An Analysis of the Bias of Variation Operators of Estimation of Distribution Programming

Dirk Schweim
Johannes Gutenberg University
Mainz, Germany
schweim@uni-mainz.de

Franz Rothlauf
Johannes Gutenberg University
Mainz, Germany
rothlauf@uni-mainz.de

ABSTRACT

Estimation of distribution programming (EDP) replaces standard GP variation operators with sampling from a learned probability model. To ensure a minimum amount of variation in a population, EDP adds random noise to the probabilities of random variables. This paper studies the bias of EDP's variation operator by performing random walks. The results indicate that the complexity of the EDP model is high since the model is overfitting the parent solutions when no additional noise is being used. Adding only a low amount of noise leads to a strong bias towards small trees. The bias gets stronger with an increased amount of noise. Our findings do not support the hypothesis that sampling drift is the reason for the loss of diversity.

Furthermore, we suggest using property vectors to study the bias of variation operators. Property vectors can represent the distribution of a population's relevant property, such as tree depth or tree size. The Bhattacharyya coefficient of two property vectors is a measure of the similarity of the two distributions of population properties. The results for EDP and standard GP illustrate that search bias can be assessed by representing distributions using property vectors and measuring their similarity using the Bhattacharyya coefficient.

CCS CONCEPTS

• **Mathematics of computing** → *Evolutionary algorithms*; *Probabilistic algorithms*; • **Theory of computation** → *Design and analysis of algorithms*;

KEYWORDS

Search Bias, Genetic Programming, Estimation of Distribution Programming, Estimation of Distribution Algorithm, n-Gram, Similarity

ACM Reference Format:
Dirk Schweim and Franz Rothlauf. 2018. An Analysis of the Bias of Variation Operators of Estimation of Distribution Programming. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205582

GECCO '18, July 15–19, 2018, Kyoto, Japan
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205582

1 INTRODUCTION

Estimation of distribution algorithms (EDAs) replace standard variation operators like crossover and mutation with a two-step approach: First, a probabilistic model is estimated to approximate the distribution of a population of solutions. Second, new solutions are sampled from the learned model. A variety of EDAs using different probabilistic models have been proposed in the literature [8]. Often, these EDAs were evaluated only on the quality of found solutions ("fitness values") on benchmark problems. We argue that an additional analysis can help us better understand when to apply one type of EDA model rather than another. The analysis of EDAs should be guided by what is expected from a capable EDA model. Similar to standard variation operators, an EDA model should allow us

(1) to create new solutions that have not yet been found during an EDA run,
(2) while simultaneously transferring relevant properties to the new population, so that the new solutions have similar properties to the parent solutions. Relevant properties are for example the distribution of tree depths or of sub-trees.

The ability to create new solutions is relatively easy to measure, e.g. by caching all solutions found in the search process up until then and counting the number of newly-created solutions in a generation. In contrast, it is difficult to assess if relevant properties of solutions are similarly distributed in two populations (e.g. a parent and an offspring population).

The question of how the distributions of the properties of solutions differ between subsequent populations has been discussed in the literature under the term *search bias*. A search bias exists if the underlying distributions of the properties of solutions change over multiple generations in a heuristic search [2, 14, 15]. The degree to which properties deviate between populations determines the strength of the bias. Large differences indicate a strong bias, low differences a weak bias. The bias of variation operators can be measured by performing random walks where only variation operators are applied (but no selection operators). If the properties of a population of solutions change significantly during a random walk, the used variation operators are biased, meaning that some solution characteristics are favored over others [14].

Common studies of the bias of variation operators take into account the properties of solutions such as tree depth, tree size, or fitness values. In order for each of the (subsequent) populations to be compared, the distribution of the property is estimated and a single, aggregated measure that characterizes the distribution is determined (e.g. median tree depth or best observed fitness value).

By studying the change of the aggregated measure over a number of generations, the bias of variation operators can be assessed [14].

However, focusing only on aggregated measures that represent the distributions of a solution property and ignoring the underlying distributions in the populations can lead to incorrect conclusions. For example, the distribution of a property can strongly change over time even though the median of the distribution remains constant, falsely indicating a low bias. The underlying problem is that the aggregation of a properties distribution takes place before the populations are compared: The distribution of a property is aggregated before the aggregated measures are compared between populations.

In this paper, we suggest assessing the bias of variation operators by directly comparing the distributions of a property between populations. In order to assess a potential bias, we first estimated the distributions of a property characterizing the solutions in the populations. Second, we compared these distributions between populations by applying a similarity measure (Bhattacharyya coefficient). This approach also allows us to study the similarity of two populations on nominal properties that cannot be numerically organized or ranked, like the *n*-grams of ancestors.

An *n*-gram of ancestors in a tree is the sequence of a node *i* and its $n-1$ ancestor nodes on the same branch (parent, grandparent, great-grandparent, etc.) [4]. Hemberg et al. [4] studied the dependency structure of nodes in GP parse trees over multiple generations and found that during a GP run statistical "dependency patterns" between nodes emerge as a result of the evolutionary process. *n*-grams of ancestors were found to be important because the values of nodes usually depended on the values of the associated parent and grandparent nodes. Thus, GP variation operators should preserve as many *n*-grams as possible while generating new and yet unseen feasible solutions.

In this paper, we also studied the bias of estimation of distribution programming (EDP) [16] by using properties such as tree depth, tree size, and *n*-grams of ancestors. EDP estimates the probability distribution of the solutions in a population by learning position-dependent relationships between the nodes of individual solutions. To ensure a minimum amount of variation in a population, EDP adds a small amount of random noise to the probabilities of random variables. EDP explicitly learns hierarchical dependencies between nodes in order to prevent losing relevant properties of populations during variation [16]. Thus, we expected the distribution of *n*-grams to be similar in parent and offspring populations.

