This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

Recursion-Based Biases in Stochastic Grammar Model Genetic Programming

Kangil Kim*, R.I. (Bob) McKay, and Nguyen Xuan Hoai

***Abstract*—Estimation of distribution algorithms applied to genetic programming have been studied by a number of authors. Like all estimation of distribution algorithms, they suffer from biases induced by the model building and sampling process. However, the biases are amplified in the algorithms for genetic programming. In particular, many systems use stochastic grammars as their model representation, but biases arise due to grammar recursion. We define and estimate the bias due to recursion in grammar-based estimation of distribution algorithms in genetic programming, using methods derived from computational linguistics. We confirm the extent of bias in some simple experimental examples. We then propose some methods to repair this bias. We apply the estimation of bias, and its repair, to some more practical applications. We experimentally demonstrate the extent of bias arising from recursion, and the performance improvements that can result from correcting it.**

***Index Terms*—Genetic programming, estimation of distribution algorithm, stochastic context-free grammar, recursion depth, bias**

I. Introduction

Over the past twenty years since they were first introduced by Baluja [1], evolutionary algorithms based on stochastic models, and known as probabilistic model building genetic algorithms (PMBGA) or estimation of distribution algorithms (EDA) [2], have become increasingly influential. By explicitly learning a distribution that generates good solutions, they can take advantage of knowledge about the structure of solutions, and thus have become highly competitive in solving tough problems [2]. Genetic programming (GP) [3] is known for difficult fitness landscapes, so it is not surprising that similar approaches, known as estimation of distribution algorithm in genetic programming (EDA-GP), have been popular [4]–[6]. Despite the algorithmic similarity to EDAs, the complex and variable structures of GP individuals lead to a number of issues that do not arise in EDAs. To handle these differences, a variety of stochastic models have been introduced ([7]–[18] - see [6] for a relatively up-to-date survey), most falling into one of two classes – probabilistic prototype trees (PPT) [10] and grammars, especially stochastic context-free grammars (SCFG) [14]. Whereas the theory of probabilistic graphical models (PGMs) is well-studied [2], [19], these EDA-GP representations are new and lack a 'gold standard' for comparison. In particular, sampling and learning, well understood in PGMs, are little studied in EDA-GP, leading to unanticipated biases which can damage the effectiveness of the algorithms.

In PPT models, these issues reveal themselves as amplified stochastic bias, critically limiting the performance of PPT-based EDA-GPs [20]: stochastic bias can outweigh the influence of selection, generation by generation, and thus decrease the likelihood of finding the best solution. Similar problems arise in grammar-based EDA-GPs, but stemming from a different factor, grammar recursions. These recursions create cyclic dependence between random variables in the model. In sampling, the models can (in principle) generate individuals of unbounded, or even infinite, size. However in practice, we cannot sample such individuals. The restriction to bounded-sized individuals, that we can actually generate, creates a bias (because the unsampled individuals may follow a different distribution from the bounded sampled ones). If we do not take this bias into account in learning the next model, then the new model may be biased, and as with PPT models, this bias can outweigh the effect of selection, and lead the system to regions of poor solutions.

The underlying issue is well-known and heavily studied in computational language processing [21]–[26]. However, the context of computational language processing differs substantially from EDA-GP; in particular, the EDA-GP systems intentionally and iteratively rely on grammars to generate bias through selection, whereas the whole aim of computational linguistics is to eliminate bias in a one-shot learning system (there being no sampling stage). Thus, the theory requires substantial extension and adaptation for EDA-GP. We previously developed such a theory independently [27] for grammars with a single nonterminal. Here, we extend it to general recursive structures, and re-frame it in the terminology from computational language processing. But estimating the bias, on its own, has little benefit, other than to engender caution in the application of SCFG EDA-GP. We develop a method to use the theoretical bias estimates to counteract the bias, illustrating it with applications to some typical GP problems. The results show not only that recursion-based biases seriously affect the behaviour of SCFG EDA-GP, but that negative effects can be reduced if estimates of the bias are available.

The overall structure of the paper is as follows. Section II discusses the relevant background to this work. Section III analyses recursion bias in EDA-GP, explains why it is a problem, and determines when it occurs. Section IV analyses the sources of the bias, lays the ground work for its estimation, and demonstrates the estimation for a class of simple SCFGs. Section V empirically tests the estimate (in the absence of

Kangil Kim is with Electronics and Telecommunications Research Institute, Korea. Bob McKay is with Seoul National University, Korea and the Australian National University. Nguyen Xuan Hoai is with Hanoi University, Hanoi, VietNam.

*Corresponding author: kangil.kim.01@gmail.com, https://sc.snu.ac.kr

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

selection) by injecting the inverse of the estimate and investigating whether it reduces the bias. Section VI extends the analysis to more complex and general SCFGs, deriving an algorithmic approach to obtain quantitative estimates of the bias. Section VII extends the empirical analysis of Section IV to some more complex benchmark problems. Section VIII introduces a methodology to ameliorate the bias in the presence of selection, providing a framework for understanding the distribution transition in SG-GPs. Section IX evaluates the behaviour of SG-GPs with the sampling bias correction and selection operators. The remaining sections summarise the conclusions and discuss potential future extensions.

II. Background

A. Estimation of Distribution Algorithm-Genetic Programming (EDA-GP)

EDA-GPs extend the incremental model learning paradigm of EDAs (illustrated in Algorithm 1[1]) to GP search spaces. There are two complementary perspectives. As search methods, they are evolutionary algorithms in which sampling from and learning of stochastic models replace the crossover and mutation operators of a classical evolutionary algorithm. As machine learners, they are iterative model learning algorithms, aiming to represent the distribution of fit individuals (and thus identify the optima), through repeated cycles of observation of selected solutions from samples.

Algorithm 1 Typical Estimation of Distribution Algorithm

$t \leftarrow 0$
Initialise probability model M_0 using the prior distribution
{Frequently a uniform distribution}
while not termination condition **do**
1. $t \leftarrow t + 1$
2. *sampling* – Sample individuals from M_{t-1}, generating new population P_t
3. *evaluation* – Evaluate fitness of all individuals in P_t
4. *selection* – Select fitter subpopulation S_t from P_t
5. *update* – Generate model M_t from S_t and M_{t-1}
end while

In sampling, typical EDAs create individuals by repeated stochastic selection of a value for each random variable of $\mathcal{M}$. The fitness of each individual is evaluated, and used to select fitter solutions. In the update step, a new stochastic model is learnt from the selected solutions, possibly incorporating information from the models of previous generations.

In EDA-GP, solutions are represented in structured forms such as the expression trees of classical genetic programming [3]. This generates substantial differences from other EDAs; in particular, the variable complexity of individuals requires more complex stochastic models. A variety of different model representations have been proposed.

The first was the probabilistic prototype tree (PPT) of probabilistic incremental program evolution (PIPE) [10]. It assumed statistical independence between PPT locations, paralleling the original EDA assumptions of Baluja [1]. The subsequent development of the PPT model has closely reflected contemporary development in EDAs, with increasing emphasis on representing complex conditional dependences [7]–[9], [28].

Slightly before PIPE, grammar based representations were introduced into GP; they have become increasingly influential [29]. Soon after, stochastic grammar models arose in EDA-GP, where they have had an even greater influence. As with PPT representations, subsequent work has largely emphasised more complex interdependence [9], [11]–[13], [30]–[35].

Beyond these two widely-used families, other complex models adapting Bayesian networks, N-grams, ant colony optimisation [36], and hybrid systems have been proposed [4].

Most of the emphasis in EDA-GP has fallen on introduction of new stochastic models, and demonstrating their effectiveness on relatively simple test problems, but with limited theoretical analysis. This paper is part of a series in which we aim to analyse the effects of bias in EDA-GP.

B. Stochastic Bias in EDA-GP

In EDAs, three steps are repeated each generation: sampling, selection, and learning. Sampling transforms the distribution of components from a model to a population, while learning does the reverse. Selection deliberately introduces a bias to the process, with the intention that the learning step will gradually converge to a region of good solutions. If the two other processes act in an unbiased way, (and if no unintended biases are introduced into selection) this convergence to a good region is likely to succeed. With infinite resources, it is often possible to remove all other sources of bias from sampling, selection and learning. Unfortunately, we never have infinite resources, and as a consequence, bias is introduced. The easiest way to observe it is to turn off selection. In that scenario, an unbiased system would not converge; a biased system may converge to a location dictated by the bias. When selection is present in an otherwise biased system, the two sources of bias will compete, and may lead to convergence to a different region than would be dictated by selection alone, so that the system fails to optimise its objective.

One important form of finiteness is population size. As a result of finite sample populations, the sample distribution may represent the model distribution inaccurately. In [20], we showed that this bias, which is present in all EDA systems, is amplified in PPT-model EDA-GP systems, and that (as usually implemented) this amplification is exponential in the depth of the PPT tree. Thus, unremediated PPT-based systems cannot hope to solve problems with depths typical of real GP problems (perhaps 12-15 deep). We proposed a solution based on likelihood weighting, and demonstrated that it works.

Because of the dependence inherent in EDA-GP representations, this form of amplified stochastic bias is universal in EDA-GP systems, and the analysis in [20] can be directly extended, with obvious changes to take into account different notations. However, its importance varies according to the depth of dependence typical in the representation. Specifically in the case of grammar-based representations, the explicit representation of recursion means that typical depths are much

[1]This code does not aim at complete generality, but rather gives a flavour of typical EDAs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

less – of the order of three or four. Thus, amplification of stochastic biases is much less than in PPT representations.

C. Other Sources of Bias in Grammar-Based EDA-GP

In grammar-based EDA-GP in general, other limitations are more important. The key limitation is solution size: since it incorporates recursion, a grammar model can in principle generate infinite individuals. In practice, we cannot support this. This restriction introduces a bias that is potentially very important in grammar-based systems. We presented an initial study of this bias, limited to grammars with a single nonterminal, in [27]. At the time, we were unaware of previous work on some aspects of this topic in computational language processing (CLP, see below),so that the terminology was not consistent with that preceding theory. We use the CLP terminology here.

There is one exception to the assumption that recursion means that grammar-based systems have low depths of dependence. This arises in grammar-learning systems (systems that learn not only the probabilities in a grammar model, but the structure of the grammar itself). In these cases, the grammar complexity is less controlled, and deep dependence may arise. Because we would like this paper to be relatively self-contained, and for brevity, we leave the synthesis of the current research (on biases arising from finite size limitations) and the earlier research (on biases arising from finite population sizes and their interaction with dependence) to future work.

