Estimation of Distribution Algorithm based on Probabilistic Grammar with Latent Annotations

Yoshihiko Hasegawa† and Hitoshi Iba†

Abstract— **Genetic Programming (GP) which mimics the natural evolution to optimize functions and programs, has been applied to many problems. In recent years, evolutionary algorithms are seen from the viewpoint of the estimation of distribution. Many algorithms called EDAs (Estimation of Distribution Algorithms) based on probabilistic techniques have been proposed. Although probabilistic context free grammar (PCFG) is often used for the function and program evolution, it assumes the independence among the production rules. With this simple PCFG, it is not able to induce the building-blocks from promising solutions. We have proposed a new function evolution algorithm based on PCFG using latent annotations which weaken the independence assumption. Computational experiments on two subjects (the royal tree problem and the DMAX problem) demonstrate that our new approach is highly effective compared to prior approaches.**

I. Introduction

In this paper, we propose a function optimization algorithm based on probabilistic context free grammar (PCFG) with latent annotations (PCFG-LA) [12]. Our approach named PAGE (Programming with Annotated Grammar Estimation) uses latent annotations which enable an induction of promising sub-functions (sub-routines).

GP (Genetic Programming) [7], [8] is an extended algorithm of GA (Genetic Algorithm) and is capable of handling programs and functions. Because GP has structural chromosomes which are very flexible, GP has been applied to many problems (robot engineering, financial engineering, bio informatics, etc) and is considered to be a very powerful way for solving these problems.

GA and GP evolve solution candidates with crossover and mutation operators. These methods are considered to be highly effective because they take advantage of the mechanism of natural evolution. Recently, Evolutionary Algorithms are seen from the viewpoint of the distribution estimation. A new way for evolving solutions based on the estimation of distribution has been proposed. More and more attention has been paid to this method, which is called EDA (Estimation of Distribution Algorithm) [10], [14]. In EDAs, there are two types of algorithms : those based on GA style linear chromosome (GA-EDA) and those based on GP style tree chromosome (GP-EDA).

In GP-EDA, algorithms can be broadly classified into two groups: proto-type tree based method and PCFG based method. The former method translates variable length tree-structures into fixed length structures (mostly linear arrays). The later method uses context free grammar (CFG) for expressing programs and functions. Algorithms based on PCFG are considered to be well suited for expressing functions in GP. Thus many algorithms based on the later approach have been proposed (Section II).

The basic PCFG adopts the context freedom assumption that the probabilities of production rules do not depend on the ascendant nodes or sibling nodes. The basic GP-EDA based on PCFG just estimates parameters (production rule probability) and generate the programs using the parameters. Although the independence assumption makes the statistical estimation easier, this PCFG basically can not take into account the interactions among nodes. Thus it is not suitable for grasping the sub-functions or sub-routines in GP programs, because GP functions often have strong interactions among nodes. In order to weaken the independence assumption in PCFG, annotations have been proposed in the natural language. For example, vertical markovization annotates the symbols with their ancestor symbols. Prior GP-EDAs such as PEEL or Grammar Transformation in EDA (GT-EDA) used annotated PCFG for base line grammars. In these algorithms, the depth of the production rules is taken into account. Although these annotations may work in specific problems, this assumption may be too strong or wrong in other problems.

Matsuzaki et. al. proposed the PCFG with latent annotations (PCFG-LA) [12] which assumes that the annotations are hidden and the production rules with annotations can be estimated with EM algorithm. We expect that PCFG-LA is suitable for more precisely grasping the interactions in GP compared to previous methods and it is more flexible compared to the method based on heuristics based annotations as *depth*.

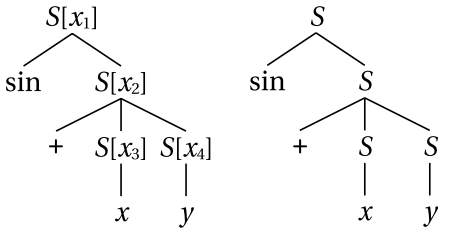
This paper is structured as follows. In the next section, we briefly introduce the program evolution methods based on probabilistic techniques. In Section III we explain the PCFG-LA which is used in our proposal describing parameters update formula. In section V, we compare the performance of our method to other methods, selecting two benchmark tests for experiments. We discuss the results obtained in the experiment in section VI. Finally, we conclude this paper in section VII.

II. Related Works

Probabilistic evolutionary algorithms based on probabilistic models have been first carried in GA style linear chromosome. Because GP uses tree structures which are more complex than linear arrays used in GA, GP-EDA is a newer

†Department of Frontier Informatics, Graduate School of Frontier Sciences, The University of Tokyo (email: yoshihiko.hasegawa@gmail.com, iba@iba.k.u-tokyo.ac.jp)

1-4244-1340-0/07$25.00



(a) Complete tree (b) Observed tree

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

field than GA-EDA. Many GP-EDAs have been proposed up to now and can be broadly classified into two groups: (i) prototype tree based method and (ii) grammar model based methods.

The method (i) takes advantage of the techniques devised in GA-EDA. This type of algorithms translates tree structures into fixed length chromosomes used in GA and applies the probabilistic model. PIPE (Probabilistic Incremental Program Evolution) [19] adopted the univariate model which can be considered as the combination of PBIL (Population Based Incremental Learning) [1] and GP. EDP (Estimation of Distribution Programming) [25], [26] considered the parent-children relationship in the tree structure. EDP considers conditional probability distribution of children given parent nodes. ECGP (Extended Compact GP) [20] combines the ECGA (Extended Compact GA) [4] with GP to take into account the relationship among nodes. This algorithm estimates the group of joint distribution with MDL principle. BOAP extends BOA (Bayesian Optimization Algorithm) [15] to be able to handle programs and functions. BOAP (BOA Programming) [11] uses *zig-zag* tree which is based on lambda function. POLE (Program Optimization with Linkage Estimation) [5], [6] estimated the interactions among nodes with estimating the Bayesian network. In this algorithm, special chromosome called *expanded parse tree* is used to convert the GP programs into linear arrays.