However, we found that EDP has a strong bias and favors very small trees when only a low amount of random noise is used. The bias gets stronger with an increased amount of noise. In addition, the diversity in EDP is low after few generations. As a reason we find that the complexity of the EDP model is too high so that it is able to memorize the solutions after typically between 10 and 20 generations. Indeed, this seems supported by the fact that when no additional noise is used, the diversity of solutions is very low. When a higher amount of noise is added, the diversity is higher, but a strong bias towards small trees is introduced instead.

In Sect. 2, we discuss *n*-grams in the context of EDAs and how they are used in EDP. In Sect. 3, we describe different approaches of measuring bias. The experimental setup and results are presented in Sect. 4. We end the article with concluding remarks.

2 *n*-GRAMS IN ESTIMATION OF DISTRIBUTION ALGORITHMS

Since the *n*-grams of ancestors are an important property of GP solutions [4], they can also be used for probabilistic models of EDAs. Poli and McPhee [11] presented an *n*-gram EDA that works with a linear representation (genotypes are sequences of instructions). *n*-gram GP "learns and samples the joint probability of 3-grams of instructions at the same time as learning and sampling a program length distribution" [11]. The authors used variants of two standard problems (the symbolic regression problem and the lawn mower problem [9]) and found that *n*-gram GP performs only slightly worse than linear GP for small instances of the problems. For larger and more difficult instances of the problems, the *n*-gram GP showed superior performance compared to the linear GP. Therefore, the authors concluded that the *n*-gram GP has good scalability.

Hemberg et al. [4] presented Operator Free Genetic Programming (OFGP), an EDA that uses *n*-grams of ancestors as probabilistic model. Each node together with its $(n-1)$ ancestors forms an *n*-gram. The *n*-grams are used to learn hierarchical patterns, independently from the position in the parse trees. Similar to [11], the mean tree size is determined and used for sampling to decide the sizes of new trees. Hemberg et al. [4] compared their algorithm to standard GP on the Pagie-2D problem [10] and reported a performance close to standard tree-based GP. This indicates that *n*-grams are capable of capturing relevant hierarchical structures between nodes in the tree representation of the solutions.

Just like other EDAs, EDP [16] uses a probabilistic prototype tree (PPT) model [8, 12] during model building and sampling. The PPT is a full tree where the nodes have an arity equal to the maximum arity of the functions in the function set. The depth of the PPT is set to the maximum allowed depth (or to the maximum depth observed in a population). Each node of the PPT represents a random variable X_i with a multinomial probability distribution over all allowed functions in the function set F and terminals in the terminal set T (i is a node position in the PPT, N is the set of all possible node positions in the PPT, and $i \in N$) [8, 12].

The value of each random variable X_i depends on the values of other random variables $X_j, X_k, \ldots \in C_i$, with C_i being an ordered set of variables on which X_i is dependent ($j, k, \ldots \in N$ denote the respective node positions). The variables in C_i and their order is set as a prior assumption to the model, depending on the assumed dependency relationships between random variables [16]. For example, a possible assumption is that child nodes X_i are dependent variables of the corresponding parent node X_j and grandparent node X_k. The PPT model uses 3-grams of ancestors as an assumption to model variable dependencies, and therefore is able to learn position-dependent hierarchical dependencies between nodes [4, 16].

During model building in EDP, the frequencies of the different values of dependent variables X_i have to be counted, depending on the values of the corresponding independent variable(s) $X_j, X_k, \ldots \in C_i$ for each solution in a population [16]. If the corresponding independent variables are not available, these are set to "null". For example, when using 3-grams of ancestors, the root nodes do not have parent and grandparent nodes, and the nodes at depth 1 do not have grandparent nodes.

The conditional probabilities of each random variable are initialized (e.g. uniformly) and are updated in every generation of the algorithm, based on the observed relative frequencies of node values in the solutions of a population. The observed relative frequencies are used as an estimate of the underlying conditional probabilities of node values. New solutions are sampled using the estimated probabilities [8, 16].

With S denoting the multiset of solutions to a given problem (a "population"), EDP estimates the conditional probabilities for the dependent variable X_i as

$$P(X_i = x | C_i = c) = \frac{\sum_{s \in S} \delta}{\sum_{s \in S} \gamma}, \tag{1}$$

where

$$\delta = \begin{cases} 1 & \text{if } X_i = x \text{ and } C_i = c \text{ in the individual } s, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

and

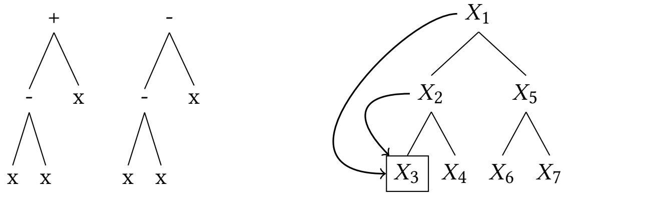
$$\gamma = \begin{cases} 1 & \text{if } C_i = c \text{ in the individual } s, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Yanai and Iba [16] recommend modifying the estimated probabilities before sampling new solutions from the model, by adding a small amount of uniformly distributed noise. The uniform random noise changes the estimated probabilities independently from the observed node values in a population. The intention is to compensate for a certain amount of variation in the population data [16]. With $\alpha \in [0, 1]$ controlling the amount of added noise, the corrected probabilities $P'(X_i = x | C_i = c)$ are calculated as

$$P'(X_i = x | C_i = c) = (1 - \alpha) \times P(X_i = x | C_i = c) + \frac{\alpha}{|F \cup T|}. \tag{4}$$