D. Stochastic Context-Free Grammars (SCFG)

An SCFG G is a weighted context-free grammar as below:

$$
\begin{aligned}
\mathrm{G} &= \{N, T, R, S, \pi\} \\
N &= \{n_i | 1 \leq i \leq n \in \mathbb{N}\} \text{ (nonterminal symbols)} \\
T &= \{t_i | 1 \leq i \leq t \in \mathbb{N}\} \text{ (terminal symbols)} \\
&\quad \text{satisfying } N \cap T = \phi \\
R &= \{r_{n,i} | 1 \leq i \leq r \in \mathbb{N}, r_i = (n \rightarrow \zeta_i), n \in N\} \\
&\quad (r_{n,i} \text{ represents the grammar rule } n \rightarrow \zeta_i) \\
\zeta_i &: \{1, \ldots, z \in \mathbb{N}\} \rightarrow (N \cup T)^l \\
&\quad \text{(rule RHS, a finite string of } l \text{ symbols)} \\
\pi &= \{\pi_{n,i} | 1 \leq i \leq |R| \in \mathbb{R}, n \in N\} \\
&\quad \text{(rule probabilities of } r_{n,i}) \\
\forall n \in N \quad &, \quad \sum_{\{i : r_{n,i} \in R\}} \pi_{n,i} = 1.0
\end{aligned}
$$

(adapted from Manning's introductory natural language processing (NLP) text [21]).

Each nonterminal of an SCFG generates a multinomial distribution over the corresponding right-hand-sides (RHSs). It is usually derived from maximum likelihood estimation of observed derivation trees, which are composed of symbols and links, each uniquely mapping to a nonterminal or a production. The tree samples are usually generated by *ancestral sampling*, which stochastically selects a production from the grammar's starting symbol, and then iterates the process over successive nonterminals. The iteration is repeated until a terminated tree is generated.

SCFGs were first defined in NLP, where they have been heavily analysed. In particular, learning SCFGs is a well-studied topic. Thus, it might appear that we could simply apply the theory from NLP directly to EDA-GP, with nothing more to be done. Unfortunately, it is not so simple: there are important differences, for two separate reasons, both relating to assumptions about the data. In NLP, the training instances are observed linear sentences. This raises the issue of ambiguity, because the same sentence may have multiple parses, thus complicating the learning process. By contrast, in EDA-GP, the derivation trees are directly observed, so the issue of ambiguous parses does not arise. On the other hand, NLP generally deals directly with observed sentences: there is no preceding sampling phase potentially generating infinite derivation trees. Thus, it is reasonable to assume that the stochastic grammar describing the data generates finite trees with probability 1. In contrast, typical grammars in EDA-GP may well generate infinite trees with nonzero probability (this is almost always true of the initial grammars chosen as unbiased priors). Restricting the initial grammars is no panacea, because subsequent learning may well generate such a grammar.

E. SCFGs in EDA-GP

SCFGs as a model for EDA-GP were foreshadowed in Whigham's work on learning bias [37], but were first explicitly used as the probability model of an EDA-GP in stochastic grammar-based GP (SG-GP) [14], with a fixed SCFG in which only the probability values were free to change. In the simplest version, scalar SG-GP, the SCFG has only one random variable for each nonterminal; an extended version, vectorial SG-GP, learns to assign separate variables at different depths for the same nonterminals. It is thus able to represent a more complex distribution over the observed symbols. A similar mechanism is used in program evolution with explicit learning (PEEL) [30]. It also adopts structure learning, modifying the grammar structure during the evolution. Two later systems, grammar model with program evolution (GMPE) [11] and program with annotated linkage estimation (PAGE) [31], extend the learning beyond position-based variable assignments, leading to position-independent models [4], [38].

F. Maximum Likelihood Estimation of SCFGs

The first study of maximum likelihood estimation in SCFGs was that of Chi and German [22]. They noted that an estimator $\hat{p}$ for a production $(A \rightarrow \alpha) \in R$ of a nonterminal $A \in N$ could be written as what they called the *relative frequency*:

$$
\hat{p}(A \rightarrow \alpha) = \frac{\sum_{i=1}^{n} f_{\omega_i}(A \rightarrow \alpha)}{\sum_{\beta:(A \rightarrow \beta) \in R} \sum_{i=1}^{n} f_{\omega_i}(A \rightarrow \beta)} \tag{1}
$$

where $\omega_i : i \in \{1, \ldots, n\}$ is an observed tree, with production $r_A^i \equiv (A \rightarrow \alpha)$ having frequency $f_{\omega_i}(A \rightarrow \alpha)$. They showed that the relative frequency is a sufficient statistic for a mutinomial variable. Here we study this estimator mathematically and empirically to gain an understanding of the resulting bias.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

G. Bias Analysis of SCFG in Computational Language Processing

The main sources of bias in SCFG-based EDA-GPs are recursive derivation paths, potentially generating infinite trees. This risk to learning SCFGs has been deeply studied in computational Language Processing [21]–[26]. In that field, parsing natural sentences has been a key issue in automated language understanding, translation, and other aspects of language processing [21]. Building a parser from real-world data requires a model representing the sentence structure accurately. Probabilistic context-free grammars were proposed very early as a model for representing this sentence structure, but the implied biases predicted structures with probabilities differing from the observed data. This led to theoretical studies attempting to understand these biases and improve accuracy, which we briefly introduce here.

We start with some definitions:

Definition An SCFG G is *BT-proper* iff the RHS probabilities for each nonterminal sum to 1:[2]

$$\forall A \in N \sum_{\alpha \in (N \cup T):(A \to \alpha) \in R} p\left(A \to \alpha\right) = 1 \tag{2}$$

Definition An SCFG G is *proper* if the sum over all finite length trees, of the probability of generating that tree, is 1.[3]

Definition An SCFG G is *consistent* iff sampling an infinite number of finite-length samples from G then applying the maximum likelihood estimator of equation 1 gives a learnt probability for each production the same as the original (i.e. infinite sampling of finite trees, followed by maximum likelihood estimation, generates no bias).[4]

The first and the second conditions were introduced by Booth and Thompson [24]. The first is a reasonable requirement of any SCFG. The second limits consideration to grammars that only (with nonzero probability) generate finite trees. The third permits grammars to specify infinite trees with nonzero probability, but requires that failing to generate them should not introduce a bias.

Using these definitions, Chi proved various properties of SCFGs [22], [23]; we focus on two:

- Any SCFG learnt by equation 1's maximum likelihood estimator from finite length trees is proper [22].
- An SCFG is consistent if it is proper [23].

As a result, in most realistic language learning scenarios, NLP systems can safely restrict themselves to finite length sentences; no bias will result. However, it is a different matter in EDA-GP, as we will detail in Section III.

One way of viewing SCFG sampling is as a stochastic *branching process* [25]. Miller and O'Sullivan [26] introduced an $|N| \times |N|$ matrix representation $\mathbf{M}$ given by

$$\mathbf{M}(i,j) = \sum_{\alpha \in (N \cup T):(n_i \to \alpha) \in R} p\left(n_i \to \alpha\right) c\left(n_j; \alpha\right) \tag{3}$$

where n_i is the ith element of N, i denotes rows, j denotes columns, and $c(n_j;\alpha)$ is the number of occurrences of n_j in α. Thus, each entry represents the update to the expected number of occurrences of the corresponding nonterminal in one round of the iterative process of sample generation.

Given a vector $\mathbf{X}_{d-1}$ representing the frequencies of nonterminals at stage $d-1$, we have

$$\mathbf{X}_d = \mathbf{X}_{d-1}\mathbf{M}$$

so that if X_0 is the initial frequency (for example the vector consisting only of a single start nonterminal), then we have

$$\mathbf{X}_d = \mathbf{X}_0\mathbf{M}^d \tag{4}$$

H. Depth Constraints in Genetic Programming

Improper SCFGs raise the issue of unterminated tree growth. This issue has been recognised (under different terminology) right from the start of GP, because the SCFG implicitly underlying the initialisation procedure of 'standard' expression tree GP is improper. It gives nonzero probability mass to infinite trees, which we can neither generate nor sample. While much discussion of this issue has been pragmatically focused, there are deeper analyses in [39]–[41].

Typical GP systems handle this problem by setting limits on measures of tree complexity (e.g. depth) and rejecting oversized trees. There are two common approaches. One generates a new tree from the model every time an individual is rejected; the other constrains the generation so that it cannot generate oversized trees. The latter explicitly changes the model distribution, whereas the former might appear to preserve it, so it might seem preferable. However, the distribution will in fact only be preserved if the distribution of components in rejected trees is the same as in accepted trees – which will not normally be the case. Re-generating can impose a substantial computational cost, so constraint methods are more common. We focus on constraint methods; however the analyses can be recast to cover regeneration methods as well.

Efficient implementations of constraint methods work this way. To avoid generating trees that will be rejected, systems need to determine, at each stage, which productions are still able to terminate within the depth limit, and to select only among them. A number of such algorithms have already been proposed [11], [42]. They start by computing the recursively-defined termination depth d_t of each rule and each symbol.

1) The termination depth of a terminal $t \in T$ is $d_\mathrm{t}(t) = 0$.
2) For a rule $r = (A \to \alpha) \in R$, the termination depth is the maximum of the termination depths of all RHS symbols, plus one. $d_\mathrm{t}(t) = 1 + (\max_{s \text{ occurs in } \alpha} d_\mathrm{t}(s))$
3) For a nonterminal $n \in N$, the termination depth is the minimum termination depth over all rules of which it is the LHS. $d_\mathrm{t}(n) = \min_{n \text{ is LHS of } r \in R} d_\mathrm{t}(r)$

The system compares the termination depths of all rules with the remaining available depth; the probabilities of those which exceed it are reset to zero (with the remainder being correspondingly renormalised). Here we investigate the bias resulting from this repair mechanism [11], [42], [43].

[2]Defined as 'proper' in the original Booth and Thompson paper [24].

[3]Our terminology follows Chi [23]; Booth and Thompson refer to this as 'consistent' [24].

[4]Our terminology again follows Chi [23].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

I. Benchmark Problems

To illustrate the bias analysis we develop in the next few sections, and to demonstrate the value of remediating it, we work with three well-known benchmark GP problems.

1) Symbolic Regression Problems [3]: This class of problems is a traditional benchmark for GP. The target is to find an expression tree for a mathematical function, using at least the general arithmetic operators $+, -, \times, /$, and frequently trigonometric, logarithmic and exponential operators as well. Because of the difficulties arising from division by zero, $/$ is generally regarded as protected division, in which $x/0 = 1$ for any value of x. The system is given function values at a set of predefined data points, and the goal is to minimise the absolute error summed over those data points.

2) The Max Problem [44]: The goal is to maximise the value of the arithmetic expression represented by a tree composed from function set I and terminal set T, within a predefined depth limit. Typically, $I = \{+, \times\}$ and $T = \{0.5\}$.

3) The Royal Tree Problem: The goal is to find a specific form of tree [45] whose fitness is recursively defined by the content of a node and the structure of its children. A subtree is called a *complete level-k tree*, if its root is labelled k (which implies that it has k children) and all children are *complete level-$(k-1)$ trees*. The search target is a specific level-k tree; the search is increasingly difficult as k increases.

