Bounding Bloat in Genetic Programming

Benjamin Doerr
Laboratoire d'Informatique (LIX)
École Polytechnique
Palaiseau, France

Timo Kötzing
Hasso Plattner Institute
Potsdam, Germany

J. A. Gregor Lagodzinski
Hasso Plattner Institute
Potsdam, Germany

Johannes Lengler
ETH Zürich
Zürich, Switzerland

ABSTRACT

While many optimization problems work with a fixed number of decision variables and thus a fixed-length representation of possible solutions, genetic programming (GP) works on variable-length representations. A naturally occurring problem is that of bloat (unnecessary growth of solutions) slowing down optimization. Theoretical analyses could so far not bound bloat and required explicit assumptions on the magnitude of bloat.

In this paper we analyze bloat in mutation-based genetic programming for the two test functions ORDER and MAJORITY. We overcome previous assumptions on the magnitude of bloat and give matching or close-to-matching upper and lower bounds for the expected optimization time.

In particular, we show that the (1+1) GP takes (i) $\Theta(T_{\text{init}} + n \log n)$ iterations with bloat control on ORDER as well as MAJORITY; and (ii) $O(T_{\text{init}} \log T_{\text{init}} + n(\log n)^3)$ and $\Omega(T_{\text{init}} + n \log n)$ (and $\Omega(T_{\text{init}} \log T_{\text{init}})$ for $n = 1$) iterations without bloat control on MAJORITY.

CCS CONCEPTS

•Theory of computation →Optimization with randomized search heuristics; Genetic programming; Theory of randomized search heuristics;

KEYWORDS

Genetic Programming, Mutation, Theory, Run Time Analysis

1 INTRODUCTION

While much work on nature-inspired search heuristics focuses on representing problems with strings of a fixed length (simulating a genome), genetic programming considers trees of variable size. One of the main problems when dealing with a variable-size representation is the problem of *bloat*, meaning an unnecessary growth of representations, exhibiting many redundant parts and slowing down the search.

GECCO '17, Berlin, Germany
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. 978-1-4503-4920-8/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3071178.3071271

In this paper we study the problem of bloat from the perspective of run time analysis. We want to know how optimization proceeds when there is no explicit bloat control, which is a setting notoriously difficult to analyze formally: Previous works were only able to give results conditional on strong assumptions on the bloat (such as upper bounds on the total bloat), see [NUW13] for an overview.

We use recent advances from drift theory as well as other tools from the analysis of random walks to bound the behavior and impact of bloat, thus obtaining unconditional bounds on the expected optimization time even when no bloat control is active.

Our focus is on mutation-based genetic programming (GP) algorithms, which has been a fruitful area for deriving run time results in GP. We will be concerned with the problems ORDER and MAJORTIY as introduced in [GO98]. This is in contrast to other theoretical work on GP algorithms which considered the PAC learning framework [KNS11] or the Max-Problem [KSNO12] as well as Boolean functions [MMM13, MM14, MO16].

Individuals for ORDER and MAJORTIY are binary trees, where each inner node is labeled J (short for *join*, but without any associated semantics) and leaves are labeled with variable symbols; we call such trees *GP-trees*. The set of variable symbols is $\{x_i \mid i \leq n\} \cup \{\overline{x}_i \mid i \leq n\}$, for some n. In particular, variable symbols are paired (x_i is paired with $\overline{x}_i$). We say that in a GP-tree t a leaf u comes *before* a leaf v if u comes before v in an in-order parse of the tree.

For the ORDER problem fitness is assigned to GP-trees as follows: we call a variable symbol x_i *expressed* if there is a leaf labeled x_i and all leaves labeled $\overline{x}_i$ do not come before that leaf. The fitness of a GP-tree is the number of its expressed variable symbols x_i.

For the MAJORITY problem, fitness is assigned to GP-trees as follows. We call a variable symbol x_i *expressed* if there is a leaf labeled x_i and there are at least as many leaves labeled x_i as there are leaves labeled $\overline{x}_i$ (the positive instances are in the majority). Again, the fitness of a GP-tree is the number of its expressed variable symbols x_i.