Figure 1 and Table 1 give an example. We assume a population size of $|S| = 2$ (Figs. 1a and 1b show the two individuals), the function set $F = \{+, -\}$, and the terminal set $T = \{x\}$. We use 3-grams of ancestors to model dependencies between variables. Figure 1c shows the PPT model with variable X_3 depending on variables X_1 and X_2. Table 1 shows the resulting estimated distribution $P(X_3 = x_3 | C_3 = \{X_1 = x_1, X_2 = x_2\})$ and the corrected probabilities $P'(X_3 = x_3 | C_3 = \{X_1 = x_1, X_2 = x_2\})$ for $\alpha = 0.1$. For example, the frequency of the combination of $X_3 = x$, $X_1 = +$, and $X_2 = -$ is 1 (observed in tree 1). Since the combination $C_3 = \{X_1 = +, X_2 = -\}$ occurs only once in the population, $\gamma = 1$. Thus, $P(X_3 = x | C_3 = \{X_1 = +, X_2 = -\}) = 1/1 = 1$ and $P'(X_3 = x | C_3 = \{X_1 = +, X_2 = -\}) = 0.9 \times 1 + \frac{0.1}{3} \approx 0.933$ (for $\alpha = 0.1$ and $|F \cup T| = 3$).

Besides being used as an EDA model, n-grams of ancestors can also characterize a population of solutions. In that case, we inspect all possible n-grams of ancestors and count how often they occur in a population of solutions, independent from their position in the trees. Table 2 lists the frequency of 3-grams of ancestors for the two example trees. For example, the value of the root node of tree 1 is +. Since the root has no parent or grandparent nodes, these values are set to "null" and the frequency of this 3-gram is 1.



(a) tree 1 **(b) tree 2** **(c) Prototype tree with dependencies of variable X_3**

Figure 1: Example (based on [16])

Table 1: Estimated $P(X_3 = x_3 | C_3 = \{X_1 = x_1, X_2 = x_2\})$ for EDP

X_1	X_2	X_3	Frequency	P	P'
+	-	x	1	100 %	93.33 %
+	-	+	0	0 %	3.33 %
+	-	-	0	0 %	3.33 %
-	-	x	1	100 %	93.33 %
-	-	+	0	0 %	3.33 %
-	-	-	0	0 %	3.33 %

Table 2: Frequencies of n-grams of ancestors for example trees 1 and 2

grandparent	parent	child	frequency
null	null	+	1
null	null	-	1
null	null	x	0
null	+	+	0
null	+	-	1
null	+	x	1
null	-	+	0
null	-	-	1
null	-	x	1
+	+	+	0
+	+	-	0
+	+	x	0
+	-	+	0
+	-	-	0
+	-	x	2
-	+	+	0
-	+	-	0
-	+	x	0
-	-	+	0
-	-	-	0
-	-	x	2

3 MEASURING THE BIAS OF OPERATORS

In this section, we will discuss the different ways of assessing a potential bias of the variation operators and suggest directly comparing the distributions of a solution property between populations.

3.1 Aggregated Bias Measures

Often, the bias of variation operators is assessed over multiple successive generations of a random walk by comparing relevant properties of the populations' solutions. Relevant properties of trees are for example tree depth, tree size, or tree density [14]. Kim et al. [6, 7] assessed the bias of variation operators by studying the change of entropy and fitness values during a random walk.

After estimating the distribution of a relevant property in a population, a value is typically determined that characterizes the distribution. Usually, a measure such as the median, the average, the quartiles, the minimum of a distribution, or the maximum of a distribution is used (e.g. best observed fitness value [6, 7] or median tree size [14]). The selected measure is expected to represent the underlying distribution of the property.

However, comparing populations using only single aggregated measures can create problems. Indeed, drawing conclusions from the observed aggregated measures can be misleading. For example, the distribution of a property can change fundamentally without affecting the median, the average, or the minimum and maximum values. Furthermore, aggregated measures such as the average of a distribution can be strongly affected by, for example, outliers. The underlying problem is that the aggregation takes place before the populations are compared. Furthermore, comparing the distributions of a nominal property (e.g. n-grams of ancestors) by using standard statistical aggregation measures such as median, average, minimum, or maximum is not possible since nominal properties cannot be ranked or organized numerically.

3.2 Property Vector of Populations

To overcome these limitations, we suggest to assessing the search bias by directly comparing the distributions of a property between populations. The approach is based on the following two ideas:

(1) The distribution of a property characterizing the solutions in a population can be represented using a k-dimensional property vector v.
(2) Given two k-dimensional property vectors, we can compare their similarity by using standard vector similarity metrics. We thus obtain a measure of similarity between two populations based on the distributions of the property. When performing a random walk, this similarity measure is an indicator for the strength of the bias of the variation operators.

The question of how to represent a distribution as a property vector depends on the type of random variable used. We distinguish between discrete and continuous random variables. For example, the tree depth d is a discrete value with $d \in \mathbb{N}_0$, whereas fitness values f of solutions are often defined as continuous variables over a range of possible values, e.g. $f \in \mathbb{R}_{\geq 0}$.