The detailed fitness is determined by how close the tree is to a complete tree; it is recursively calculated from the bottom up. The fitness of a node is the sum of the fitnesses of all child nodes, weighted by parameters that ensure the maximum fitness is given to complete trees. If a child of a node labelled $k+1$ is a level-k complete tree, its fitness is weighted by the *Full bonus* parameter. If not, it is weighted by *Penalty*. After summing the fitness of all children, the sum is weighted by *Complete bonus* if the subtree rooted at the node is complete. If not, it is multiplied by *Partial bonus*. Usually, *Full bonus*, *Penalty*, *Complete bonus*, and *Partial bonus* are set to 2, $\frac{1}{3}$, 2, and 1, respectively. A level-1 complete tree is composed of a node with only one terminal child.

III. Consistency in EDA-GP SCFGs

As we mentioned in Section II, the problem of recursion-based bias in probabilistic learning is well understood in computational linguistics. Chi's demonstration [22], [23] that an SCFG learnt from finite length trees is proper, and therefore consistent, implies that a system learning initially from a set of finite sentences will maintain that structure over repeated cycles of sampling finite sentences and learning from them. Does this not mean that the same will apply to the EDA-GP cycle, so that EDA-GP systems will be unbiased, at least with respect to recursion and finiteness limitations?

Unfortunately, this is not the case. Paradoxically, the problems mainly arise from measures undertaken to limit other sources of bias. In EDA-GP, we typically start with a uniform probability distribution over the RHSs for each nonterminal (so as not to bias which RHS is chosen). If the grammar includes a number of binary operators, as is common, this is highly likely to imply nonzero probability mass for infinite individuals – and hence, to have an improper initial grammar. If this were all, we could probably live with it. The initial grammar would be improper, but since the second generation grammar would be learnt from a set of finite trees, Chi's argument would apply, and from then on everything would be fine.

But this is not the end of the story. To improve exploration and slow down convergence, we typically feed some form of the uniform distribution back into the EDA as it progresses – either directly, through a conservative learning rate, or indirectly through mutation and through retention of individuals from earlier generations. Things get even worse if we incorporate hybrid mechanisms such as local search or crossover operators. All of these alter the distribution in ways that are likely to destroy propriety.

In fact, there is an argument that propriety is not even what we want. We know from GP bloat research that for most realistic problems, the vast majority of correct solutions to a GP problem are large, the expected size being unbounded. While there has been progress in controlling bloat, methods that directly favour smaller individuals (e.g. through a parsimony pressure in the fitness function) risk premature convergence to small, relatively unfit solutions because the size landscape is smooth and hence, easy to solve, while the problem performance landscape is typically very difficult. Even if we could preserve propriety after the first generation, it is highly likely to have similar deleterious effects.

In practice, we are not aware of any grammar-based EDA-GP which either initially generates, or preserves, propriety. In the EDA-GP systems explicitly described as such [11], [14], [30], [31], the initial grammar will generally not be proper, nor will the model learning/update mechanism preserve propriety. In the subclass typically described as ant colony optimisation GP (ACO-GP) [9], [12], [13], [32]–[35], pheromone decay directly reinforces a uniform distribution, and hence, guarantees that propriety (for most grammars) will not be preserved.

To summarise, present-day EDA-GPs neither generate propriety initially, nor preserve it. Thus, there is a practical need to study any bias induced by impropriety, and to understand how it may be handled. This might sound like an interim solution, while we try instead to design EDA-GP systems that generate, and preserve, propriety. For the reasons discussed above, we don't believe so: impropriety (in the sense of Chi) is a desirable property for EDA-GP, and it is unlikely that a propriety-preserving EDA-GP system would have good performance.

IV. Bias Analysis for Simple Recursion

A. Context for Estimating Recursion Bias

To focus on the biases resulting from finite sized trees in sampling and learning from CFG models, we need to remove other sources of bias. Selection bias is easy to remove: we simply turn off selection. This is the path we follow in most of the subsequent analyses. Other biases that are used to control learning (reintroduction of uniform prior etc.) are also omitted. Sampling noise is more difficult: in theoretical analyses, we eliminate it by assuming infinite populations; in empirical work, we rely on large sample sizes.

When we come to the focus of this study, the bias arising from finite size limitations, it splits into two parts. The first

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

is the restriction to finite trees: we cannot, even in principle, directly represent randomly sampled infinite trees. If we generate an unbiased infinite sample from a grammar, then use maximum likelihood estimation to learn from it, we will recover the original grammar probabilities. But we have no such guarantee if we eliminate some part (the infinite trees) from the sample. That is one source of bias. The second source arises from depth limits. We cannot in general be certain, as we are generating an individual by ancestral sampling, whether it is going to terminate in a finite depth. So we generally impose a depth limit. But eliminating individuals larger than the limit also changes the distribution, with the effect depending on the way the limit is enforced. Specifically, we focus on the repair method introduced in Section II.

B. Maximum Likelihood Estimation

In this section, we assume infinite population sizes. Thus the frequencies of symbols and productions are infinite, so we cannot directly compute the relative frequency as the maximum likelihood estimator as in Section II. To resolve this, we instead use the limit of this ratio as the population size tends to infinity. We call it the *proportional frequency*, denoted as $f()$ in all subsequent sections; as an extension of this notation, we use the notation $f_d()$ to denote the proportional frequency of generating an object at a specific depth d.

In the infinite population limit, and assuming there is no depth limit, samples will follow the distribution specified in the grammar, and the sampling of a production is independent of how its LHS was derived, so we can derive the proportional frequency of a production as in equation 5.

$$f(A \to \alpha) = f(A)p(A \to \alpha) \tag{5}$$

or for a specific depth

$$f_d(A \to \alpha) = f_d(A)p(A \to \alpha) \tag{6}$$

We often need to iterate over all productions matching a given LHS; for brevity we define the set $M(A, R)$ of productions in rule set R matching A as equation 7.

Definition

$$M(A, R) = \{A \to \alpha : (A \to \alpha) \in R\} \tag{7}$$

As Miller and O'Sullivan noted, every occurrence of a nonterminal A at depth $d-1$ gives rise to an expected $p(A \to \alpha)c(B; \alpha)$ occurrences of nonterminal B at depth d. Specialising equations 3 and 4 to the case of a single nonterminal (in which case we no longer need the matrix notation), we have the recurrence relation equation 8.

$$f_d(A) = f_{d-1}(A) \sum_{(A\to\alpha)\in M(A,R)} p(A \to \alpha)c(A; \alpha) \tag{8}$$

Denoting $\sum_{(A\to\alpha)\in M(A,R)} p(A \to \alpha)c(A; \alpha)$ (expected number of A generated from A in one rule application) by h, solving this recursion gives equation 9.

$$\begin{aligned} f_d(A) &= h^d f_0(A) \\ &= h^d \end{aligned} \tag{9}$$

Applying equation (6), we obtain the proportional frequency of productions $r \in R$ as equation 10.

$$f_d(r) = h^d p(r) \tag{10}$$

When we impose a depth limit $\bar{d}$, this changes. Let R_T denote the productions of the grammar that have only terminals in the RHS (i.e. $R_T = \{A \to \alpha : \alpha \in T^*\}$) and $\bar{R_T}$ its complement. For any non-degenerate grammar with a single nonterminal,[5] R_T is nonempty, because grammars with a single non-terminal can only terminate the generation process through a production $A \to \alpha$ with $\alpha \in T^*$. Thus, equation 10 still holds up to depth $\bar{d}-1$. At depth $\bar{d}$, we have a different probability distribution for $r = (A \to \alpha) \in R$, equation 11.

$$p_{\bar{d}-1}(r) = \begin{cases} \frac{p(r)}{\sum_{s\in M(A,R_T)} p(s)} & \text{if } \alpha \in R_T \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Thus, applying equation 11, the proportional frequency calculation for $r = (A \to \alpha) \in R$ at $\bar{d}-1$ changes correspondingly from equation 10 to equation 12.

$$f_{\bar{d}-1}(r) = \begin{cases} h^{\bar{d}-1} \frac{p(r)}{\sum_{s\in M(A,R_T)} p(s)} & \text{if } r \in R_T \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

We can now find the relative frequency of rule $r = (A \to \alpha) \in R$ by summing over all depths to get equation 13.

$$\begin{aligned} f(r) &= \sum_{d=0}^{\bar{d}} f_d(r) \\ &= \sum_{d=0}^{\bar{d}-2} f_d(r) + f_{\bar{d}-1}(r) \\ &= \sum_{d=0}^{\bar{d}-2} f_d(A)p(r) + \\ &\quad \begin{cases} f_{\bar{d}-1}(A) \frac{p(r)}{\sum_{s\in M(A,R_T)} p(s)} & \text{if } r \in R_T \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

Equation 9 and recursion elimination give equation 14.

$$\begin{aligned} f(r) &= f_0(A)\frac{1-h^{\bar{d}-1}}{1-h}p(r) + \\ &\quad \begin{cases} h^{\bar{d}-1} f_0(A) \times \frac{p(r)}{\sum_{s\in M(A,R_T)} p(s)} & \text{if } r \in R_T \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{14}$$

To find the denominator of the maximum likelihood estimator (equation 1), $\sum_{s\in M(A,R)} \sum_\omega f(r; \omega)$, we can directly apply equation 14 as in equation 15.

$$\begin{aligned} &\textstyle\sum_{s\in M(A,R)} f(A \to \beta) \\ &= \textstyle\sum_{s\in M(A,R_T)} f(s) + \sum_{s\in M(A,\bar{R_T})} f(s) \\ &= f_0(A)\frac{1-h^{\bar{d}-1}}{1-h} \textstyle\sum_{s\in M(A,R_T)} p(s) + \\ &\quad h^{\bar{d}-1} f_0(A) \frac{\sum_{s\in M(A,R_T)} p(s)}{\sum_{t\in M(A,R_T)} p(t)} + \\ &\quad f_0(A)\frac{1-h^{\bar{d}-1}}{1-h} \textstyle\sum_{s\in M(A,\bar{R_T})} p(s) \\ &= f_0(A)(\frac{1-h^{\bar{d}-1}}{1-h} + h^{\bar{d}-1}) \end{aligned} \tag{15}$$

[5] An SCFG is degenerate in this sense if it can only generate infinite trees.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

We then derive the maximum likelihood estimator $\hat{p}(r)$ for nonterminating rules $r = (A \to \alpha) \in \bar{R}_T$ as equation 16.

$$\hat{p}(r) = \frac{p(r)\frac{1-h^{\bar{d}-1}}{1-h}}{\frac{1-h^{\bar{d}-1}}{1-h} + h^{\bar{d}-1}} \tag{16}$$

Terminating rules $r = (A \to \alpha) \in R_T$ give equation 17.

$$\hat{p}(r) = \frac{p(r)\frac{1-h^{\bar{d}-1}}{1-h} + h^{\bar{d}-1}\frac{p(r)}{\sum_{s\in M(A,R_T)} p(s)}}{\frac{1-h^{\bar{d}-1}}{1-h} + h^{\bar{d}-1}} \tag{17}$$