A first run time analysis of genetic programming on ORDER and MAJORITY was conducted in [DNO11]. This work considered the algorithm (1+1) GP proceeding as follows. A single operation on a GP-tree t chooses a leaf u of t uniformly at random and randomly either relabels this leaf (to a random variable symbol), deletes it (i.e. replacing the parent of u with the sibling of u) or inserts a leaf here (i.e., replaces u with an inner node with one randomly labeled child and u as the other child, in random order). The (1+1) GP is provided with a parameter k which determines how many such operations make up an atomic mutation; in the simplest case with

Table 1: Summary of best known bounds. Note that $T_{\max}$ denotes the maximal size of the best-so-far tree in the run until optimization finished (we consider bounds involving $T_{\max}$ as conditional bounds).

Problem	k	Without Bloat Control	With Bloat Control
ORDER	1	$O(nT_{\max})$, [DNO11]	$\Theta(T_{\text{init}} + n \log n)$, [Neu12]
	$1 + \text{Pois}(1)$	$O(nT_{\max})$, [DNO11]	$\Theta(T_{\text{init}} + n \log n)$, Theorem 4.1
MAJORITY	1	$O(T_{\text{init}} \log T_{\text{init}} + n \log^3 n)$, Theorem 5.2 $\Omega(T_{\text{init}} \log T_{\text{init}})$, $n = 1$, Theorem 5.1 $\Omega(T_{\text{init}} + n \log n)$, Theorem 5.1	$\Theta(T_{\text{init}} + n \log n)$, [Neu12]
	$1 + \text{Pois}(1)$	$O(T_{\text{init}} \log T_{\text{init}} + n \log^3 n)$, Theorem 5.2 $\Omega(T_{\text{init}} \log T_{\text{init}})$, $n = 1$, Theorem 5.1 $\Omega(T_{\text{init}} + n \log n)$, Theorem 5.1	$\Theta(T_{\text{init}} + n \log n)$, Theorem 4.1

$k = 1$, but a random choice of $k = 1 + \text{Pois}(1)$ (where $\text{Pois}(1)$ denotes the Poisson distribution with parameter $\lambda = 1$) is also frequently considered. The (1+1) GP then proceeds in generations with a simple mutation/selection scheme (see Algorithm 1).

A straightforward version of bloat control for this algorithm was introduced in [LP02] as *lexicographic parsimony pressure*. Here the algorithm always prefers the smaller of two trees, given equal fitness. For this [Neu12] was able to give tight bounds on the optimization time in the case of $k = 1$: in this setting no new redundant leaves can be introduced. The hard part is now to give an analysis when $k = 1 + \text{Pois}(1)$, where bloat can be reintroduced whenever a fitness improvement is achieved (without fitness improvements, only smaller trees are acceptable). With a careful drift analysis, we show that in this case we get an (expected) optimization time of $\Theta(T_{\text{init}} + n \log n)$ (see Theorem 4.1). Previously, no bound was known for MAJORITY and the bound of $O(n^2 \log n)$ for ORDER required a condition on the initialization.

Without such bloat control it is much harder to derive definite bounds. From [DNO11] we have the conditional bounds of $O(nT_{\max})$ for ORDER using either $k = 1$ or $k = 1 + \text{Pois}(1)$, where $T_{\max}$ is an upper bound on the maximal size of the best-so-far tree in the run (thus, these bounds are conditional on these maxima not being surpassed). For MAJORITY and $k = 1$ [DNO11] gives the conditional bound of $O(n^2 T_{\max} \log n)$. We focus on improving the bound for MAJORITY and obtain a bound of $O(T_{\text{init}} \log T_{\text{init}} + n \log^3 n)$ for both $k = 1$ and $k = 1 + \text{Pois}(1)$ (see Theorem 5.2). The proof of this theorem requires significant machinery for bounding the extent of bloat during the run of the optimization.