The method (ii) which is based on PCFG has a lot to do with GGGP (Grammar Guided Genetic Programming) [24]. GGGP applies the CFG to GP. SG-GP (Stochastic Grammar based GP) [17] uses simple PCFG. SG-GP integrates PBIL with GGGP. In this algorithm, the parameters are updated using an incremental learning strategy. PEEL (Program Evolution with Explicit Learning) [21] is an extended algorithm of SG-GP which takes into account the depth of symbols. GMPE (Grammar Model based Program Evolution) [22] merge the production rules to find the general production rules. GMPE starts from specialized production rules and merge non-terminals to be more general production rules with MDL principle. Grammar Transformation in EDA (GT-EDA) [2] takes advantage of MDL principle to estimate expanded production rules (tree structure) to extract good subroutines. BAP (Bayesian Automatic Programming) [18] uses Bayesian network to consider the relation among production rules.

Unlike PEEL and GT-EDA which use fixed annotations, our proposal uses annotations which are induced from promising solutions using EM algorithm. Our method is more flexible than the fixed annotation model and we expect that our method is more effective than the methods based on fixed annotations. In GA-EDA, an algorithm taking advantage of latent variables have been proposed [23]. In this work, EM algorithm is also used to estimated the parameters.

III. PCFG WITH LATENT ANNOTATIONS

In CFG, many approaches have been proposed to weaken the context freedom assumption by annotating non-terminal symbols with many features. Many of the works use fixed annotations as labels of ancestor symbols, and sibling nodes. Matsuzaki et. al. [12] proposed the probabilistic context free grammar with latent annotations (PCFG-LA) which automatically induces the production rules with latent annotations. They also derived the parameter update formula with EM algorithm.

In PCFG-LA, every non-terminal is labeled with annotations. In the complete form, non-terminals are represented $A[x]$, where A is the non-terminal symbol and x is an annotation which is not observed ($x \in H$, where H is a set of annotations). Figure 1 is an example of trees with annotations (a) and observed tree (b). The probability of the annotated tree can be represented with Equation 1.

$$P(T, \mathbf{X}; \Theta) = \pi(\mathcal{S}[x_1]) \prod_{r \in D_{T[\mathbf{X}]}} \beta(r) \tag{1}$$

In this equation, T is a derivation tree, x_i is an annotation of ith non-terminal (all the non-terminals are numbered from the root), $\mathbf{X}$ is $\mathbf{X} = \{x_1, x_2, ...\}$, $\pi(\mathcal{S}[x])$ is a probability of $\mathcal{S}[x]$ at the root position, $\beta(r)$ is a probability of annotated production rule r, $D_{T[X]}$ is a multi-set of used annotated rules in tree T and Θ denotes a set of parameters $\Theta = \{\pi, \beta\}$.

The probability of a observed tree can be calculated with summing over annotations given by

$$P(T; \Theta) = \sum_{\mathbf{X}} P(T, \mathbf{X}; \Theta). \tag{2}$$

Because annotations are not observed, the parameters (π and β) have to be estimated by EM algorithm. The difference of log-likelihood between parameters Θ' and Θ can be calculated by Equation 3.

$$\begin{aligned}
&\log P(T; \Theta') - \log P(T; \Theta) \\
&= \log \frac{P(T; \Theta')}{P(T; \Theta)} \\
&= \sum_{\mathbf{X}} P(\mathbf{X}|T; \Theta) \log \frac{P(T, \mathbf{X}; \Theta')}{P(T, \mathbf{X}; \Theta)} \frac{P(\mathbf{X}|T; \Theta)}{P(\mathbf{X}|T; \Theta')} \\
&\geq \sum_{\mathbf{X}} P(\mathbf{X}|T; \Theta) \log \frac{P(T, \mathbf{X}; \Theta')}{P(T, \mathbf{X}; \Theta)}
\end{aligned} \tag{3}$$

Using Equation 3, the update formula can be obtained by optimizing $Q(\Theta'|\Theta)$ (Equation 4).

$$Q(\Theta'|\Theta) = \sum_{T_i \in \mathbf{T}} \sum_{\mathbf{X}_i} P(\mathbf{X}_i|T_i;\Theta) \log P(T_i, \mathbf{X}_i;\Theta') \quad (4)$$

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

$$\mathcal{S} \rightarrow g\,\mathcal{S}\,...\mathcal{S}, \quad (5)$$

which is a subset of GNF (Greibach Normal Form). Here $\mathcal{S} \in \mathcal{N}$ and $g \in \mathcal{T}$ ($\mathcal{N}$ and $\mathcal{T}$ are sets of non-terminal and terminal symbols in CFG, respectively). g, terminal symbol 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. In the right-side of Equation 5, the number of $\mathcal{S}$ is identical to the arity of g. If $g = +$, then $\mathcal{S} \rightarrow +\,\mathcal{S}\,\mathcal{S}$ and if $g = x$, then $\mathcal{S} \rightarrow x$. Annotated production rules can be expressed by Equation 6,

$$\mathcal{S}[x] \rightarrow g\,\mathcal{S}[z_1]\,...\mathcal{S}[z_a] \quad (6)$$

where $x, z_m \in H$, a is the arity of g in GP. However, we made an assumption in our algorithm because of the following reason. Let us h be the size of annotation H. If g has a arity, the number of parameters for the production rule $\mathcal{S} \rightarrow g\,\mathcal{S}...\mathcal{S}$ with annotations becomes h^{a+1}. Estimating the large number of parameters may cause an over fitting problem and requires much more computational power. In order to reduce the number of parameters, we assume that all right-side non-terminal symbols have the same annotation.