C. Evaluation of Bias

The depth bias – the change in probability due to sampling – can be formalised as $\delta p(r)$ for $r = (A \to \alpha)$ as in equation 18.

$$\delta p(r) = \hat{p}(r) - p(r) \tag{18}$$

For grammars with a single nonterminal, the bias is readily derived from equations 16 and 17 as equation 19.

$$\delta p(r) = \begin{cases} \frac{h^{\bar{d}-1}(1-h)}{1-h^{\bar{d}}} \times \\ \left[\frac{p(r)}{\sum_{s\in M(A,R_T)} p(s)} - p(r)\right] & \text{if } \alpha \in R_T \\ \frac{h^{\bar{d}-1}(1-h)}{1-h^{\bar{d}}} p(r) \times -1 & \text{otherwise} \end{cases} \tag{19}$$

Equation 19 is directly useful for estimating the bias resulting from a depth limit, but it is also interesting to understand its asymptotic behaviour as $\bar{d} \to \infty$. When $h \leq 1$, the limit bias is 0, so the SCFG is proper. Otherwise, by eliminating $h^{\bar{d}}$, we can rewrite the maximum likelihood as equation 20.

$$\hat{p}(r) = \begin{cases} \frac{p(r) + (h-1)\frac{p(r)}{\sum_{s\in M(A,R_T)} p(s)}}{h} & \text{if } r \in R_T \\ \frac{p(r)}{h} & \text{otherwise} \end{cases} \tag{20}$$

The corresponding bias is given by equation 21.

$$\delta p(r) = \begin{cases} p(r)\frac{h-1}{h}\frac{1-\sum_{s\in M(A,R_T)} p(s)}{\sum_{s\in M(A,R_T)} p(s)} & \text{if } \alpha \in R_T \\ p(r)\frac{h-1}{h} \times -1 & \text{otherwise} \end{cases} \tag{21}$$

We thus obtain the same condition for propriety as has previously been obtained in the SCFG literature [22], [23], [26], determined by evaluating the eigenvalues of the matrix. If we take a simple grammar such as

$$\begin{aligned} E &\to EE \\ &\to X \end{aligned}$$

we get $h = \frac{1}{2}$ as in Chi's example [22]. In that literature, the primary emphasis is on determining the location of the phase transition between propriety and impropriety; this is a relatively minor issue for EDA-GP, since we are generally unable to control the probabilities to maintain propriety. Our interest, instead, is to measure the extent of bias, and if possible, to counterbalance it for consistency.

D. Examples of Bias in Simple Grammars

Examples of the detailed analysis of bias in simply-recursive grammars are provided in the supplementary materials.

V. Experimental Validation: Simple Grammars

In this section, we experimentally test both the existence and extent of a bias problem due to finitisation/boundedness, in grammars with simple recursions. In so doing, we verify that our estimate of the extent of bias is accurate, by showing that we can accurately invert the bias when there is no selection, and hence avoid convergence to random values in the absence of selection. We test it with scalar SG-GP [14], the simplest grammar-based EDA-GP system, on three well-known benchmark problems: Royal tree, symbolic regression and Max. In the tests, we compare the performance of a normal SG-GP and one incorporating the proposed bias corrections.

A. EDA-GP System Design

We chose scalar SG-GP for analysis because its simplicity will help to reduce interference from other factors. Its probability model consists of an SCFG built by maximum likelihood estimation; it generates individuals by ancestral sampling.

B. Benchmark Problems

TABLE I
Grammars for Benchmark Problems (E0: Starting Symbol)

Symbolic Regression			Max			Royal Tree		
E0	→	E0 + E0	E0	→	E0 + E0	E0	→	c E0 E0 E0
E0	→	E0 - E0	E0	→	E0 × E0	E0	→	b E0 E0
E0	→	E0 × E0	E0	→	0.5	E0	→	a E0
E0	→	E0 / E0				E0	→	X
E0	→	X						

TABLE II
Parameter Settings for Experiments

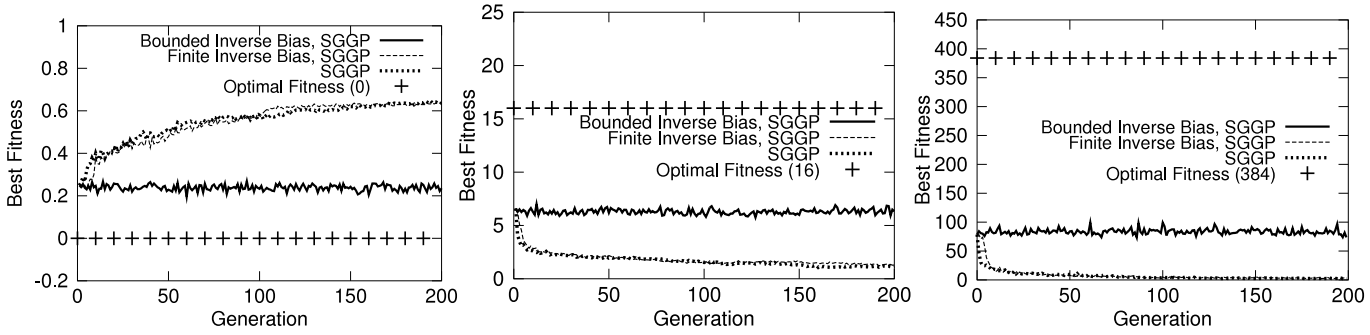
Common Parameters		Problem-dependent Parameters	
Genotype	derivation tree	Derivation depth limit	
Population	1000	Symb. Regression	4
Generations	200	Max	5
Runs	30	Royal tree	4
Selection	truncation	Symbols of mapped exp. trees	
Ratios	30,60,90,100	Symb. Regression	+,−,×,/,X
Update	max. likelihood	Max	+, ×, 0.5
Sampling	ancestral	Royal tree	a, b, c, X

The solution spaces for the problems are defined by the grammars in Table I, combined with the derivation depth limits in Table II (we count the depth of a terminal node as 0). Fitness functions are as defined in Section II. Other parameters are shown in Table II.

1) Symbolic Regression: The symbol set for the simple symbolic regression was $\{+, -, \times, /, X\}$, with the target function being $f(x) = x^3 + x^2 + x$, as in [3]. Fitness was the root mean square error over the 21 points, with the optimum of this minimisation problem being 0.

2) Max: The max problem used the symbol set $\{\times, +, 0.5\}$. Fitness was the value of the corresponding mathematical expression, with a global optimum of 16 for the specified depth. Attaining this value requires a depth of 5 derivations, generated from a single E0 by one (E0 → E0 + E0), fourteen (E0 → E0 × E0), and sixteen (E0 → 0.5). The max problem is particularly challenging for scalar SG-GP, whose depth-free

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation



(a) Symbolic Regression (b) Max (c) Royal tree

Fig. 1. Mean Best Fitness of Scalar SG-GP without Selection for Problem Spaces Defined by Simple Recursion

grammar representation gives it no way to directly represent the optimal solution space; nevertheless, the problem can help to illustrate some of the issues with recursion bias.

3) Royal Tree: For the Royal tree problem, we used the level-3 problem, with an optimal fitness of 384.

C. Results

TABLE III
FINAL GENERATION MEAN BEST FITNESS OF SCALAR SG-GP, 30 RUNS (SIMPLE GRAMMARS, NO SELECTION)

Problem	Uncorrected Bias	Finite Inverse Bias	Bounded Inverse Bias
Regression (min: 0)	0.64 ± 0.07	0.64 ± 0.04	0.24 ± 0.08
Max(max: 16)	1.18 ± 1.03	1.33 ± 1.00	6.34 ± 1.82
Royal tree (max: 384)	2.14 ± 1.78	2.66 ± 2.95	73.6 ± 18.0

The mean best fitness over the 30 runs is illustrated in Fig. 1 for three different forms of sSG-GP (uncorrected finiteness bias, bias correction for finiteness only, and bias correction for both finiteness and depth bounds denoted by SG-GP, finite inverse bias, and bounded inverse bias). The figures are supplemented by the detailed values in Table III, which show the final generation mean best fitness (over all runs).

D. Discussion

These results confirm our hypothesis: bias due to finiteness/boundedness limitations is severe, and is highly likely to adversely affect the performance of grammar-based EDA-GP systems. The experimental results demonstrate that we have correctly estimated the extent of the bias, and are able to accurately cancel it. The results presented are only for sSG-GP, i.e. a system which learns grammar probabilities alone, without updating the grammar structure. However, the finiteness bias problem is also present in systems that learn the grammar structure, since they incorporate the same source of bias (that the sample distribution does not match the prior model distribution because of finitisation). In fact, it is highly likely to be exacerbated, because the resulting mis-learning may be incorporated not merely in the model probability values (which can, in principle, be subsequently recovered), but in the structure, which is much more difficult to un-learn.

VI. BIAS IN MORE COMPLEX GRAMMARS

The preceding analysis focused on singly recursive grammars for purposes of illustration. While many real-world applications (including symbolic regression) and most GP benchmarks only require such grammars, other real-world applications require more complex grammars with multiple nonterminals. In systems that learn the grammar structure in addition to probabilities, the user does not control the structure, and generation of multiple recursive nonterminals is the norm. While interactions between recursions complicate the picture, the derivation process is as much as simple grammars. We can use similar methods to compute nonterminal frequencies; we just need to handle the complexity symbolically. Fortunately, matrix representations can help.

We already have the core update formula in matrix form, equation 4, from computational linguistics [23], [26].

Before presenting the detailed analysis, we outline the process informally. The matrix for update is extracted from the grammar (equations 22, 23). If a maximum depth is imposed, the frequency update represented by the matrix changes, so we give a formal definition of the set of rules that can still terminate at a given depth (equation 24), and use this to derive a general form for the rule probability combining the normal probabilities and the distortions induced by the depth limit. (equation 25). Using this form, we can obtain the update matrix adjusted by the distortion at each depth (equation 26). Repeatedly applying the update matrix, we can derive the frequency of rules in observed trees satisfying the depth constraint (equations 27, 28). Summing these frequencies over all available depths, we obtain the rule frequencies (equation 29). Summing over all rules from the same nonterminal and normalising give us the relative rule probability (equation 30). Finally, the bias is the difference between this probability and the probability specified in the SCFG (equations 31, 32). We formalise this below.