The paper is structured as follows. In Section 2 we will give a short introduction to the studied algorithm. In Section 3 the main tool for the analysis is explained, that is the analysis of drift. Here we state a selection of known theorems as well as a new one (Theorem 3.3), which gives a lower bound conditional on a multiplicative drift with a bounded step size. In Section 4 we will study the case of bloat control given $k = 1 + \text{Pois}(1)$ operations in each step. Subsequently we will study MAJORITY without bloat control in Section 5. Section 6 concludes this paper. Due to space restrictions we omit the proof of Theorem 3.3 and only sketch the proofs in Section 4 and 5. In particular, we omit the proofs of all technical lemmas.

2 PRELIMINARIES

We consider tree-based genetic programming, where a possible solution to a given problem is given by a syntax tree. The inner nodes of such a tree are labeled by function symbols from a set F_S and the leaves of the tree are labeled by terminals from a set T.

We analyze the problems ORDER and MAJORITY, whose only function is the join operator (denoted by J). The terminal set X consists of $2n$ variables, where $\overline{x}_i$ is the complement of x_i:

- $F_S := \{J\}$, J has arity 2,
- $X := \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$.

For a given syntax tree t the value of the tree is computed by parsing the tree in-order and generating the set S of *expressed* literals in this way. For ORDER a literal i is expressed if a variable x_i is present in t and there is no $\overline{x}_i$ that is visited in the in-order parse before the first occurrence of x_i. For MAJORITY a literal i is expressed if a variable x_i is present in t and the number of variables x_i is at least the number of variables $\overline{x}_i$.

In this paper we consider simple mutation-based genetic programming algorithms which use a modified version of the HIERARCHICAL VARIABLE LENGTH (HVL) operator ([O'R95], [OO94]) called HVL-Prime as discussed in [DNO11]. HVL-Prime allows to produce trees of variable length by applying three different operations: insert, delete and substitute (see Figure 1). Each application of HVL-Prime chooses one of these three operations uniformly at random, whereas k denotes the number of applications of HVL-Prime we allow for each mutation.

We associate with each tree t the complexity C, which denotes the number of nodes t contains. Given a function F, we aim to generate an instance t maximizing F.

We consider two problems. The first one is the single problem of computing a tree t which maximizes F. During an optimization run

Given a GP-tree t, mutate t by applying HVL-Prime k times. For each application, choose uniformly at random one of the following three options.

substitute	Choose a leaf uniformly at random and substitute it with a leaf in X selected uniformly at random.
insert	Choose a node $v \in X$ and a leaf $u \in t$ uniformly at random. Substitute u with a join node J, whose children are u and v, with the order of the children chosen uniformly at random.
delete	Choose a leaf $u \in t$ uniformly at random. Let v be the sibling of u. Delete u and v and substitute their parent J by v.

Figure 1: Mutation operator HVL-Prime

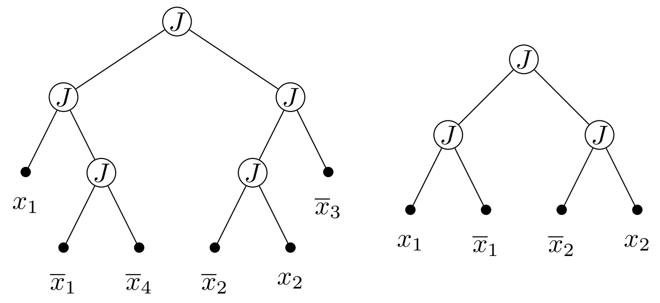

Figure 2: Two GP-trees with the same fitness. For ORDER the fitness is 1 since only the first literal occurs with a non-negated variable first. For MAJORITY the fitness is 2, since the literal 1 and 2 have one variable x_i and also one variable $\overline{x}_i$. However, the left one has complexity 11 whereas the right one has complexity 7.

Algorithm 1: (1+1) GP

```
1 Let $t$ be the initial tree;
2 **while** *optimum not reached* **do**
3     $t' \leftarrow t$;
4     Choose $k$;
5     **for** $i = 1$ **to** $k$ **do**
6         $t' \leftarrow \text{mutate}(t')$;
7     **if** $f(t') \geq f(t)$ **then** $t \leftarrow t'$;
```

we can use the complexity C to generate an order for solutions with the same fitness by preferring solutions with smaller complexity (see Figure 2). This gives us a way of breaking ties between solutions with the same fitness. Hence, the second problem consists of maximizing the multi-objective function given by F and C.