$$\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]\,\mathcal{S}[y]...\mathcal{S}[y] \quad (7)$$

With this assumption, the size of parameters can be reduced to h^2, which is tractable. Because there are many functions which are commutative (e.g. $+$, $\times$, etc) in GP, we think that this assumption may not be so strong. From now on, we assume Equation 7 in the following explanation. In the following sections, let $\mathcal{R}[H]$ be the set of annotated rules expressed by Equation 7.

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

A. Forward-backward probability

For calculating the Equation 4, forward-backward probability is used. Backward probability $b^i_T(x)$ stands for the probability that the tree beneath ith non-terminal $\mathcal{S}[x]$ is generated. Forward probability $f^i_T(y)$ stands for the probability that the tree above ith non-terminal $\mathcal{S}[y]$ is generated . These probabilities can be recursively calculated (Equation 8, 9 and 10).

$$b^i_T(x) = \sum_{y \in H} \beta(\mathcal{S}[x] \rightarrow g_i\,\mathcal{S}[y]...\mathcal{S}[y]) \prod_{j \in ch(i,T)} b^j_T(y) \quad (8)$$

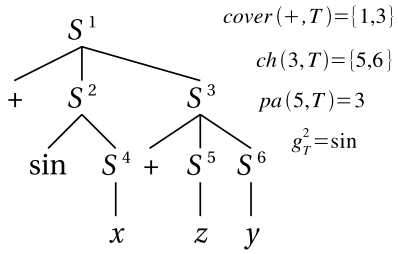


Fig. 2. An example of the derivation tree and values of the specific functions. Superscriptions denote the indices of non-terminals.

$$f^i_T(y) = \sum_{x \in H} f^{pa(i,T)}_T(x)\beta(\mathcal{S}[x] \rightarrow g^{pa(i,T)}_T\,\mathcal{S}[y]...\mathcal{S}[y]) \times \prod_{j \in ch(pa(i,T),T), j \neq i} b^j_T(y) \quad (i \neq 1) \quad (9)$$

$$f^i_T(y) = \pi(\mathcal{S}[y]) \quad (i = 1) \quad (10)$$

In these equations, $ch(i,T)$ is a function which returns the set of non-terminal children indices of ith non-terminal in T. $pa(i,T)$ returns a parent index of ith non-terminal in T. g^i_T is a terminal symbol in CFG and is connected to ith non-terminal symbol in T. For example, $ch(3,T) = \{5,6\}$, $pa(5,T) = 3$ and $g^2_T = \sin$ in Figure 2.

Using the forward-backward probability, $P(T;\Theta)$ can be represented by following Equations.

$$P(T;\Theta) = \sum_{x \in H} \pi(\mathcal{S}[x]) b^1_T(x) \quad (11)$$

$$P(T;\Theta) = \sum_{x,y \in H} \beta(\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y]) f^i_T(x) \times \prod_{j \in ch(i,T)} b^j_T(y) \quad (i \in cover(g,T)) \quad (12)$$

In this equation, $cover(g, T_i)$ represents a function which 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 a tree represented in Figure 2, then $cover(+,T) = \{1,3\}$.

B. Parameter update formula

Using the forward-backward probability and optimizing $Q(\Theta'|\Theta)$, the update formula can be derived. Because our approach is not based on CNF and we limit the right-side annotations in production rules, the update formula is different from that in the original paper (see Appendix).

$$\pi'(\mathcal{S}[x]) \propto \pi(\mathcal{S}[x]) \sum_{T_i \in \mathbf{T}} \frac{b^1_{T_i}(x)}{P(T_i)} \quad (13)$$

$$\beta'(\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y]) \propto \beta(\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y]) \times \sum_{T_i \in \mathbf{T}} \frac{1}{P(T_i)} \sum_{j \in cover(g,T_i)} f^j_{T_i}(x) \prod_{k \in ch(j,T_i)} b^k_{T_i}(y) \quad (14)$$

EM algorithm maximizes the log-likelihood ($L(\Theta;\mathbf{T}) = \log P(\mathbf{T};\Theta) = \sum_{T_i \in \mathbf{T}} \log P(T_i|\Theta)$) monotonically from the initial parameters. For initial parameters, we used the values represented with Equation 15 and 16.

$$\beta(\mathcal{S}[x] \to g\,\mathcal{S}[y]...\mathcal{S}[y]) \propto \frac{e^{-\kappa}}{h-1}\gamma(\mathcal{S} \to g\,\mathcal{S}\,...\mathcal{S}) \quad (15)$$

$$\beta(\mathcal{S}[x] \to g) \propto e^{-\kappa}\gamma(\mathcal{S}[x] \to g) \quad (16)$$

In Equation 15, κ is a random value which is uniformly distributed over $[-\log 3, \log 3]$ and $\gamma(\mathcal{S} \to g\,\mathcal{S}....\mathcal{S})$ is a probability of observed production rule (without annotations). We also set $\beta(\mathcal{S}[x] \to g\,\mathcal{S}[x]...\mathcal{S}[x]) = 0$.

Because the obtained parameters are greatly affected by initial values, we carried EM algorithm several times (in this paper, 3 times at each generation) with different initial parameters and then adopted the parameters which gave the best likelihood.