For any rule $r \in R$ of a CFG, we can compute the minimum termination depth $d_{\mathrm{t}}(r)$ defined in Section II. The maximum over all rules, $d_{t_{\mathrm{m}}} = \max_{r \in R}(d_{\mathrm{t}}(r))$, defines the maximum propagation of the effects of the depth limit $\bar{d}$ under the repair mechanism: nonterminals at depths further from the depth limit (i.e. shallower than or equal to $\bar{d} - d_{t_m}$) are not affected

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

by the limit.[6] Within this range, we can apply equation 4. For $A \in N$ whose index in the matrix representation is i, we define the selector unit vector $\mathbf{u}(A)$ as $\mathbf{u}(A)(i) = 1$, $\mathbf{u}(A)(j) = 0$ for $i \neq j$. We can thus write equation 22.

$$f_d(A) = \mathbf{X}_0 \mathbf{M}^d \mathbf{u}(A)^{\mathrm{T}} \tag{22}$$

Applying equation 5 to $r = (A \to \alpha) \in R$, gives equation 23.

$$f_d(r) = \mathbf{X}_0 \mathbf{M}^d \mathbf{u}(A)^{\mathrm{T}} p(r) \tag{23}$$

Beyond depth $\bar{d} - d_{t_m}$ modifications are needed. We do so in a general way for all depths. We define R_d, the rules which, at depth d, can still terminate by depth $\bar{d}$, as in equation 24.

$$R_d = \{r \in R : d_{\mathrm{t}}(r) \leq (\bar{d} - d)\} \tag{24}$$

For $d \leq (\bar{d} - d_{t_m})$, $R_d = R$, while for $d > (\bar{d} - d_{t_m})$, $R_d = \phi$. We can then define a modified rule probability $\tilde{p}_d(r)$ at depth d for $r = (A \to \alpha) \in R$ by renormalising over only rules that can still terminate, as in equation 25.

$$\tilde{p}_d(r) = \begin{cases} \frac{p(r)}{\sum_{s \in M(A,R_d)} p(s)} & \text{if } r \in R_d \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

Of course, for $d \leq d_{\bar{t}}$, $\tilde{p}_d(r) = p(r)$. For brevity, we denote $(\bar{d} - d_{t_m})$ by $d_{\bar{t}}$. We can then define an adjusted matrix $\mathbf{T}_d$ of adjusted probabilities analogous to $\mathbf{M}$, as in equation 26.

$$\mathbf{T}_d(i,j) = \sum_{(n_i \to \alpha) \in M(n_i,R)} \tilde{p}_{d-1}(n_i \to \alpha) c(n_j; \alpha) \tag{26}$$

Correspondingly, for $(d-1) \leq d_{\bar{t}}$, $\mathbf{T}_d = \mathbf{M}$.

We can now rewrite equation 23 in a form covering all depths d, first as equation 27, and then as equation 28.

$$f_d(r) = \mathbf{X}_0 \prod_{i=1}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}} \tilde{p}_d(r) \tag{27}$$

$$f_d(r) = \begin{cases} \mathbf{X}_0 \mathbf{M}^d \mathbf{u}(A)^{\mathrm{T}} p(r) & \text{if } d \leq d_{\bar{t}} \\ \mathbf{X}_0 \mathbf{M}^{d_{\bar{t}}} \prod_{i=d_{\bar{t}}+1}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}} \tilde{p}_d(r) & \text{if } d > d_{\bar{t}} \end{cases} \tag{28}$$

With this, we can derive the cumulative frequency of a production over all depths as equation 29.

$$\begin{aligned} f(r) &= \sum_{d=0}^{d_{\bar{t}}} f_d(r) + \sum_{d=1+d_{\bar{t}}}^{\bar{d}} f_d(r) \\ &= p(r)\mathbf{X}_0 \sum_{d=0}^{d_{\bar{t}}} \mathbf{M}^d \mathbf{u}(A)^{\mathrm{T}} + \\ &\quad \mathbf{X}_0 \mathbf{M}^{d_{\bar{t}}} \sum_{d=d_{\bar{t}}+1}^{\bar{d}} \tilde{p}_d(r) \prod_{i=d_{\bar{t}}+1}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}} \end{aligned} \tag{29}$$

Although this appears complex, it can be efficiently computed through dynamic programming. Applying it to all productions of the nonterminal A gives the denominator for the maximum likelihood as equation 30.

$$\begin{aligned} &\sum_{r \in M(A,R)} f(r) \\ &= \sum_{r \in M(A,R)} \Bigg\{ p(r)\mathbf{X}_0 \sum_{d=0}^{d_{\bar{t}}} \mathbf{M}^d \mathbf{u}(A)^{\mathrm{T}} + \\ &\quad \mathbf{X}_0 \mathbf{M}^{d_{\bar{t}}} \sum_{d=d_{\bar{t}}+1}^{\bar{d}} \tilde{p}_d(r) \prod_{i=d_{\bar{t}}+1}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}} \Bigg\} \\ &\quad \text{(application of equation 29)} \\ &= \mathbf{X}_0 \sum_{d=0}^{d_{\bar{t}}} \mathbf{M}^d \sum_{r \in M(A,R)} p(r) \mathbf{u}(A)^{\mathrm{T}} + \\ &\quad \mathbf{X}_0 \mathbf{M}^{d_{t_m}} \sum_{d=d_{\bar{t}}+1}^{\bar{d}} \sum_{r \in M(A,R)} \tilde{p}_d(r) \prod_{i=d_{\bar{t}}+1}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}} \\ &\quad \text{(distributing)} \\ &= \mathbf{X}_0 \sum_{d=0}^{d_{\bar{t}}} \mathbf{M}^d \mathbf{u}(A)^{\mathrm{T}} + \mathbf{X}_0 \mathbf{M}^{d_{\bar{t}}} \sum_{d=d_{\bar{t}}+1}^{\bar{d}} \prod_{i=d_{\bar{t}}+1}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}} \\ &\quad \text{(from the definition of rule probability in SCFG)} \end{aligned} \tag{30}$$

Combining equations 1, 29 and 30, we can obtain the maximum likelihood estimator $\hat{p}(r)$ and use this to find the bias $\delta p(r)$. The initial form is complex, and unlike the single nonterminal case, does not simplify easily. Nevertheless, it supplies all we need to find the bias for a specific grammar. We first apply the sum-splitting technique of equation 28 to equation 1, resulting in the estimate of the bias $\delta p(r)$ of a rule $r = (A \to \alpha) \in R$ as equation 31. Substituting the detailed results of equations 29 and 30 gives us the form of equation 32.

$$\begin{aligned} &\delta p(r) \\ &= \hat{p}(r) - p(r) \\ &= \frac{f(r)}{\sum_{s \in M(A,R)} f(s)} - p(r) \\ &= \frac{\sum_{d=0}^{\bar{d}-1} f_d(A) \tilde{p}_d(r)}{f(A)} - \frac{\sum_{d=0}^{\bar{d}-1} f_d(A) p(r)}{f(A)} \\ &= \frac{\sum_{d=d_{\bar{t}}+1}^{\bar{d}-1} f_d(A) \left(\tilde{p}_d(r) - p(r)\right)}{f(A)} \end{aligned} \tag{31}$$

$$= \frac{\sum_{d=d_{\bar{t}}+1}^{\bar{d}-1} \mathbf{X}_0 \mathbf{M}^{d_{\bar{t}}} \prod_{i=d_{\bar{t}+1}}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}} (\tilde{p}_d(r) - p(r))}{\mathbf{X}_0 (\sum_{d=0}^{d_{\bar{t}}} \mathbf{M}^d) \mathbf{u}(A)^{\mathrm{T}} + \mathbf{X}_0 \mathbf{M}^{d_{\bar{t}}} \sum_{d=d_{\bar{t}}+1}^{\bar{d}} \prod_{i=d_{\bar{t}}+1}^{d} \mathbf{T}_i \mathbf{u}(A)^{\mathrm{T}}} \tag{32}$$

Equation 32 is consistent with equation 19 for single terminals, as are all results of this section. In practical implementations, this equation is calculated by the algorithm briefly noted in Algorithm 2. It is readily automated, since equation 32 depends only on the SCFG and its depth limit.

VII. Experimental Validation: Complex Grammars

While the mathematical analysis for more complex grammars is more complicated than for the simpler grammars we

[6] Note that this analysis is specific to this repair mechanism. In particular, sudden death mechanisms for size control potentially affect the distribution at all depths.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

Algorithm 2 A Bias Estimation Algorithm in EDA Frame

$t \leftarrow 0$
Initialise probability model M_0 using the prior distribution
{Frequently a uniform distribution}
Evaluate $d_{\bar{t}}$ from skeleton of M_0 limited by L
{through the algorithm in Section II}
while not termination condition **do**
1. $t \leftarrow t + 1$
BiasEstimation(M_t, L)
2. *sampling* – Sample individuals from M_{t-1}, generating new population P_t
3. *evaluation* – Evaluate fitness of all individuals in P_t
4. *selection* – Select fitter subpopulation S_t from P_t
5. *update* – Generate model M_t from S_t and M_{t-1}
end while

BiasEstimation(M_t, $d_{\bar{t}}$) :
do
Extract $\mathbf{M}$,$\mathbf{u}$, $\mathbf{X}_0$ (by equation 3)
for i in $d_{\bar{t}} + 1 \leq i \leq d$ **do**
Extract $\tilde{p}_i$ (by equation 25)
Extract T_i (by equation 26)
end for
Calculate equation 32 from collected terms with given $d_{\bar{t}}$

TABLE IV
COMPLEX GRAMMARS FOR BENCHMARK PROBLEMS (E0: STARTING SYMBOL)

Symbolic Regression		Max		Royal Tree	
E0		E0		E0	
→	E0 + E0	→	E0 + E0	→	d E0 E0 E0 E0
→	E0 - E0	→	E0 × E0	→	c E0 E0 E0
→	E1 × E1	→	E1	→	b E1 E1
→	E1 / E1			→	a E1
→	X			→	X
E1		E1		E1	
→	E0 + E0	→	E1 + E1	→	d E0 E0 E0 E0
→	E0 - E0	→	E1 × E1	→	c E0 E0 E0
→	E1 × E1	→	X	→	b E1 E1
→	E1 / E1			→	a E1
→	X			→	X

considered earlier, there is no in-principle difference: if we can estimate the resulting bias, we can invert it. So how will this play out in complex cases? We illustrate the process using a similar experimental setting to Section V.

a) EDA-GP systems: As before, we used SG-GP, with parameter settings as in Table II.

b) Benchmark Problems: Complex grammars may arise in practical applications for a variety of reasons. In [46], they were needed to represent interacting boolean and arithmetic types. In other cases, complex grammars provided better representation power, leading to performance improvements over previous grammar-based EDA-GPs [11], [14], [31]. However, to simplify comparisons, we used essentially the same problems as in Section V, increasing the grammar complexity by duplicating the grammar and constructing paths through the two copies. The primary difference between these problems and those of Section V is the level of finitisation bias implicit in the grammar, as shown in Table IV. The parameter settings were those of Table II with one exception, the depth limits for Royal tree and max were increased by 1, because the increased structure of the new grammars made the earlier problems too easy to solve (in Royal tree) or required longer derivation paths for terminating trees (in Max), masking the effects of interest.