For distributions of a discrete random variable X^d, D denotes the set of $k = |D|$ different values of X^d. Thus, we can represent the distribution with a k-dimensional vector $v = (v_1, v_2, \dots, v_k)$. We set each component v_i to the observed relative frequency $p(e_i)$ of each possible value $e_i \in D$ of the random variable X^d (for all $i = 1, 2, \dots, k$). For example, we characterize a population by the tree depths of the solutions. In that case, D is the set of tree depths that can occur in the population, $k = |D|$ is the number of different possible tree depths in the population, and $p(e_i)$ is the relative frequency of a particular tree depth in the population. We assume a population where individuals have a maximum depth of 2 so that $D = \{0, 1, 2\}$. We further assume that all trees in the population have a depth of 2. Thus, the resulting property vector is $v = (0, 0, 1)$.

To determine a k-dimensional property vector v to represent distributions of a continuous random variable X^c, we partition the interval of possible values of X^c into k different non-overlapping intervals e_i ($i = 1, 2, \dots, k$). The components v_i of the property vector are set to the observed relative frequencies $p(e_i)$ of the values of X^c in the intervals e_i (for all $i = 1, 2, \dots, k$). For example, we characterize a population by the fitness value f of the solutions. We set $k = 2$ and partition the interval of possible fitness values $f \in [0, 100]$ in two intervals, e.g. [0,50),[50,100]. For each of the two intervals, we count the relative frequency of the fitness values of the individuals in the population. If we assume three individuals with fitness values $f_1 = 10$, $f_2 = 30$, and $f_3 = 90$, we obtain $v = (2/3, 1/3)$.

For continuous random variables, a proper choice of interval sizes is important to be able to properly characterize a distribution. If the intervals chosen are too large, many observations will lie within the same interval and no differences between distributions will be observed. If the intervals chosen are too small, too many intervals exist and each interval obtains only a few observations of the random variable. It is possible to define intervals of different sizes, but we recommend all intervals to have the same size when comparing two vectors by applying a similarity metric.

The property vector v represents the distribution of a property in a population of solutions. It can also represent the distributions of nominal values, for example 3-grams of ancestors. When determining a property vector v for 3-grams of ancestors, we need to determine an ordered set D of k different 3-grams and count the relative frequencies of the 3-grams in the solutions of a population. For the example from Table 2, the ordered set D contains 21 different 3-grams of ancestors. The values of v_i ($i = 1, 2, \dots, 21$) are the relative frequencies of the different 3-grams in the population.

3.3 Measuring the Similarity of Property Vectors

To compare the similarity of two vectors, a number of well-established methods exist [3]. To measure the similarity between two property vectors a and b characterizing the properties of two populations, we suggest using the Bhattacharyya coefficient [1].

Given two k-dimensional property vectors a and b with components a_i and b_i ($i = 1, 2, \dots, k$), we calculate the Bhattacharyya coefficient $B(a, b)$ of the two vectors as

$$B(a, b) = \sum_{i=1}^{k} \sqrt{a_i \times b_i}. \tag{5}$$

With $a_i, b_i \in [0, 1]$, the Bhattacharyya coefficient is restricted to values between zero and one ($B(a, b) \in [0, 1]$). $B = 0$ indicates no similarity between the two distributions; $B = 1$ indicates identical distributions [1].

Table 3: GP and EDP Parameters

population size	250
initialization	ramped-half-and-half (range 2–6)
selection	random
replacement	generational
generations	100
max. tree depth	8
terminal set	$T = \{x, y, 1.0\}$
function set	$F = \{+, -, *, /, sin, cos, exp, log\}$
variation operator	*GP*: standard sub-tree crossover (internal node bias: 90% functions, 10% terminals) *EDP*: model building and sampling with $\alpha \in \{0.00, 0.01, 0.10, 0.50, 1.00\}$

4 EXPERIMENTAL RESULTS

In this section, we will study and compare the bias of the variation operators used in EDP and standard GP using various bias measures.

4.1 Experimental Setup

To assess the bias of the EDP and GP variation operators, we performed random walks. In random walks, only variation operators are used (but no selection operators). Thus, no fitness comparisons between solutions are necessary and we do not not need to define a fitness function. The parameters of EDP and GP are set to the values listed in Table 3.

EDP adds an amount of uniformly distributed noise to the conditional probabilities of the dependent variables X_i (Eq. (4)) which is controlled by the parameter α. We applied five different values of $\alpha \in \{0.00, 0.01, 0.10, 0.50, 1.00\}$. For EDP's probability model, we assumed that the value of a node depends on the values of its parent and grandparent node (3-grams of ancestors). Although [16] recommended to weight solutions by their fitness during model building, we did not follow this recommendation as it would 1) introduce an additional bias and 2) make the definition of a fitness function necessary. Analogously, we did not use elitism as suggested in [16].

We compared the bias of variation operators for standard GP and EDP with five different values of α. For each configuration, we performed 100 test runs. Each run was terminated after 100 generations.

4.2 Aggregated Bias Measures

We analyzed the bias of the variation operators by studying the behavior of aggregated values of a distribution such as median tree depth or size [14] during a random walk. We studied three types of aggregated bias measures:

1. The number of newly-created solutions in a generation. A tree is a new solution if it has not been found in the same or previous generations of a particular run. In order to measure the number of new solutions in a generation, we had to store all solutions generated during a run.
2. The median depth of the trees in a population. In addition, we measured the 25th and 75th percentile of the distribution.
3. The median size of the trees in a population. In addition, we measured the 25th and 75th percentile of the distribution.

Figure 2 plots the number of new solutions in each population (top), the median tree depth (middle), and the median tree size (bottom) over the number of generations for standard GP and EDP with different values of α. For depth and size, we also plotted the 25th and 75th percentiles. All results were averaged over 100 runs.