IV. Flowchart

Combining the PCFG-LA with GP, we propose a new algorithm PAGE (Programming with Annotated Grammar Estimation). In this section, a flowchart of PAGE is shown.

1) **Initialization of individuals**
   Individuals are generated randomly according to the initial parameters. The number of initialized individuals is represented with M.
2) **Evaluation of individuals**
   All individuals are assigned fitness values according to the evaluation function.
3) **Selection of individuals**
   Promising individuals are selected which are used for the parameter estimation. In our implementation, a truncate selection is used. The number of individuals to select is represented by $M \times P_s$.
4) **Estimation of parameters**
   Parameters are estimated with EM algorithm explained in the proceeding section. Parameters are estimated at every generation. We stop updating the parameters if the log-likelihood improvement is smaller than 0.5%.
5) **Generation of new individuals**
   New individuals are generated with estimated parameters. If we want to use an elitist strategy, the best $M \times P_e$ individuals are copied from the previous generation.

Steps from (2) to (5) are repeated until termination criteria are met.

V. Comparative Experiments

We carried comparative experiments to show the effectiveness of our approach (PAGE). We compared the search performance among the algorithms listed below.

- **PAGE**
  This is our proposed method. This method extends PCFG-GP by introducing the latent annotations. The number of latent annotations $h(h > 1)$ is given beforehand. The annotations in PAGE are represented by integer ($H = \{0, 1, ..., h-1\}$).
- **PCFG-GP**
  We denote PCFG-GP for expressing a GP based on a simple PCFG. This algorithm does not take annotations into account. This is $h = 1$ case of PAGE, and it is basically identical to the GP-EDA which uses simple PCFG for probabilistic model (SG-GP [17]).
- **Simple GP (SGP)**
  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) are used. Crossover points are selected in the following way: When applying crossover to two individuals, we select the first crossover point from function nodes at the 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.

TABLE I
Main parameters for PAGE.

Variable	Meaning	Value
M	Population size	–
G	Generations to run	–
h	The number of annotations	–
P_s	Selection rate	0.1
P_e	Elite rate	0.1

A. Royal Tree Problem

We applied PAGE to the royal tree problem to compare the performance between PAGE and PCFG-GP. This experiment shows the effectiveness of latent annotations.

1) Problem Description: In order to show the effectiveness of latent annotations, we compared the search performance in the royal tree problem [16]. The royal tree problem is an extension of the royal road function[13] which has been the famous benchmark test in GA. The royal tree problem uses a set of functions which are defined with alphabetical symbols $\mathfrak{F} = \{a, b, c, d, ...\}$ which have increasing arity (a has 1 arity, b has 2 arity and so on) and terminal nodes $\mathfrak{T} = \{x\}$ ($\mathfrak{F}$ and $\mathfrak{T}$ represents sets of GP functions and GP terminals, respectively). The royal tree problem defines the state *perfect tree* in each level. Perfect tree of some level is composed of the perfect tree whose level is smaller by 1 level than the level. 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. For more details, see [16].

In our setups, we used level d royal tree problem. The production rules for the royal tree problem are shown in the list below, where symbols with small case denote terminal symbols in CFG. $a \sim d$ are function nodes and x is a terminal node in GP.

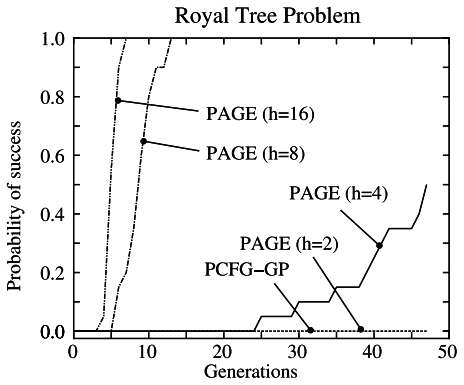


Fig. 3. The probability of success in the royal tree problem. The number attached to PAGE in the parentheses represents the annotation size h.

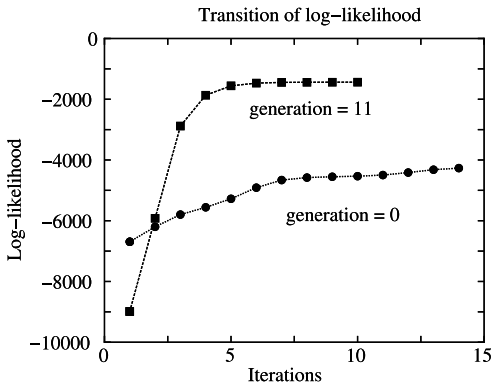


Fig. 4. Transition of log-likelihood in the royal tree problem using PAGE. We show the transition at generation = 0 and 11.

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

In this experiment, $M = 1000$ is used in both PAGE and PCFG-GP. We observe the behavior of our algorithm in terms of the number of latent annotations (h) with $h = 2, 4, 8, 16$ in PAGE.

2) Results: Figure 3 represents the probability of success at each generation. As can been seen in Figure 3, PCFG-GP did not obtain the optimum at all. This reason can be easily estimated. Table II (lower table) describes the obtained production rule probabilities of PCFG-GP. As can been seen with this table, probability of production rule $\mathcal{S} \rightarrow x$ is the highest. Thus using simple PCFG, the size of generated individuals tends to be small, because probabilities of large trees tends to be very small. On the other hand, our approach uses the production rules with latent annotations. In Table II (upper table), the production rules with annotations are described ($h = 8$). In this table, because of space limitation, we omitted the production rules which have very small probabilities (smaller than 0.01). In the probabilities with annotations, the annotations are used for generating different symbols. For example, $\mathcal{S}[0]$ and $\mathcal{S}[3]$ generate x and $\mathcal{S}[1]$ generates d exclusively. Using these production rules, we can grasp the sub-structures with higher probability.