Consequently, we study the following problems:

- ORDER and MAJORITY without bloat control, which consist of maximizing the given function without studying the complexity.
- ORDER and MAJORITY with bloat control, which consist of maximizing the given function and preferring solutions with smaller complexity, if two solutions have the same function value.

To solve these problems we study the (1+1) GP proceeding as follows. It starts with a given initial tree with T_{init} leaves and tries to improve its fitness iteratively. In each iteration, the number of mutation steps k is chosen according to a fixed distribution; important options for this distribution is (i) constantly 1 and (ii) $1 + \text{Pois}(1)$, where $\text{Pois}(\lambda)$ denotes the Poisson distribution with parameter λ. The choices for k in the different iterations are i.i.d. The (1+1) GP then produces an offspring from the best-so-far individual by applying mutation k times in a row; the offspring is discarded if its fitness is worse than the best-so-far, otherwise it is kept to replace the previous best-so-far. Recall that the fitness in the case with bloat control contains the complexity as a second order term. Algorithm 1 states the (1+1) GP more formally.

3 DRIFT THEOREMS

We will use a variety of drift theorems to derive the theorems of this paper. *Drift*, in this context, describes the *expected change* of the best-so-far solution within one iteration with respect to some *potential*. In later proofs we will define potential functions on best-so-far solutions and prove bounds on the drift; these bounds then translate to expected run times with the use of the drift theorems from this section. We start with a theorem for *additive drift*.

Theorem 3.1 (Additive Drift [HY04]). *Let $(X_t)_{t \geq 0}$ be random variables describing a Markov process over a finite state space $S \subseteq \mathbb{R}$. Let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. If there exist $c > 0$ such that*

$$\mathbb{E}[X_t - X_{t+1} \mid T > t] \leq c,$$

then

$$\mathbb{E}[T \mid X_0] \geq \frac{X_0}{c}.$$

We will use the following *variable drift theorem* due to [RS12], a extension of the variable drift theorem from [Joh10, Theorem 4.6].

Theorem 3.2 (Variable Drift [RS12]). *Let $(X_t)_{t \geq 0}$ be random variables describing a Markov process over a finite state space $S \subseteq \mathbb{R}_0^+$ and let $x_{\min} := \min\{x \in S \mid x > 0\}$. Furthermore, let T be the random variable that denotes the first point in time $t \in \mathbb{N}$ for which $X_t = 0$. Suppose that there exists a monotone increasing function $h : \mathbb{R}^+ \to \mathbb{R}^+$ such that $1/h$ is integrable and*

$$\mathbb{E}[X_t - X_{t+1} \mid X_t] \geq h(X_t)$$

holds for all $t < T$. Then,

$$\mathbb{E}[T \mid X_0] \leq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{X_0} \frac{1}{h(x)} dx.$$

For our lower bounds we need the following new drift theorem, which allows for non-monotone processes (in contrast to, for example, the lower bounding multiplicative drift theorem from [Wit13]), but requires an absolute bound on the step size.

Theorem 3.3 (Multiplicative Drift, bounded step size). *Let $(X_t)_{t\geq 0}$ be random variables describing a Markov process over a finite state space $S \subseteq \mathbb{R}_+$. Let $\kappa > 0$, $s_{\min} \geq \sqrt{2}\kappa$ and let T be the random variable denoting the earliest point in time $t \geq 0$ such that $X_t \leq s_{\min}$. If there exists a positive real $\delta > 0$ such that, for all $s > s_{\min}$ and $t \geq 0$ with $\Pr[X_t = s] > 0$ it holds that*

(1) $|X_t - X_{t+1}| \leq \kappa$, and
(2) $\mathbb{E}[X_t - X_{t+1} \mid X_t = s] \leq \delta s$,

then, for all $s_0 \in S$ with $\Pr[X_0 = s_0] > 0$,

$$\mathbb{E}[T \mid X_0 = s_0] \geq \frac{1 + \ln(s_0) - \ln(s_{\min})}{2\delta + \frac{\kappa^2}{s_{\min}^2 - \kappa^2}}.$$

4 RESULTS WITH BLOAT CONTROL

In this section we show the following theorem.