We found large differences in the number of newly-generated solutions for the different variation operators (Fig. 2, top row). In each new population of 250 individuals, GP's crossover operator generates on average between 150 and 220 new solutions. In contrast, EDP variation operators are not able to generate a reasonable number of new solutions. After only a few generations (typically between 10 and 20), the number of new solutions per generation is very low and ranges between 0 ($\alpha = 0$) and 75 new solutions ($\alpha = 1$). Thus, the EDP model building and sampling process is not able to generate new, diverse solutions. Instead, already visited solutions are again re-sampled (especially with low values of α).

Kim et al. [5–7] recognized that EDP populations have problems with low diversity and suggested "sampling drift" as a possible reason for the rapid loss of diversity [5–7]. Analogously to classical drift, sampling drift describes that a random variable "does not take one of its allowed values anywhere in the entire population" [13]. In this case "that value can never be restored" [13] since the model estimates a probability of zero for this value of the random variable. For an EDA that uses a PPT model, the number of tree instances (size of the sample) from which position-dependent probabilities can be learned becomes smaller as the depth of a node in the PPT increases. Thus, the sampling size becomes generally smaller as the depth of a node in the PPT increases. For the root node of the PPT, the sample size is equal to the population size. For nodes deeper in the PPT, the sample size becomes smaller since some solutions do not have nodes at the particular depth. For nodes at maximum depth, usually only few trees can be used to estimate the probabilities of the variables. Since the size of the sample decreases with depth, Kim et al. [5–7] recognized that the strength of sampling drift increases with the depth of a random variable in the PPT: "The size of the sample actually used to generate meaningful instructions reduces (exponentially) with depth" [5–7].

Although sampling drift can be a problem in PPT models, our findings do not support the hypothesis that this is the reason for the rapid loss of diversity. We argue that for $\alpha > 0$, variables always have a small probability of being chosen, which means that no sampling drift occurs. For high values of α, e.g. $\alpha = 1$, EDP also strongly suffers from a rapid loss of diversity (Fig. 2, top row), which can not be explained by sampling drift since all the values of the variables are sampled with approximately equal probability over all generations.

For EDP with $\alpha = 0$, the number of new solutions drops to about zero within the first 10 to 15 generations. However, the median tree size as well as the median tree depth remains relatively stable at values of around 10 and 4, respectively (Fig. 2). This indicates that EDP's probability model memorizes the initial solutions and samples them again and again with high probabilities. This behavior suggests that the complexity of the model is too high and that the model is overfitting the solutions of the initial population. The introduction of a random bias ($\alpha > 0$) slightly reduces the overfitting behavior of the model since the number of new solutions increases,

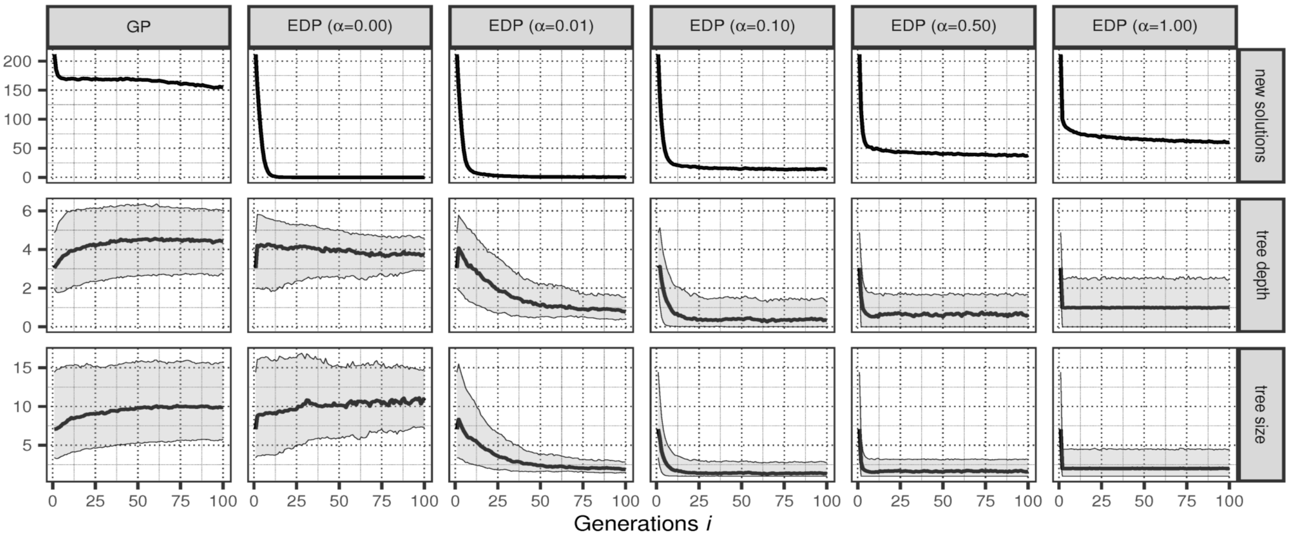


Figure 2: Properties of EDP and GP populations over the number of generations of a random walk

but it also introduces an additional strong bias towards very short trees.

Since we used the same initialization method for all six algorithms (Table 3), the median tree depth and median tree size was initialized to the same values of 3 and 7.5, respectively. For $\alpha \geq 0.01$, the median tree depth and size drops fast over the first generations and converges to low values of a median depth of 1 and a median size of 2 (Fig. 2, middle and bottom rows). Thus, many of the sampled trees are just a terminal node or only one function with one or two terminals. At the extreme case $\alpha = 1$, where the probabilities $P'(X_i = x|C_i = c)$ are completely random, the median tree depth is 1 and the median size is 2 in all populations sampled from the model.