In PAGE, the number of annotations h strongly affect the search performance. In Figure 3, we can see that the larger h gives the better performance. In the case of smaller annotation size, there are many overlapping symbols. For example in Table III, we can see $\mathcal{S}[2]$ mainly generates $a\,\mathcal{S}[0]$ and $a\,\mathcal{S}[3]$. However, $\mathcal{S}[2]$ also generates $b\,\mathcal{S}[3]\,\mathcal{S}[3]$ and $c\,\mathcal{S}[3]\,\mathcal{S}[3]\,\mathcal{S}[3]$. This overlap makes the probability $P(T;\Theta)$ of the optimum smaller and it is more difficult to find the optimum. On the other hand, in the case of large h, there are many redundant symbols. Thus overlappingly used symbol as $\mathcal{S}[2]$ is less in this case.

TABLE II

OBTAINED PROBABILITIES OF PAGE ($h = 8$) AND PCFG-GP. WE SELECTED THE RULE WITH THE HIGHEST PROBABILITIES FROM A START SYMBOL (GRAY RULES).

Production rule (PAGE)	Probability (π, β)
$\mathcal{S}[1]$	1.0
$\mathcal{S}[0] \rightarrow x$	1.0
$\mathcal{S}[1] \rightarrow d\,\mathcal{S}[7]\,\mathcal{S}[7]\,\mathcal{S}[7]\,\mathcal{S}[7]$	1.0
$\mathcal{S}[2] \rightarrow a\,\mathcal{S}[0]$	0.305
$\mathcal{S}[2] \rightarrow a\,\mathcal{S}[3]$	0.395
$\mathcal{S}[2] \rightarrow a\,\mathcal{S}[4]$	0.0744
$\mathcal{S}[2] \rightarrow b\,\mathcal{S}[3]\,\mathcal{S}[3]$	0.0306
$\mathcal{S}[2] \rightarrow c\,\mathcal{S}[3]\,\mathcal{S}[3]\,\mathcal{S}[3]$	0.0865
$\mathcal{S}[2] \rightarrow d\,\mathcal{S}[3]\,\mathcal{S}[3]\,\mathcal{S}[3]\,\mathcal{S}[3]$	0.0421
$\mathcal{S}[2] \rightarrow x$	0.0393
$\mathcal{S}[3] \rightarrow x$	1.0
$\mathcal{S}[4] \rightarrow x$	1.0
$\mathcal{S}[5] \rightarrow a\,\mathcal{S}[0]$	0.393
$\mathcal{S}[5] \rightarrow a\,\mathcal{S}[3]$	0.223
$\mathcal{S}[5] \rightarrow a\,\mathcal{S}[4]$	0.363
$\mathcal{S}[6] \rightarrow b\,\mathcal{S}[2]\,\mathcal{S}[2]$	0.352
$\mathcal{S}[6] \rightarrow b\,\mathcal{S}[5]\,\mathcal{S}[5]$	0.596
$\mathcal{S}[6] \rightarrow d\,\mathcal{S}[2]\,\mathcal{S}[2]\,\mathcal{S}[2]\,\mathcal{S}[2]$	0.0230
$\mathcal{S}[6] \rightarrow x$	0.0117
$\mathcal{S}[7] \rightarrow c\,\mathcal{S}[6]\,\mathcal{S}[6]\,\mathcal{S}[6]$	0.998

Production rule (PCFG-GP)	Probability(π, β)
$\mathcal{S}$	1.0
$\mathcal{S} \rightarrow x$	0.60
$\mathcal{S} \rightarrow a\,\mathcal{S}$	0.10
$\mathcal{S} \rightarrow b\,\mathcal{S}\,\mathcal{S}$	0.10
$\mathcal{S} \rightarrow c\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}$	0.10
$\mathcal{S} \rightarrow d\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}\,\mathcal{S}$	0.09

PAGE estimates the parameters with EM algorithm which monotonically increases the likelihood. Figure 4 describes the increase of log-likelihood ($\sum_{T_i \in \mathbf{T}} \log P(T_i;\Theta)$) in the royal tree problem using PAGE. We show the transition of generation 0 and generation 11. As can been seen with this figure, parameters are converged around by 10 iterations. The log-likelihood improvement at generation 11 is larger than that at generation 0, because the tree structures are converged at the end of the search.

B. DMAX problem

1) Problem Description: We applied our approach to DMAX problem [5] to show the superiority of our proposal over Simple GP. DMAX problem is an extended problem of the MAX problem. Because the original MAX problem does not have deceptiveness, it is very easy to solve by GP-EDAs without considering the interactions. We extended the MAX problem by adding the deceptiveness and we call this extended problem *deceptive MAX problem* (DMAX problem). We employed the function $\mathfrak{F}$ and terminal $\mathfrak{T}$ given by

$$\begin{aligned} \mathfrak{F} &= \{+_m, \times_m\} \\ \mathfrak{T} &= \{\lambda, 0.95\} \end{aligned} \tag{17}$$

with

$$\lambda^r = 1, \lambda \in \mathbb{C}, r \in \mathbb{N} \tag{18}$$

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

The main objective of the DMAX problem is identical to the original one: to find the functions which return the largest *real* value under the limitation on maximum tree depth D. However, the symbols used in the DMAX problem are different from those used in the MAX problem. The DMAX problem uses the symbols represented with Equation 17. A fitness value of individual is the real part of its function. If the value of a function is $a + bi$ (where $a, b \in \mathbb{R}$,$i = \sqrt{-1}$), then its fitness value is a. The DMAX problem can be represented with 3 parameters, m (arity), r (power) and D (maximum tree depth).