Theorem 4.1. *The (1+1) GP with bloat control choosing $k = 1+Pois(1)$ on ORDER and MAJORITY takes $\Theta(T_{\text{init}} + n \log n)$ iterations in expectation.*

4.1 Lower Bound

Regarding the proof of the lower bound, let T_{init} and n be given. Let t be a GP-tree which contains T_{init} leaves labeled $\overline{x}_1$. From a simple coupon collector's argument we get a lower bound of $\Omega(n \log n)$ for the run time to insert each x_i. As an optimal tree cannot list any of the leaves in t in addition to the expected number of deletions performed by (1+1) GP being in O(1), we obtain a lower bound of T_{init} from the additive drift theorem (Theorem 3.1).

4.2 Upper Bound

The following subsection is dedicated to the proof of the upper bound. Let T_{init}, and n be given. We want to apply a theorem concerning variable drift (Theorem 3.2). In order to construct a suitable potential function we partition the leaves of a GP-tree t into three pairwise disjoint sets:

$R(t)$ Redundant leaves, i.e. leaves v, where the fitness of t is not affected by deleting v.
$C^+(t)$ Critical positive leaves, i.e. leaves v, where the fitness of t decreases by deleting v.
$C^-(t)$ Critical negative leaves, i.e. leaves v, where the fitness of t increases by deleting v.

We denote the number of expressed literals of t by $v(t)$. Additionally, we denote by $r(t)$, $c^+(t)$ and $c^-(t)$ the cardinality of $R(t)$, $C^+(t)$ and $C^-(t)$, respectively. Let $s(t)$ be the number of leaves of t (we call it the *size* of t). Due to the above defined sets we obtain

$$s(t) = r(t) + c^+(t) + c^-(t). \tag{1}$$

Before proving the upper bound on the expected optimization time we are going to prove upper bounds on the number of critical leaves.

Lemma 4.2. *Let t be a GP-tree, then for ORDER and MAJORITY we have*

$$c^+(t) \leq r(t) + v(t).$$

Lemma 4.3. *Let t be a GP-tree, then for ORDER and MAJORITY we have*

$$c^-(t) \leq 2r(t).$$

Given the best-so-far GP-tree t we are going to construct a potential function in order to apply drift analysis. Here we want to reward strongly an increase of fitness given by a decrease of the unexpressed variables. Furthermore, we want to reward a decrease of size but without punishing an increase of fitness. Thus, we associate with t the potential function

$$g(t) = 10(n - v(t)) + s(t) - v(t).$$

This potential is 0 if and only if t contains no redundant leaves and for each $i \leq n$ there is an expressed x_i. Furthermore, by Lemma 4.2 and Lemma 4.3 $s(t) - v(t)$ is also 0 since $r(t)$ is 0.

Let t be a GP-tree, the best-so-far tree in a run of the (1+1) GP. Let t' be the random variable describing the best-so-far solution in the next iteration. We are going to derive a bound on the drift, i.e. the expected change $g(t) - g(t')$, which we denote by $\Delta(t)$.

For this purpose we will distinguish between the case D_1, where the algorithm chooses to do exactly one operation in the observed mutation step, and D_2, where the algorithm chooses to do at least two operations in the observed mutation step. Since the algorithm chooses in each step at least one operation, we observe

$$\Pr[D_1] = \Pr[\text{Pois}(1) = 0] = \frac{1}{e},$$
$$\Pr[D_2] = 1 - \frac{1}{e}.$$

Furthermore, let E be the event that $v(t') = v(t)$. Note that, conditional on E, the potential cannot increase since the number of leaves can only decrease. However, conditional on $\overline{E}$, the potential can increase since every addition of a leaf is accepted as long as the fitness increases. Therefore, we are going to derive conditional bounds on the drift and apply the law of total probability in order to obtain the general drift.