In contrast, for EDP with $\alpha = 0$, the depth increases to a median between approximately 4 and 5 and the median size also increases to about 10. For GP, the median tree depth and size behaves similarly as for EDP with $\alpha = 0$. In contrast with GP, in EDP ($\alpha = 0$) the differences between the 75th and 25th percentiles slowly become smaller, indicating a lower variation in the population. A possible reason for this relatively small bias is sampling drift. Overall, EDP ($\alpha = 0$) and GP show similar behavior for tree depth and size and the bias is relatively weak.

In summary, the variation operators of EDP with $\alpha \geq 0.01$ have a very strong bias towards small trees. This bias cannot be a result of sampling drift since bias is also strong for $\alpha = 1$. Thus, we hypothesize that adding uniform noise favors the selection of terminals (in comparison to functions), which, in turn, leads to the strong bias towards small trees.

4.3 Property Vectors

We studied the bias of EDP and standard GP using a property vector v (see Sect. 3.2) representing the distributions of tree depths, tree sizes, and 3-grams of ancestors. We measured the bias by comparing the properties of the initial population with the properties of the ith population of a random walk. We limited the number of generations to 20 since the properties of the populations remain stable for larger numbers of generations. Again, all values were averaged over 100 runs. Tree size, depth and 3-grams are discrete variables. Since the maximum depth is 8, we set the dimension of the property vector v_d to $k = 9$. Similarly, the dimension of the size property vector v_s was set to the maximum possible tree size, and the dimension of the 3-gram property vector v_g was set to the number of possible different 3-grams (using the function and terminal set in Table 3).

Figure 3 plots the change of the distribution of tree depth over the number of generations i. The top of Fig. 3 plots $1-B(v_d^0, v_d^i)$ over the generations i, where $B(v_d^0, v_d^i)$ is the Bhattacharyya coefficient of v_d^0 and v_d^i, which are property vectors representing the distribution of tree depths at generation 0 and generation i, respectively. A value of zero indicates no bias. The bottom of Fig. 3 plots the difference $\widetilde{v}_d^i - \widetilde{v}_d^0$ between the median depth $\widetilde{v}_d^i$ of the trees in generation i and median depth $\widetilde{v}_d^0$ of the trees in generation 0. A value of zero indicates that the median of the distribution of tree depths does not change between the two populations.

For GP, both plots indicate a weak bias over the number of generations. For EDP with $\alpha = 0$, the differences between the medians (lower plot) show that the depth increased by 1 in the second generation and then remains constant. However, the Bhattacharyya coefficient indicates that the distributions become more dissimilar compared to the initial population over the number of generations. This is in line with the results plotted in Fig. 2, which show that the range between the 75th and 25th percentiles slowly becomes smaller, indicating a less diverse population.

For EDP with $\alpha = 0.01$, $\widetilde{v}_d^i - \widetilde{v}_d^0$ suggests a reduction of the bias over the first generations with a minimum at generation 10. The bias then seems to become larger again. In contrast, the Bhattacharyya coefficient shows (in line with Fig. 2) that the distributions become more dissimilar over the number of generations. For EDP with

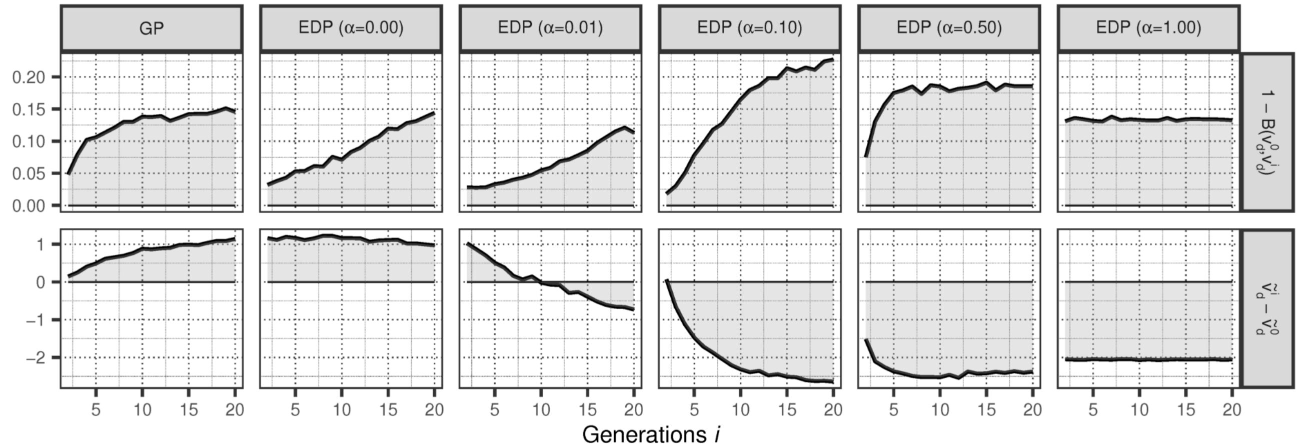


Figure 3: The similarity (measured using the Bhattacharyya coefficient) of the distribution of tree depths between the first and *i*th generation (top) and the difference between the median tree depths of the first and *i*th generation (bottom) over the number *i* of generations