In this problem, the optimum value is more complicated than that of the MAX problem [3], [9] which does not have the deceptiveness. We explain the optimum with an intuitive example. We use $m = 5, r = 3$, for example, because DMAX problem has very strong deceptiveness with this setup. In the comparative experiment, we used $D = 4$, but for simplification, we explain the case with $D = 3$ (Figure 5). In this parameter setup, $\lambda = \cos\frac{2\pi}{3} + i\sin\frac{2\pi}{3}$. First, we add λ with $+_5$ to make 5λ. Then we multiply this value with $\times_5$ which leads to $(5\lambda)^5 = 5^5\lambda^5 = 5^5\lambda^2$. However, $Re(5^5\lambda^2)$ is negative value and is not a good solution. So 2 values out of 5 values multiplied have to be real values. 5×0.95 is used as a substitution for 5λ and the maximum value is $(5\lambda)^3(0.95 \times 5)^2 = 2820.3125$. In $D = 4$, the maximum value is $(5\lambda)^{24}(0.95 \times 5) = 2.83 \times 10^{17}$. Figure 6 shows a visualization of the values in the complex plane whose horizontal (real) axis expresses a fitness value. As can been seen, the replacement of one 0.95×5 with 5λ increases the argument by 120 degrees in the complex plane. After three times of replacement, the value returns to the same phase but has a larger absolute value.

The production rules employed in PAGE and PCFG-GP are represented by following equations.

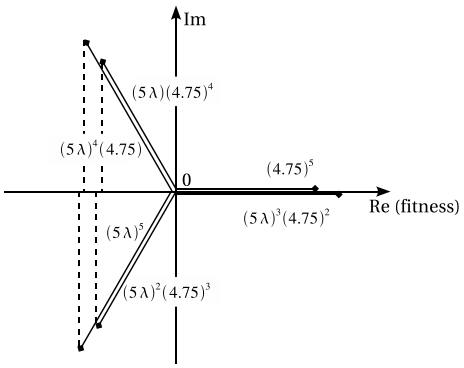


Fig. 5. The optimum of the DMAX problem in the complex plane (see text).

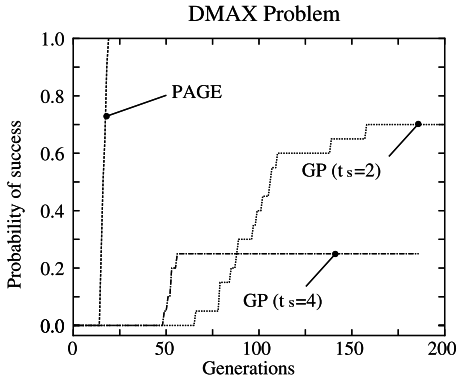


Fig. 6. Probability of success in the DMAX problem with $M = 3000$ PAGE and Simple GP . The result of PCFG-GP is not presented because it could not obtain optimum even at once.

$$\begin{aligned} \mathcal{S} &\rightarrow +_5 \mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S} \\ \mathcal{S} &\rightarrow \times_5 \mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S}\mathcal{S} \\ \mathcal{S} &\rightarrow 0.95 \\ \mathcal{S} &\rightarrow \lambda \end{aligned}$$

For the DMAX problem of $D = 4$, $m = 5$ and $r = 3$, we used $M = 3000$ in PAGE, PCFG-GP and SGP. The number of latent annotations h is set to 8. The tournament size of SGP is $t_s = 2$, $t_s = 4$. We run each algorithm 20 times and observed the probability of success at each generation. We stop every algorithm when they obtain the optimum. Furthermore, we stop PAGE and PCFG-GP when the best fitness is not improved for 10 consecutive generations and stop SGP when the best fitness is not improved for generation g to $2g$ ($g > 10$).

2) Result: We show the probability of success in DMAX problem in Figure 6. Because PCFG-GP could not find the optimum even at once, we omit the result of PCFG-GP. In DMAX problem, two building-blocks have to be used to obtain the optimum: the one composed of five 0.95 and

TABLE III

ESTIMATED PROBABILITIES IN PAGE ($h = 8$). BECAUSE OF SPACE LIMITATION, RULES WITH PROBABILITIES SMALLER THAN 0.01 ARE OMITTED.

Production Rule (PAGE)	Probability (π, β)
$\mathcal{S}[2]$	0.693
$\mathcal{S}[4]$	0.132
$\mathcal{S}[5]$	0.174
$\mathcal{S}[0] \rightarrow \times_5\, \mathcal{S}[6]\, \mathcal{S}[6]\, \mathcal{S}[6]\, \mathcal{S}[6]\, \mathcal{S}[6]$	1.0
$\mathcal{S}[1] \rightarrow \lambda$	0.0218
$\mathcal{S}[1] \rightarrow 0.95$	0.978
$\mathcal{S}[2] \rightarrow \times_5\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]$	1.0
$\mathcal{S}[3] \rightarrow \lambda$	0.972
$\mathcal{S}[3] \rightarrow 0.95$	0.0277
$\mathcal{S}[4] \rightarrow \times_5\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]$	1.0
$\mathcal{S}[5] \rightarrow \times_5\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]\, \mathcal{S}[0]$	1.0
$\mathcal{S}[6] \rightarrow +_5\, \mathcal{S}[1]\, \mathcal{S}[1]\, \mathcal{S}[1]\, \mathcal{S}[1]\, \mathcal{S}[1]$	0.300
$\mathcal{S}[6] \rightarrow +_5\, \mathcal{S}[3]\, \mathcal{S}[3]\, \mathcal{S}[3]\, \mathcal{S}[3]\, \mathcal{S}[3]$	0.260
$\mathcal{S}[6] \rightarrow +_5\, \mathcal{S}[7]\, \mathcal{S}[7]\, \mathcal{S}[7]\, \mathcal{S}[7]\, \mathcal{S}[7]$	0.440
$\mathcal{S}[7] \rightarrow \lambda$	0.986
$\mathcal{S}[7] \rightarrow 0.95$	0.0136