TABLE V
FINAL GENERATION MEAN BEST FITNESS OF SCALAR SG-GP, 30 RUNS (COMPLEX GRAMMARS, NO SELECTION)

Problem	Uncorrected Bias	Bounded Inverse Bias
Regression (min: 0)	0.63 ± 0.04	0.23 ± 0.11
Max (max: 16)	1.36 ± 1.00	8.33 ± 2.20
Royal tree (max: 6144)	2.6 ± 1.5	69.3 ± 30.9

c) Results: We present the results for complex grammars in a similar way to those of Section V. Fig 2 corresponds to Fig. 1, and Table V to Table III. The only differences underlying the results are the different grammars used (complex vs simple), and the slightly different variants of the Royal tree and Max problems; the only difference in the presentation lies in the omission of the finiteness inverse bias, since our results in Section V suggest that it is of low value.

The results confirm that there is a serious bias arising from finitisation and size bounds; the effective cancellation of this bias by the bias inversion method demonstrates that our estimate of the bias is correct, and that (in the absence of selection) it can be compensated for.

VIII. AMELIORATION OF RECURSION BIAS

A. Issues in Bias Correction

The preceding results show that there is substantial finitisation-related bias for both simple and complex grammars, and that we can accurately estimate it. To be useful, we need to transfer our methods for inverting the effect of equation 19 to practical applications with selection pressure.

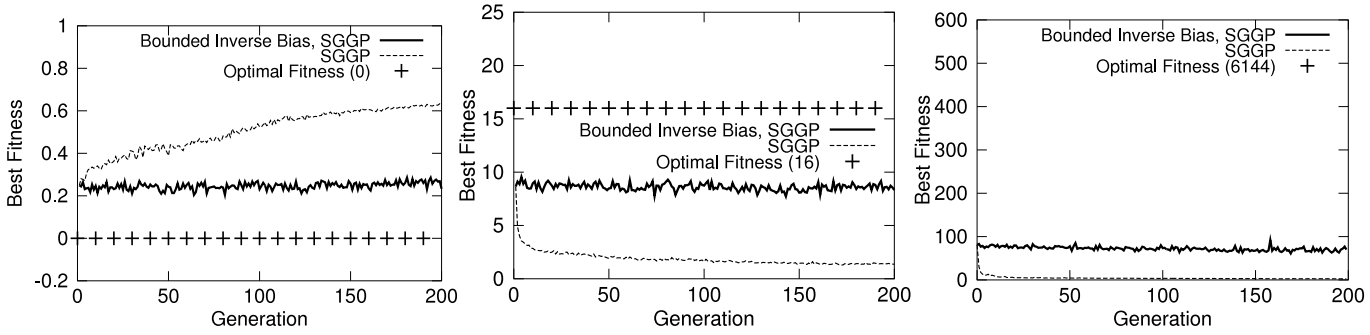
In the case of an infinite population, this would completely invert the effect of depth bounding (for brevity, we use the name 'finitisation' for this). In the practical case, of a finite population, we will be left with only finite sample effects.

However, it is more difficult to perform such a correction when EDA-GP systems include a selection operator. We can still estimate the effects of sampling, finitisation and learning (since these are independent of the specific fitness function). But the effect of selection is generally unknown – it depends directly on the problem. In practical domains, we generally do not know a closed form for the fitness function – and even when we do, it is rarely invertible. To handle these problems, we need to make simplifying assumptions. We first provide a framework for discussing distribution change in EDA-GPs, and then present simplifying assumptions and correction methods based on those assumptions. We continue, as before, to assume an infinite population to simplify discussion.

B. A Framework for EDA Distribution Transitions

Figure 3 illustrates the probability distribution transitions of EDA-GPs. It is a slightly different conceptualisation of EDA processes than the usual, which conceives of an EDA

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation



(a) Symbolic Regression (b) Max (c) Royal tree

Fig. 2. Mean Best Fitness of Scalar SG-GP without Selection for Problem Spaces Defined by Complex Recursion

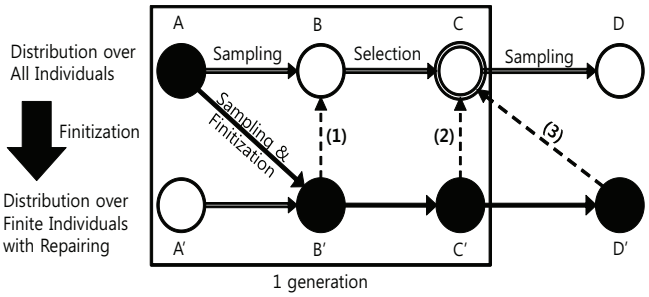


Fig. 3. A Framework for EDA Distribution Transitions

system in the order sampling – selection – learning. In the above framework, learning is viewed as an implicit operator that aims to preserve the distribution (as far as possible) while representing it. In the case of maximum likelihood estimation with infinite populations, this approximation is exact (i.e. the relative sample frequency is the distribution probability).

Nodes on the upper row denote the distribution over all individuals (including infinite-length individuals) generated by an 'ideal' EDA that can represent infinite individuals; the lower row shows the corresponding distribution over the depth-limited individuals generated by our repair mechanism.

The filled nodes denote the sequence of distributions we would observe in a finitised system with no correction mechanism. The distribution A (whether an initial prior or an intermediate generated by EDA-GP mechanisms) might be improper, but sampling with finitisation would automatically generate a proper grammar model (B'). Selecting from it would yield C', which is proper – but potentially biased. Even though C' is proper, it can still yield individuals beyond the depth bound, so further finitisation bias may be added in transiting from C' to D'.

On the other hand, an unbiased system would generate B, C and D – unbiased, but also likely to be improper. Of course, we could invert the bias via arrow (1). How would this work out? We need to invert the bias of equation 16. Denoting the components of B as $p(r)$, we treat the components of B' as $\hat{p}(r)$ and apply an inverse multiplication as in equations 33 (terminating) and 34 (non-terminating).

$$p(r) = \hat{p}(r) \frac{\frac{1-h^{\bar{d}-1}}{1-h} + h^{\bar{d}-1}}{\frac{1-h^{\bar{d}-1}}{1-h} + h^{\bar{d}-1} \frac{1}{\sum_{s \in M(A,R_T)} p(s)}} \tag{33}$$

$$p(r) = \hat{p}(r) \frac{\frac{1-h^{\bar{d}-1}}{1-h} + h^{\bar{d}-1}}{\frac{1-h^{\bar{d}-1}}{1-h}} \tag{34}$$

In a more general form, we can rewrite equation 16 and 17 in terms of vectors of probabilities over all rules r corresponding to a given nonterminal, with $\mathbf{p}_A$ consisting of $p(r)$ and $\hat{\mathbf{p}}_A$ of $\hat{p}(r)$. In this form, we can write:

$$\hat{\mathbf{p}}_A = \mathbf{F}\mathbf{p}_A \tag{35}$$

where $\mathbf{F}$ is an $|R| \times |R|$ diagonal constant transition matrix determined by the distribution A:

$$\mathbf{F}(i,j) = \begin{cases} \frac{\hat{p}(r_{A,i})}{p(r_{A,i})} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

Here i and j are the row and column numbers and $r_{A,i}$ is a rule whose LHS is A. The inverses of equations 33 and 34 are generated by multiplying both sides of equation 35 by the inverse of $\mathbf{F}$ – if it exists. If the matrix is not invertible, the degenerate entries provide no information for estimating the distribution in the C state. In this case, we set them equal to the distribution of the A state.

Some problems could arise in these cases: if finitisation generates zero probability for a given production, then the inverse will be infinite, and the product undefined. In an infinite population, this would only occur if the production could not generate trees within the depth limit – in which case, neglecting to perform the correction would leave a bias, but one that could have no practical effect. In finite populations, degeneracy could arise simply from finite sampling, a more serious problem in principle, but again, with reasonable-sized populations, this is unlikely to be a practical issue.

However, there is a more important problem: correcting to generate distribution B is pointless, because the next step still requires us to select from an un-finitised population. In general we cannot do so. So all we can hope to do is to attempt to invert the bias at (2), generating model C. We then apply sampling and finitisation again over (3) to find D', and so on.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

What bias correction should we apply at (2)? One solution is to apply that of equations 16 and 17 over transition (2) instead of (1). But there are a number of complications. The first is that we are implicitly assuming that the path B' $\rightarrow$ B $\rightarrow$ C is equivalent to that B' $\rightarrow$ C' $\rightarrow$ C – that selection commutes with inverse finitisation, or in statistical terms, the distribution effects of finitisation and of selection are independent. This is a very strong assumption. It may well be reasonable for fitness functions that are less dependent on tree complexity (symbolic regression or Boolean problems). It is highly unlikely for positional fitness functions (Royal tree, Max).

The second issue is what to do in the case of degeneracy, i.e. if a probability in C' is zero. If it was already zero at B', we can use the previous solution, leaving it zero. If the reduction to zero was in selection, similar arguments apply. Either the production was eliminated because it would be eliminated by selection in all contexts (in which case selection is acting very clearly, and we should leave it as zero), or else the only contexts in which it could be selected do not arise in finitised populations – in which case the previous arguments apply. Either way, we should not resurrect degenerate productions.

IX. Bias Amelioration Experiments

A. Simple Grammars

We start with the simple grammars and parameter settings we used in Section V. We tested the range of truncation selection ratios shown in Table II.

TABLE VI
FINAL GENERATION MEAN BEST FITNESS OF SCALAR SG-GP, 30 RUNS
(SIMPLE GRAMMARS. ρ: SELECTION RATIO;
α: THE BEST OF MEAN BEST OVER ALL GENERATIONS)

Problem (θ: max fitness)	Uncorrected Bias Final Gen.	α	Bounded Inverse Bias Final Gen.	α
Regression (θ: 0)				
$\rho = 30\%$	0.56 ± 0.00	0.24	0.01 ± 0.05	0.00
$\rho = 60\%$	0.56 ± 0.00	0.25	0.56 ± 0.00	0.02
$\rho = 90\%$	0.62 ± 0.05	0.25	0.64 ± 0.02	0.17
Max (θ: 16)				
$\rho = 30\%$	6.40 ± 0.44	6.75	8.00 ± 0.00	14.68
$\rho = 60\%$	6.00 ± 0.45	6.60	8.00 ± 0.00	15.47
$\rho = 90\%$	4.68 ± 0.63	6.60	15.73 ± 1.44	16.00
Royal tree (θ: 384)				
$\rho = 30\%$	32.0 ± 0.0	78.5	126.5 ± 17.17	185.3
$\rho = 60\%$	4.0 ± 0.0	78.5	3.1 ± 2.5	102.2
$\rho = 90\%$	4.0 ± 0.0	78.5	1.0 ± 0.0	85.3

1) Results: The mean best fitness over 30 runs is illustrated in Fig. 4, for a 30% selection ratio. The figure is supplemented by the detailed values in Table VI, which show the corresponding final generation best fitnesses (over all runs). The first column shows the mean and standard deviation of the best fitness in the final generation, while the second shows the absolute best fitness achieved in a run (over all generations).