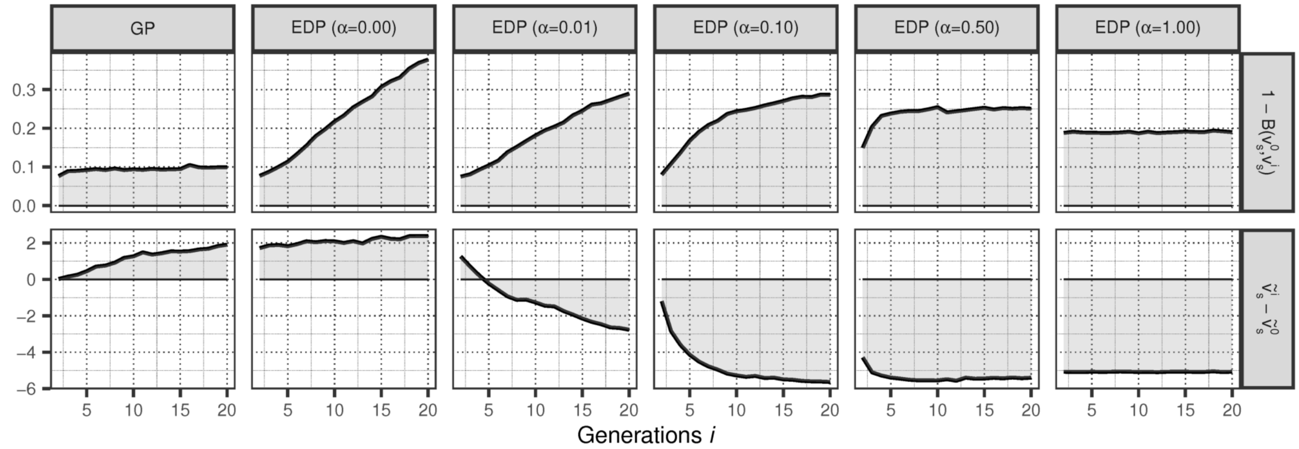


Figure 4: The similarity (measured using the Bhattacharyya coefficient) of the distribution of tree sizes between the first and *i*th generation (top) and the difference between the median of tree sizes of the first and *i*th generation (bottom) over the number *i* of generations

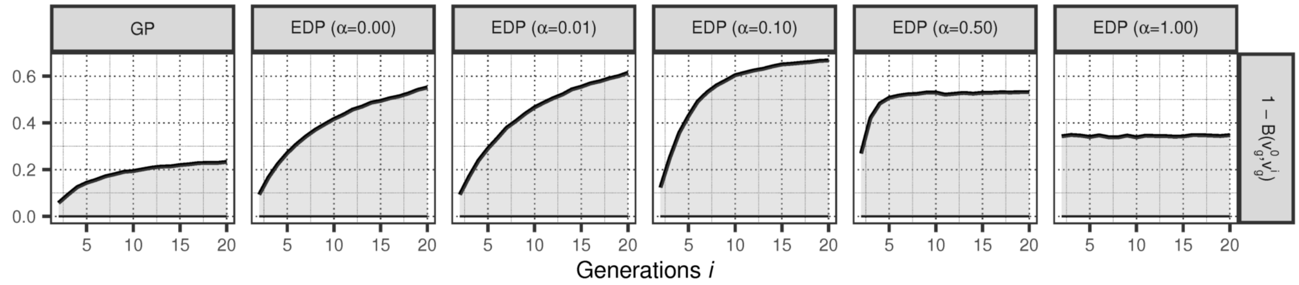


Figure 5: The similarity (measured using the Bhattacharyya coefficient) of the distribution of 3-grams of ancestors between the first and *i*th generation over number *i* of generations

$\alpha \geq 0.1$, Fig. 2 shows a strong bias towards short trees after the first generation. Thus, these populations are very dissimilar from the initial population generated by ramped-half-and-half (Fig. 3). For EDP with $\alpha = 1$, each generation is sampled from the same random distribution. Thus, the difference between the populations does not change over multiple generations.

Analogously to Fig. 3, Fig. 4 plots the change of the distribution of tree size over the number of generations i. The top of Fig. 3 plots $1 - B(v_s^0, v_s^i)$ over generations i, where $B(v_s^0, v_s^i)$ is the Bhattacharyya coefficient of v_s^0 and v_s^i, which are property vectors representing the distribution of tree sizes at generation 0 and generation i, respectively. The bottom of Fig. 4 plots the difference $\widetilde{v}_s^i - \widetilde{v}_s^0$ between the median size $\widetilde{v}_s^i$ of the trees in generation i and median size $\widetilde{v}_s^0$ of the trees in generation 0.

The results are similar to the ones for tree depths, with one exception. For GP, $\widetilde{v}_s^i - \widetilde{v}_s^0$ suggests a constant bias that leads to an increasing dissimilarity between a population and the initial population over the number of generations. In contrast, the Bhattacharyya coefficient $B(v_s^0, v_s^i)$ shows a change of the distribution from the initial to the first population. However, after this, we no longer observe any bias, and the similarity between the distributions at generations 0 and i remains constant.

Finally, Fig. 5 plots $1 - B(v_g^0, v_g^i)$ over the generations i, where $B(v_g^0, v_g^i)$ is the Bhattacharyya coefficient of v_g^0 and v_g^i, which are property vectors representing the distribution of 3-grams of ancestors at generation 0 and generation i, respectively. The plots indicate that the bias of the populations is lowest for GP. New populations generated by standard GP variation operators are more similar to the initial population in comparison to EDP. Independently of α, all EDP variants have a strong bias. For EDP with $\alpha < 1$, the bias is strong in the first generations and then becomes smaller since only small trees can then be sampled, and no new solutions are found. For EDP with $\alpha = 1$, the bias is strong in the first generation, but remains close to zero after that. This is expected as each generation was sampled from the same random distribution independently from the first population and only very small trees are sampled.