one composed of five λ. Production rules which generate 0.95 and λ have the same left-side symbol ($\mathcal{S} \rightarrow 0.95$ and $\mathcal{S} \rightarrow \lambda$). Because PCFG-GP does not distinguish two symbols, structures composed of mixture of 0.95 and λ are generated (e.g. $\lambda + \lambda + 0.95 + \lambda + 0.95$). Thus the algorithm without annotations failed to obtain the optimum. On the other hand, in PAGE, 0.95 and λ are generated from different production rules. Table III shows the estimated production rules by PAGE ($h = 8$). As can been seen with Table III, λ and 0.95 are generated from different symbols. $\mathcal{S}[0]$, $\mathcal{S}[3]$, $\mathcal{S}[4]$, $\mathcal{S}[5]$ generate $\lambda$and $\mathcal{S}[7]$ generates 0.95. Thus the substructures which are mixtures of λ and 0.95 are not generated.

As can been seen with Figure 6, PAGE shows the much better performance compared to those of GPs ($t_s = 2$ and $t_s = 4$). As shown in the paper [5], DMAX problem has very strong deceptiveness when using the crossover in GP especially when using higher selection pressure. DMAX problem takes advantage of the defect of a crossover operator. Our approach effectively estimates the parameters with latent annotations, the deceptiveness of DMAX problem can be overcome.

VI. DISCUSSION

We have employed the PCFG with latent annotations which weakens the independence assumption. Because the statistical estimation in PAGE uses EM algorithm and is a much more complex way compared to simple PCFG, our approach is more computationally expensive. However, the performance of PAGE is much more superior to the algorithm not including the annotations. We can say that PAGE is a stronger algorithm than simple PCFG-GP as a whole.

PAGE has extra parameters and the number of annotations has to be given in advance. The experiment of the royal tree problem showed that the larger h gave the better performance. However, an estimation of larger h requires more computational time. We have to optimize these two contradicting factors which will be examined in the future work. Because EM algorithm is basically hill-climbing strategy, the optimized parameters are greatly affected by the initial parameters. In this paper, we tried EM algorithm 3 times at each generation and adopted the set of parameters which offered the best log-likelihood. More sophisticated way to optimize parameters may be used.

VII. CONCLUSION

We proposed the probabilistic program evolution algorithm named PAGE. Our proposal takes advantage of latent annotations which weaken the context freedom assumption in CFG. With two computational experiments, we confirmed that the latent annotations are highly effective for grasping the building-blocks. In the royal tree problem, we showed that the number of annotations greatly affects the search performance and larger annotation size offered better performance. The result of DMAX problem showed that PAGE is highly effective for the problem with strong deceptiveness. We hope that POLE is applied to a wide class of real world problems, which is our future subject.

REFERENCES

[1] Shumeet Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Pittsburgh, PA, 1994.

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

[3] Chris Gathercole and Peter Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 291–296, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

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

[5] Yoshihiko Hasegawa and Hitoshi Iba. Estimation of Bayesian Network for Program Generation. In *Proceedings of The Third Asian-Pacific Workshop on Genetic Programming*, pages 35–46, Hanoi, Vietnam, 2006.

[6] Yoshihiko Hasegawa and Hitoshi Iba. Optimizing Programs with Estimation of Bayesian Network. In *Proceedings of the 2006 World Congress on Computational Intelligence*, pages 5527–5534, Vancouver, 2006. IEEE press.

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

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

[9] W. B. Langdon and R. Poli. An analysis of the MAX problem in genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 222–230, Stanford University, CA, USA, 13-16 1997. Morgan Kaufmann.

[10] Pedro Larrañaga and Jose A. Lozano. *Estimation of Distribution Algorithms*. Kluwer Academic Publishers, 2001.

[11] Mosche Looks. Learning computer programs with the Bayesian optimization algorithm, 2005. Master thesis, Washington University Sever Institute of Technology.

[12] T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilistic CFG with latent annotations. In *In Proceedings of the 43rd Metting of the Association for Computational Linguistics (ACL)*, pages 75–82, 2005.

[13] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In Francisco J. Varela and Paul Bourgine, editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1991*, pages 245–254, Paris, 11–13 1992. A Bradford book, The MIT Press.

[14] H. Mühlenbein and T. Mahnig. The Factorized Dsitribution Algorithm for additively decomposed functions. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 752–759. IEEE press, 1999.

[15] Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. BOA: The Bayesian optimization algorithm. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume I, pages 525–532, Orlando, FL, 13-17 1999. Morgan Kaufmann Publishers, San Fransisco, CA.

[16] William F. Punch. How effective are multiple populations in genetic programming. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 308–313, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[17] Alain Ratle and Michele Sebag. Avoiding the bloat with probabilistic grammar-guided genetic programming. In P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001*, volume 2310 of *LNCS*, pages 255–266, Creusot, France, October 29-31 2001. Springer Verlag.

