press, Singapore, pp. 1143–1150.

[10] Hasegawa, Y. & Iba, H. [2008]. A Bayesian network approach for program generation, *IEEE Transactions on Evoluationary Computation* 12(6): 750–764.

[11] Hasegawa, Y. & Iba, H. [2009a]. Estimation of distribution algorithm based on PCFG-LA mixture model, *Transactions of the Japanese Society for Artificial Intelligence (in Japanese)* 24(1): 80–91.

[12] Hasegawa, Y. & Iba, H. [2009b]. Latent variable model for estimation of distribution algorithm based on a probabilistic context-free grammar, *IEEE Transactions on Evolutionary Computation* 13(4): 858–878.
[13] Hooper, D. & Flann, N. S. [1996]. Improving the accuracy and robustness of genetic programming through expression simplification, *Proceedings of the First Annual Conference*, MIT Press, Stanford University, CA, USA.
[14] Larrañaga, P. & Lozano, J. A. [2002]. *Estimation of Distribution Algorithms*, Kluwer Academic Publishers.
[15] Looks, M. [2005]. Learning computer programs with the Bayesian optimization algorithm. Master thesis, Washington University Sever Institute of Technology.
[16] Looks, M. [2007]. Scalable estimation-of-distribution program evolution, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 539–546.
[17] Matsuzaki, T., Miyao, Y. & Tsujii, J. [2005]. Probabilistic CFG with latent annotations, *In Proceedings of the 43rd Meeting of the Association for Computational Linguistics (ACL)*, Morgan Kaufmann, Michigan, USA, pp. 75–82.
[18] Nordin, P. [1994]. A compiling genetic programming system that directly manipulates the machine code, *Advances in genetic programming*, MIT Press, Cambridge, MA, USA, chapter 14, pp. 311–331.
[19] Pelikan, M. & Goldberg, D. E. [2001]. Escaping hierarchical traps with competent genetic algorithms, *GECCO '01: Proceedings of the 2001 conference on Genetic and evolutionary computation*, ACM Press, New York, NY, USA, pp. 511–518.
[20] Pelikan, M., Goldberg, D. E. & Cantú-Paz, E. [1999]. BOA: The Bayesian optimization algorithm, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, Vol. I, Morgan Kaufmann Publishers, San Fransisco, CA, Orlando, FL, pp. 525–532.
[21] Poli, R. & McPhee, N. F. [2008]. A linear estimation-of-distribution GP system, *Proceedings of Euro GP 2008*, Springer-Verlag, pp. 206–217.
[22] Punch, W. F. [1998]. How effective are multiple populations in genetic programming, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, Morgan Kaufmann, University of Wisconsin, Madison, Wisconsin, USA, pp. 308–313.
[23] Ratle, A. & Sebag, M. [2001]. Avoiding the bloat with probabilistic grammar-guided genetic programming, *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001*, Vol. 2310 of *LNCS*, Springer Verlag, Creusot, France, pp. 255–266.
[24] Regolin, E. N. & Pozo, A. T. R. [2005]. Bayesian automatic programming, *Proceedings of the 8th European Conference on Genetic Programming*, Vol. 3447 of *Lecture Notes in Computer Science*, Springer, Lausanne, Switzerland, pp. 38–49.
[25] Sałustowicz, R. P. & Schmidhuber, J. [1997]. Probabilistic incremental program evolution, *Evolutionary Computation* 5(2): 123–141.
[26] Sastry, K. & Goldberg, D. E. [2003]. Probabilistic model building and competent genetic programming, *Genetic Programming Theory and Practise*, Kluwer, chapter 13, pp. 205–220.
[27] Sato, H., Hasegawa, Y., Bollegala, D. & Iba, H. [2012]. Probabilistic model building GP with belief propagation, *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2012)*. accepted for publication.
[28] Shan, Y., McKay, R. I., Abbass, H. A. & Essam, D. [2003]. Program evolution with explicit learning: a new framework for program automatic synthesis, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, IEEE Press, Canberra, pp. 1639–1646.
[29] Shan, Y., McKay, R. I., Baxter, R., Abbass, H., Essam, D. & Hoai, N. X. [2004]. Grammar model-based program evolution, *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, IEEE Press, Portland, Oregon, pp. 478–485.

[30] Shin, J., Kang, M., McKay, R. I., Nguyen, X., Hoang, T.-H., Mori, N. & Essam, D. [2007]. Analysing the regularity of genomes using compression and expression simplification, *Proceedings of Euro GP 2007*, Springer-Verlag, pp. 251–260.

[31] Tanev, I. [2004]. Implications of incorporating learning probabilistic context-sensitive grammar in genetic programming on evolvability of adaptive locomotion gaits of snakebot, *GECCO 2004 Workshop Proceedings*, Seattle, Washington, USA.

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

[33] Whigham, P. A. [1995]. Grammatically-based genetic programming, *Proceedings of the Workshop on Genetic Programming : From Theory to Real-World Applications*, Tahoe City, California USA, pp. 44–41.

[34] Whigham, P. A. [1996]. Search bias, language bias, and genetic programming, *Genetic Programming 1996: Proceedings of the First Annual Conference*, MIT Press, Stanford University, CA, USA, pp. 230–237.

[35] Whigham, P. A. & Science, D. O. C. [1995]. Inductive bias and genetic programming, *In Proceedings of First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 461–466.

[36] Wineberg, M. & Oppacher, F. [1994]. A representation scheme to perform program induction in a canonical genetic algorithm, *Parallel Problem Solving from Nature III*, Vol. 866 of *LNCS*, Springer-Verlag, Jerusalem, pp. 292–301.

[37] Yanai, K. & Iba, H. [2003]. Estimation of distribution programming based on Bayesian network, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, IEEE Press, Canberra, pp. 1618–1625.

[38] Yanai, K. & Iba, H. [2005]. Probabilistic distribution models for EDA-based GP, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, Vol. 2, ACM Press, Washington DC, USA, pp. 1775–1776.

[39] Yanase, T., Hasegawa, Y. & Iba, H. [2009]. Binary encoding for prototype tree of probabilistic model building gp, *Proceedings of 2009 Genetic and Evolutionary Computation Conference (GECCO 2009)*, pp. 1147–1154.