5 CONCLUSIONS

The contribution of this paper is twofold. First, it suggests to use property vectors to measure the similarity of populations. Property vectors represent the distribution of a relevant property of a population such as tree depth, tree size, or n-grams of ancestors. The similarity between two populations can be measured using the Bhattacharyya coefficient of two property vectors. We illustrated for standard GP and EDP how property vectors can be used to assess the bias of variation operators and showed that n-grams of ancestors are adequate to measure the amount of bias. We encourage other researchers to use this approach to compare the distributions of properties between populations in other contexts, e.g. to quantify selection pressure or to study selection operators.

Second, we studied the bias of EDP's variation operators. EDP replaces standard GP variation operators by sampling from a probabilistic model that learns hierarchical dependencies between nodes (3-grams of ancestors). In order to ensure a certain amount of variation in a population, EDP adds random noise to the probabilities of the random variables. We found that the variation operator of EDP has a strong bias towards very small trees when only a low amount of additional noise is used. The bias gets stronger for higher amounts of additional noise. We hypothesized that adding uniform noise favors the selection of terminals (in contrast to functions), which, in turn, leads to a strong bias towards short trees. Although the strong bias leads to low population diversity, it cannot be the result of sampling drift [5–7] since bias is strongest for a high amount of noise. When using EDP with no random noise ($\alpha = 0$), the probability model learns the initial solutions well, the diversity still is low, suggesting that the complexity of the EDP model is too high and the model is overfitting the parent solutions.

Future studies should analyze how the strong bias of EDP affects its ability to solve problems. We expect EDP to have issues solving problems where high-quality solutions are large trees.

REFERENCES

[1] Anil K. Bhattacharyya. 1946. On a Measure of Divergence between Two Multinomial Populations. *The Indian Journal of Statistics* 7, 4 (1946), 401–406.

[2] Richard A. Caruana and J. David Schaffer. 1988. Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proceedings of the Fifth International Conference on Machine Learning*, John Laird (Ed.). Morgan Kaufmann, San Mateo, CA, 153–161.

[3] Sung-Hyuk Cha. 2007. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences* 1, 4 (2007), 300–307.

[4] Erik Hemberg, Kalyan Veeramachaneni, James McDermott, Constantin Berzan, and Una-May O'Reilly. 2012. An Investigation of Local Patterns for Estimation of Distribution Genetic Programming. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation (GECCO '12)*. ACM, New York, NY, USA, 767–774.

[5] Kangil Kim. 2012. *Analysis of Stochastic Bias in Estimation of Distribution Genetic Programming*. Ph.D. Dissertation. Seoul National University.

[6] Kangil Kim and Robert I. (Bob) McKay. 2013. Stochastic Diversity Loss and Scalability in Estimation of Distribution Genetic Programming. *IEEE Transactions on Evolutionary Computation* 17, 3 (2013), 301–320.

[7] Kangil Kim, Robert I. (Bob) McKay, and Dharani Punithan. 2010. Sampling Bias in Estimation of Distribution Algorithms for Genetic Programming Using Prototype Trees. In *Pacific Rim International Conference on Artificial Intelligence (PRICAI 2010)*, Byoung-Tak Zhang and Mehmet A. Orgun (Eds.). Springer, Berlin, Heidelberg, 100–111.

[8] Kangil Kim, Yin Shan, Xuan Hoai Nguyen, and Robert I. (Bob) McKay. 2014. Probabilistic Model Building in Genetic Programming: A Critical Review. *Genetic Programming and Evolvable Machines* 15, 2 (2014), 115–167.

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

[10] Ludo Pagie and Paulien Hogeweg. 1997. Evolutionary Consequences of Coevolving Targets. *Evolutionary Computation* 5, 4 (1997), 401–418.

[11] Riccardo Poli and Nicholas Freitag McPhee. 2008. A Linear Estimation-of-Distribution GP System. In *Proceedings of the 11th European conference on Genetic Programming (EuroGP 2008)*, Michael O'Neill, Leonardo Vanneschi, Steven Gustafson, Anna Isabel Esparcia Alcázar, Ivanoe De Falco, Antonio Della Cioppa, and Ernesto Tarantino (Eds.). Springer, Berlin, Heidelberg, 206–217.

[12] Rafal Salustowicz and Jürgen Schmidhuber. 1997. Probabilistic Incremental Program Evolution. *Evolutionary Computation* 5, 2 (1997), 123–141.

[13] Jonathan L Shapiro. 2006. Diversity Loss in General Estimation of Distribution Algorithms. In *Parallel Problem Solving from Nature - PPSN IX*, Thomas Philip Runarsson, Hans-Georg Beyer, Edmund Burke, Juan J Merelo-Guervós, L Darrell Whitley, and Xin Yao (Eds.). Springer, Berlin, Heidelberg, 92–101.

[14] Ann Thorhauer and Franz Rothlauf. 2015. On the Bias of Syntactic Geometric Recombination in Genetic Programming and Grammatical Evolution. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*, Sara Silva (Ed.). ACM Press, New York, NY, 1103–1110.

[15] Peter A. Whigham. 1996. Search Bias, Language Bias and Genetic Programming. In *Genetic Programming 1996: Proceedings of the First Annual Conference on Genetic Programming*, John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo (Eds.). MIT Press, Cambridge, MA, 230–237.

[16] Kohsuke Yanai and Hitoshi Iba. 2003. Estimation of Distribution Programming Based on Bayesian Network. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003)*, Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Robert I. (Bob) McKay, Daryl Essam, and Tom Gedeon (Eds.). IEEE Press, Canberra, 1618–1625.