[18] Evandro Nunes Regolin and Aurora Trindad Ramirez Pozo. Bayesian automatic programming. In Maarten Keijzer, Andrea Tettamanzi, Pierre Collet, Jano I. van Hemert, and Marco Tomassini, editors, *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of *Lecture Notes in Computer Science*, pages 38–49, Lausanne, Switzerland, 30 March - 1 April 2005. Springer.

[19] R. P. Salustowicz and J. Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141, 1997.

[20] Kumara Sastry and David E. Goldberg. Probabilistic model building and competent genetic programming. In Rick L. Riolo and Bill Worzel, editors, *Genetic Programming Theory and Practise*, chapter 13, pages 205–220. Kluwer, 2003.

[21] Y. Shan, R. I. McKay, H. A. Abbass, and D. Essam. Program evolution with explicit learning: a new framework for program automatic synthesis. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 1639–1646, Canberra, 8-12 December 2003. IEEE Press.

[22] Yin Shan, Robert I. McKay, Rohan Baxter, Hussein Abbass, Daryl Essam, and Nguyen Xuan Hoai. Grammar model-based program evolution. In G. Greenwood, editor, *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 478–485, Portland, Oregon, 20-23 June 2004. IEEE Press.

[23] Dong-Yeon Cho Soo-Yong Shin and Byoung-Tak Zhang. Function optimization with latent variable models. In *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)*, pages 145–152, Havana, Cuba, March 2001.

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

[25] Kohsuke Yanai and Hitoshi Iba. Estimation of distribution programming based on Bayesian network. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 1618–1625, Canberra, 8-12 December 2003. IEEE Press.

[26] Kohsuke Yanai and Hitoshi Iba. Probabilistic distribution models for EDA-based GP. In H.-G. Beyer and et al., editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1775–1776, Washington DC, USA, 25-29 June 2005. ACM Press.

APPENDIX

In Appendix, we show the derivation of Equation 13 and 14. Using Equation 1, $Q(\Theta'|\Theta)$ can be rewritten into Equation 19.

$$Q(\Theta'|\Theta) = \sum_{T_i \in \mathbf{T}} \frac{1}{P(T_i;\Theta)} \sum_{\mathbf{X}_i} P(T_i, \mathbf{X}_i; \Theta) \times \left\{ \log \pi'(\mathcal{S}[x_1^i]) + \log \prod_{r \in D_{T_i[X_i]}} \beta'(r) \right\} \quad (19)$$

In this equation, x_j^i is an annotation of a jth node in an ith tree. Equation 19 consists of two terms ($\pi'(\mathcal{S}[x])$ and $\beta'(r)$) and both parts can be minimized independently. Let the left and right part be Q_π and Q_β, respectively.

We only show the derivation of β', because π' can be derived in the similar and easier way. By using Equation 12, Q_β can be represented by Equation 20.

$$Q_\beta = \sum_{T_i \in \mathbf{T}} \frac{1}{P(T_i;\Theta)} \sum_{\mathbf{X}_i} P(\mathbf{X}_i, T_i; \Theta) \sum_{r \in D_{T_i[\mathbf{X}_i]}} \log \beta'(r) \quad (20)$$

Let us focus on the production rule which generates g ($\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y]$). Equation 20 can be decomposed into Equation 21.

$$Q_\beta = \sum_{g \in \mathcal{T}} Q_\beta(g) \quad (21)$$

Let x_j^i be the annotation of jth non-terminal at T_i and y_j^i be the annotation of right-side symbol of jth non-terminal at T_i.

$$\begin{aligned}
Q_\beta(g) &= \sum_{T_i \in \mathbf{T}} \frac{1}{P(T_i;\Theta)} \sum_{j \in cover(g,T_i)} \sum_{\mathbf{X}_i} P(\mathbf{X}_i, T_i; \Theta) \\
&\times \log \beta'(\mathcal{S}[x_j^i] \rightarrow g\,\mathcal{S}[y_j^i]...\mathcal{S}[y_j^i]) \\
&= \sum_{T_i \in \mathbf{T}} \frac{1}{P(T_i;\Theta)} \\
&\times \sum_{j \in cover(g,T_i)} \sum_{x_j^i, y_j^i \in H} \log \beta'(\mathcal{S}[x_j^i] \rightarrow g\,\mathcal{S}[y_j^i]...\mathcal{S}[y_j^i]) \\
&\times \sum_{\mathbf{X}_i \setminus x_j^i, y_j^i} P(\mathbf{X}_i, T_i; \Theta) \\
&= \sum_{T_i \in \mathbf{T}} \frac{1}{P(T_i;\Theta)} \\
&\times \sum_{j \in cover(g,T_i)} \sum_{x,y \in H} \log \beta'(\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y]) \\
&\times \beta(\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y]) f_{T_i}^j(x) \prod_{k \in ch(j,T_i)} b_{T_i}^k(y)
\end{aligned}$$

To optimize $Q(\Theta'|\Theta)$ with constraint $\sum_\zeta \beta'(\mathcal{S}[x] \rightarrow \zeta) = 1$ ($\zeta \in \{\eta | \mathcal{S}[x] \rightarrow \eta \in \mathcal{R}[H]\}$), Lagrange multiplier μ is used.

$$\mathcal{L} = Q_\beta + \mu(1 - \sum_{\zeta : \mathcal{S}[x] \rightarrow \zeta \in R[H]} \beta'(\mathcal{S}[x] \rightarrow \zeta))$$

$$\frac{\partial}{\partial \beta'(\mathcal{S}[x] \rightarrow g\,\mathcal{S}[y]...\mathcal{S}[y])} \mathcal{L} = 0 \quad (22)$$

Parameter update formula can be obtained by solving Equation 22. Update formula of $\pi'(\mathcal{S}[x])$ in Q_π can be derived in the same ways.