For the symbolic regression problem, inverting the boundedness bias combined with higher selection ratios was sufficient to solve the problem (perfectly for a 30% selection ratio, and near-perfectly for a 60% selection ratio). We also saw improvements in performance for the Royal tree problem, with fitness reaching a high level for 30% selection ratio. However for the 60% and 90% selection ratios, we first see somewhat surprising results: although the highest fitness over the whole run was good, the final generation fitness was poor. That is, the best fitness must have been quite good somewhere in the run, but subsequently declined.

We see this in extreme form for the Max problem. For all systems, the best fitness attained over the generations is much better than that in the final generation. For selection ratios of 30% and 60%, quite high fitnesses are initially attained, but the fitness converges to 50% of the optimum (still much better than the uncorrected version). The less eager search of a 90% selection ratio does better, attaining a distribution with good fitness at the final generation, and achieving the optimal fitness during the run. But even here, the final best fitness has declined from the peak.

2) Discussion: Bias correction interacts with selection in complex ways. For the symbolic regression and Royal tree problems, bias correction yields substantially better results than the uncorrected form. For the Max problem, there is an improvement, but it is more equivocal, and also yields an unexpected result: that fitness improves, but then declines very substantially. This effect is also seen in the Royal tree problem, and especially at selection ratios of 60% and 90%.

We hypothesise that the effect results from the representation limitations of sSG-GP: that it is in principle unable to represent the space of good solutions as a discrete distribution, and hence cannot converge to it. We present more detailed discussion of this issue in the supplementary materials.

B. Complex Grammars

These experiments follow the same structure as the simple grammar experiments, with parameter settings as in Table II, and using the grammar shown in Table IV.

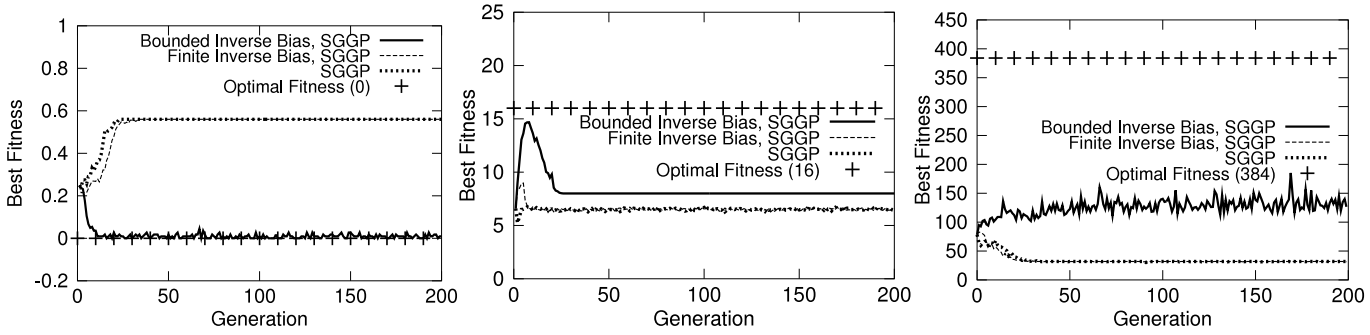
1) Results: We present the results for complex grammars in a similar way to those of Section IX. Fig. 5 corresponds to Fig. 4, and Table VII to Table VI.

TABLE VII
FINAL GENERATION MEAN BEST FITNESS OF SCALAR SG-GP, 30 RUNS
(COMPLEX GRAMMARS. ρ: SELECTION RATIO;
α: THE BEST OF MEAN BEST OVER ALL GENERATIONS)

Problem (θ: max fitness)	Uncorrected Bias Final Gen.	α	Bounded Inverse Bias Final Gen.	α
Regression (θ: 0)				
$\rho = 30\%$	0.56 ± 0.00	0.25	0.51 ± 0.09	0.00
$\rho = 60\%$	0.56 ± 0.00	0.25	0.56 ± 0.00	0.05
$\rho = 90\%$	0.62 ± 0.05	0.25	0.62 ± 0.05	0.17
Max (θ: 16)				
$\rho = 30\%$	6.38 ± 0.54	8.66	8.00 ± 0.00	15.80
$\rho = 60\%$	5.98 ± 0.57	8.66	8.00 ± 0.00	15.87
$\rho = 90\%$	4.72 ± 0.49	8.66	15.27 ± 1.82	16.00
Royal tree (θ: 6144)				
$\rho = 30\%$	6.0 ± 0.2	80.6	308.7 ± 231.8	494.0
$\rho = 60\%$	5.4 ± 1.1	80.6	142.5 ± 233.5	241.9
$\rho = 90\%$	4.3 ± 1.4	80.6	12.3 ± 3.0	85.4

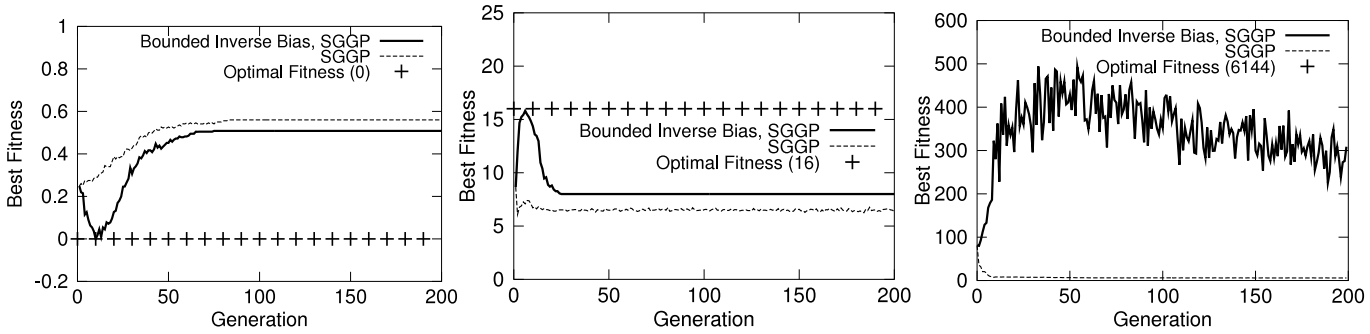
The results are more complex than those for simple grammars. Bias correction certainly improves algorithm performance in the sense that, somewhere during a run, very much

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation



(a) Symbolic Regression 30% (b) Max 30 % (c) Royal tree 30%

Fig. 4. Mean Best Fitness of Scalar SG-GP with 30% Selection for Problem Spaces Defined by Simple Recursion



(a) Symbolic Regression 30% (b) Max 30 % (c) Royal tree 30%

Fig. 5. Mean Best Fitness of Scalar SG-GP with 30% Selection for Problem Spaces Defined by Complex Recursion

better fitness values will be attained than in runs with no bias correction. However, now we see in all problems the phenomenon we previously observed with the Max problem: a peak in fitness followed by a reversion to worse results. As with the simple grammars, we believe that this results from the inability of the sSG-GP representation to capture the structure of good solutions in a converged distribution – in particular the depth-dependent structure that good solutions to Max and Royal tree require. We discuss this further in the supplementary materials.

X. Conclusions

EDA-GP systems suffer from additional sources of bias compared to other forms of EDA. For prototype-tree based EDA-GP, the major form is amplified stochastic bias [20]. Grammar-based EDA-GPs are less prone to this because the stochastic grammar model is generally learnt from larger samples because of the typically shallower depth of dependence. However, most grammars induce a countervailing sampling bias, arising from recursion and the resulting need to limit sampling to finite individuals. This bias is amplified over repetitions of the EDA sampling/learning cycle.

The grammar-induced bias can be mathematically estimated, both for simply-recursive grammars and for grammars with more complex structures. For maximum likelihood learning, a closed form can be derived for simply-recursive grammars, but it does not extend readily to complex recursions.

For the empirical part, we used a very simple EDA, scalar SG-GP, so that its behaviour might be more readily understood. We confirmed that the estimate of the bias is likely to be accurate (in the absence of selection) because inverting it eliminated the algorithm's tendency to converge to low-fitness areas of the solution space, for both simple and complex grammars. It is thus crucial, in grammar-based EDA-GPs, to take this source of bias into account in the algorithm design.

We hypothesised that bias inversion might be useful in the presence of a fitness function. In all cases, we saw substantial improvements in performance from bias inversion. For simple grammars, this was sufficient to give good performance. For more complex grammars, sSG-GP was unable to perform well, primarily because its model representation language is insufficient to represent the solution space. The inadequacy of sSG-GP's model has been recognised from its inception [14]. More complex EDA-GPs with grammar-based representations have handled symbolic regression, Max and Royal Tree problems well [47]–[51]. But they are still subject to similar recursion-induced biases. We consider this further in the next section.

The detailed analysis in this paper related to one specific form of EDA-GP, sSG-GP. But the issue of bias from recursion, and the importance of the h value of the grammar, is not so limited. h affects the bias in other EDA-GPs. Since they learn aspects of the grammar structure, the effects are complex, and more difficult to analyse. At minimum, we need awareness of h's importance, and the possibility that adjusting it (by recasting the grammar) may affect system behaviour.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

Its significance extends even further. Classical grammar-based GPs also have an h value, with a potential impact on the GP system's rate of bloat. Further study of the effect of h in grammar-based GP seems warranted. Beyond grammar-based GP, expression tree GP can be viewed as a limiting case of grammar-based GP (as Koza [3] implicitly recognised in his terminology). Thus, h is also meaningful for expression tree GP, and is affected by the relative balance between function of different arities and terminating symbols. Deeper consideration of the resulting biases may cast further light on the rates of bloat even in expression tree GP.

XI. Future Directions

In this analysis, we restricted attention to systems that learn only the model probabilities, not its structure. This was sufficient to demonstrate the existence of a problem for all grammar-based EDA-GP (because structure learning will only exacerbate the problem). But the problem remains that such systems can actually solve very few problems, because the general grammar used to define the possible solutions is not sufficient to discriminate good solutions. Practical EDA-GP systems require some form of structure learning.

If we are, in practice, to solve the problems revealed by this work, then biases associated with structure learning also require analysis. In some cases, as with the depth-annotated grammar models of vectorial SG-GP and PEEL, the direction of an extended analysis is relatively clear, if complex. For the more sophisticated structure-learning methods of more recent systems, the difficulties are more severe.

One other possible direction presents itself: the problem ultimately boils down to the use of improper distributions. If we could impose a restriction that all distributions must be proper, then all would be well. Unfortunately, it seems difficult to do this: restricting to proper distributions is likely to eliminate the very solutions we wish to find, unless we choose exactly the right proper distribution. Despite some effort, we have been unable to resolve this problem – perhaps others will find the key.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (Project No. 2012-004841). The ICT at Seoul National University provided research facilities for this study. Nguyen Xuan Hoai was partly funded for this work by The Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2014.09. This work was partly supported by the IT R&D program of MSIP/KEIT (10041807, Development of Original Software Technology for Automatic Speech Translation with Performance 90% for Tour/International Event focused on Multilingual Expansibility and based on Knowledge Learning).

References

[1] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," in *The International Conference on Machine Learning 1995 (ML-95)*, A. Prieditis and S. Russel, Eds. San Mateo, CA: Morgan Kaufmann, 1995, pp. 38–46.

[2] J. A. L. P. Larranaga, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[3] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambride, MA, USA: MIT press, 1992.

[4] Y. Shan, R. I. McKay, D. Essam, and H. A. Abbass, "A survey of probabilistic model building genetic programming," in *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2006, vol. 33, ch. 6, pp. 121–160.

[5] H. Iba, Y. Hasegawa, and T. K. Paul, *Applied Genetic Programming and Machine Learning*, ser. CRC Complex and Enterprise Systems Engineering. CRC Press, Aug. 2009.

[6] K. Kim, Y. Shan, N. X. Hoai, and R. McKay, "Probabilistic model building in genetic programming: A critical review," *Genetic Programming and Evolvable Machines*, pp. 1–53, Sep 2013.

[7] K. Yanai and H. Iba, "Probabilistic distribution models for EDA-based GP," in *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, vol. 2. Washington DC, USA: ACM Press, 25-29 Jun. 2005, pp. 1775–1776.

[8] K. Sastry and D. E. Goldberg, "Probabilistic model building and competent genetic programming," *Genetic Programming Theory and Practice*, pp. 205–220, 2003.

[9] O. Roux and C. Fonlupt, "Ant programming: or how to use ants for automatic programming," in *Proceedings of the Second International Conference on Ant Algorithms (ANTS2000)*, Brussels, Belgium, Sep. 2000, pp. 121–129.

[10] R. P. Salustowicz and J. Schmidhuber, "Probabilistic Incremental Program Evolution," *Evolutionary Computation*, vol. 5, no. 2, pp. 123–141, 1997.

[11] Y. Shan, R. I. McKay, R. Baxter, H. Abbass, D. Essam, and H. X. Nguyen, "Grammar model-based program evolution," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon, USA: IEEE Press, 20-23 Jun. 2004, pp. 478–485.

[12] C. Keber and M. G. Schuster, "Option valuation with generalized ant programming," in *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Newyork, USA: Morgan Kaufmann Publishers, 9-13 Jul. 2002, pp. 74–81.

[13] H. A. Abbass, X. Hoai, and R. I. McKay, "AntTAG: A new method to compose computer programs using colonies of ants," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, vol. 2. Honolulu, HI, USA: IEEE Press, 2002, pp. 1654–1659.

[14] A. Ratle and M. Sebag, "Avoiding the bloat with probabilistic grammar-guided genetic programming," in *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001*, ser. Lecture Notes in Computer Science, vol. 2310. Creusot, France: Springer Verlag, 29-31 Oct. 2001, pp. 255–266.

[15] R. Poli and N. F. McPhee, "A linear estimation-of-distribution gp system," in *Proceedings of the 11th European conference on Genetic Programming, EuroGP 2008*, ser. Lecture Notes in Computer Science, vol. 4971. Naples, Italy: Springer-Verlag, 26-28 Mar. 2008, pp. 206–217.

[16] M. Looks, B. Goertzel, and C. Pennachin, "Learning computer programs with the Bayesian optimization algorithm," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ser. GECCO '05, vol. 1. Washington DC, USA: ACM Press, 25-29 Jun. 2005, pp. 747–748.

[17] J. Clegg, "Combining cartesian genetic programming with an estimation of distribution algorithm," in *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*. Atlanta, GA, USA: ACM, 12-16 Jul. 2008, pp. 1333–1334.

[18] P. A. Whigham, "Grammatical bias for evolutionary learning," Ph.D. dissertation, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 1996.

[19] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[20] K. Kim and R. I. B. McKay, "Stochastic diversity loss and scalability in estimation of distribution genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 301–320, Jun. 2013.

[21] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999.

[22] Z. Chi and S. Geman, "Estimation of probabilistic context-free grammars," *Computational Linguistics*, vol. 24, no. 2, pp. 299–305, 1998.

[23] Z. Chi, "Statistical properties of probabilistic context-free grammars," *Computational Linguistics*, vol. 25, no. 1, pp. 131–160, 1999.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

[24] T. L. Booth and R. A. Thompson, "Applying probability measures to abstract languages," *IEEE Transactions on Computers*, vol. C–22, pp. 442–450, 1973.

[25] K. B. Athreya and P. E. Ney, *Branching Processes*. Springer Berlin Heidelberg, 1972.

[26] M. I. Miller and J. A. O'Sullivan, "Entropies and combinatorics of random branching processes and context-free languages," *IEEE Transactions on Information Theory*, vol. 38, no. 4, pp. 1292–1310, Jul. 1992.

[27] K. Kim, N. X. Hoai, and R. I. B. McKay, "Implicit bias in grammar-based estimation of distribution genetic programming: The effects of recursive structure," in *Proceedings of the IEEE Congress on Evolutionary Computation*. Brisbane, Austrailia: IEEE Press, June 10-15 2012, pp. 1751–1758.

[28] Y. Hasegawa and H. Iba, "A Bayesian network approach to program generation," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 750–764, Dec. 2008.

[29] R. I. B. McKay, N. Hoai, P. Whigham, Y. Shan, and M. O'Neill, "Grammar-based genetic programming: A survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3–4 (10th Anniversary Special Issue), pp. 365–396, 1 Sept 2010. [Online]. Available: http://sc.snu.ac.kr/PAPERS/gpem_giec.pdf

[30] Y. Shan, R. I. McKay, H. A. Abbass, and D. Essam, "Program evolution with explicit learning: a new framework for program automatic synthesis," in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation, CEC2003*, University of New South Wales. Canberra, Australia: IEEE Press, 8-12 Dec. 2003, pp. 1639–1646.

[31] Y. Hasegawa and H. Iba, "Latent variable model for estimation of distribution algorithm based on a probabilistic context-free grammar," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 858–878, Aug. 2009.

[32] J. L. Olmo, J. R. Romero, and S. Ventura, "A grammar based ant programming algorithm for mining classification rules," in *Proceedings of 2010 IEEE Congress on Evolutionary Computation (CEC 2010)*. Barcelona, Spain: IEEE Press, 18-23 Jul. 2010, pp. 225–232.

[33] S. Shirakawa, S. Ogino, and T. Nagao, "Dynamic ant programming for automatic construction of programs," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 3, no. 5, pp. 540–548, Sep. 2008.

[34] M. Boryczka and Z. J. Czech, "Solving approximation problems by ant colony programming," in *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*. New York, USA: Morgan Kaufmann Publishers, 9-13 Jul. 2002, p. 133.

[35] S. A. Rojas and P. J. Bentley, "A grid-based ant colony system for automatic program synthesis," in *Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference*, Seattle, Washington, USA, 26 Jul. 2004.

[36] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[37] P. Whigham, "Inductive bias and genetic programming," in *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*. Sheffield, UK: IEEE, 12-14 Sep 1995, pp. 461–466.

[38] Y. Shan, R. I. McKay, D. Essam, and J. Liu, "Modularity and position independence in EDA-GP," in *Proceedings of The Second Asian-Pacific Workshop on Genetic Programming*, Cairns, Australia, 2004.

[39] S. Silva, P. J. N. Silva, and E. Costa, "Resource-limited genetic programming: Replacing tree depth limits," in *Adaptive and Natural Computing Algorithms*. Coimbra, Portugal: Springer Vienna, 2005, pp. 243–246.

[40] E. Crane and N. McPhee, "The effects of size and depth limits on tree based genetic programming," in *Genetic Programming Theory and Practice III*, ser. Genetic Programming. Springer US, 2006, vol. 9, pp. 223–240.

[41] N. McPhee, A. Jarvis, and E. Crane, "On the strength of size limits in linear genetic programming," in *Genetic and Evolutionary Computation GECCO 2004*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3103, pp. 593–604.

[42] B. J. Ross, "Logic-based genetic programming with definite clause translation grammars," *New Generation Computing*, vol. 19, no. 4, pp. 313–337, 2001.

[43] P. A. Whigham, "Grammatically-based genetic programming," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, Tahoe City, California, USA, 9 Jul. 1995, pp. 33–41.

[44] W. B. Langdon and R. Poli, "An analysis of the MAX problem in genetic programming," in *Genetic Programming 1997: Proceedings of the Second International Conference on Genetic Programming*, CA, USA, 13-16Jul. 1997, pp. 222–230.

[45] B. Punch, D. Zongker, and E. Goodman, "The royal tree problem, a benchmark for single and multi-population genetic programming," *Advances in genetic programming*, vol. 2, pp. 299–316, 1996.

[46] A. Tsakonas, "A comparison of classification accuracy of four genetic programming-evolved intelligent structures," *Information Sciences*, vol. 176, no. 6, pp. 691–724, Mar. 2006.

[47] Y. Shan, "Program distribution estimation with grammar models," Ph.D. dissertation, University of New South Wales, Australia, 2005.

[48] Y. Shan, R. McKay, R. Baxter, H. Abbass, D. Essam, and H. Nguyen., "Grammar model-based program evolution," in *Proceedings, IEEE Congress on Evolutionary Computation*. Portland, USA: IEEE Press, June 2004, pp. 478–485.

[49] Y. Shan, R. I. McKay, H. A. Abbass, and D. Essam, "Program evolution with explicit learning: a new framework for program automatic synthesis," in *Proceedings, IEEE Congress on Evolutionary Computation*, University College, University of New South Wales, Australia. Canberra, Australia: IEEE Press, Dec 2003, pp. 1639–1646.

[50] Y. Hasegawa and H. Iba, "Estimation of distribution algorithm based on probabilistic grammar with latent annotations," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, IEEE Computational Intelligence Society. Singapore: IEEE Press, 25–28 Sep. 2007, pp. 1043–1050.

[51] ——, "Latent variable model for estimation of distribution algorithm based on a probabilistic context-free grammar," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 858–878, 2009.

Kangil Kim received his BSc in Computer Science from the Korea Advanced Institute of Science and Technology in 2006, and PhD from Seoul National University in 2012. He is working as a senior researcher in natural language processing group of Electoronics and Telecommunications Research Institute in Korea from 2012. His research interests include artificial intelligence, evolutionary computation, machine learning, and natural language processing.

Bob McKay received his BSc from the Australian National University in 1971, and his PhD in theory of computation from the University of Bristol, UK, in 1976. He has worked in CSIRO and the University of New South Wales, before joining Seoul National University, Korea, in 2005. Following his retirement in 2015, he subsequently joined the Australian National University His research interests lie in intelligent systems, evolutionary computation and ecological modelling.

Nguyen Xuan Hoai received his PhD in Computer Science from the University of New South Wales, Australia in 2005. He worked in Le Quy Don Technical University, Vietnam, then Seoul National University, Korea, before being appointed as an associate professor in computer science at Hanoi University, Vietnam, in 2010. His research interests include evolutionary computation, in particular genetic programming, and statistical machine learning.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2425420, IEEE Transactions on Evolutionary Computation

LIST OF FIGURES

LIST OF TABLES