Lemma 4.4. *For the expected negative drift measured by $g(t)$ conditional on D_2 holds*

$$\mathbb{E}[\Delta(t) \mid D_2] \geq -\frac{1}{e}\left(4 \cdot 10^{-7}\right).$$

In addition, if $s(t) > n/2$ holds, this bound is enhanced to

$$\mathbb{E}[\Delta(t) \mid D_2] > -\frac{7g(t)}{10en}\left(4 \cdot 10^{-6}\right)$$

Proof. First we note that the drift conditional on D_1 is always positive as with one operation the algorithm cannot increase the size by more than 1. However, for an offspring with increased number of redundant leaves we need to increase the size by more than 1.

Concerning the drift conditional on D_2 we observe

$$\mathbb{E}[\Delta(t) \mid D_2] \geq -\mathbb{E}[-\Delta(t) \mid \overline{E}] \Pr[\overline{E}],$$

since the drift can be negative only in this case. In particular, we observe a drift of at least 10 for the increase of fitness counteracted by the possible increase of the size. The latter is at most the number of operations the algorithm does in the observed step, since every operation can increase the size by at most 1.

Let $Y \sim \text{Pois}(1)+1$ be the random variable describing the number of operations in a round. Note that, for all $i \geq 1$,

$$\Pr[Y = i] = \frac{1}{e(i-1)!}.$$

By this probability we obtain for the expected negative drift conditional on $\overline{E}$

$$\begin{aligned}\mathbb{E}[-\Delta(t) \mid \overline{E}] &= \sum_{i=0}^{\infty} \mathbb{E}[-\Delta(t) \mid Y = i, \overline{E}] \Pr[Y = i \mid \overline{E}] \\ &\leq \sum_{i=0}^{\infty} (i-10) \Pr[Y = i \mid \overline{E}] \\ &\leq \sum_{i=11}^{\infty} (i-10) \Pr[Y = i \mid \overline{E}].\end{aligned}$$

Due to Bayes' theorem we derive

$$\mathbb{E}[-\Delta(t) \mid \overline{E}] \leq \sum_{i=11}^{\infty} (i-10) \Pr[\overline{E} \mid Y = i] \frac{\Pr[Y = i]}{\Pr[\overline{E}]},$$

which yields the first bound by pessimistically assuming $\Pr[\overline{E} \mid Y = i] = 1$

$$\begin{aligned}\mathbb{E}[\Delta(t) \mid D_2] &\geq -\sum_{i=11}^{\infty} (i-10) \Pr[Y = i] \\ &= -\frac{1}{e}\left(2e - 10e + \sum_{i=1}^{10} \frac{10-i}{(i-1)!}\right) \\ &\geq -\frac{1}{e}\left(4 \cdot 10^{-7}\right). \qquad (2)\end{aligned}$$

In order to obtain a better bound on the negative drift, we are going to bound the probability $\Pr[\overline{E} \mid Y = i]$ by a better bound than the previously applied bound of 1.

The event $\overline{E}$ requires a non-expressed variable in t to become expressed in t'. There are $n - v(t)$ non-expressed variables in t. These can become expressed by either adding a corresponding positive literal or deleting a corresponding negative literal. There are $2n$ literals in total and due to $n - v(t) \leq g(t)/10$ adding such a positive literal has a probability of at most

$$\frac{n - v(t)}{6n} \leq \frac{g(t)}{60n}$$

per operation. Regarding the deletion of negative literals, there are at most $s(t) - v(t)$ negative literals. Hence, due to $s(t) \leq g(t)$ and $s(t) > n/2$ the probability of deleting a negative literal is at most

$$\frac{s(t) - v(t)}{3s(t)} \leq \frac{2g(t)}{3n}$$

per operation. Let q_l be the probability that the l-th mutation leads an unexpressed variable to become expressed. We can bound the probability that i operations lead to the expression of a previously unexpressed bound by pessimistically assuming that the mutation is going to be accepted. This yields by the union bound

$$\begin{aligned}\Pr[\overline{E} \mid Y = i] &\leq \bigcup_{l=1}^{i} q_l \leq \sum_{l=1}^{i} q_i = \frac{ig(t)}{n}\left(\frac{1}{60} + \frac{2}{3}\right) \\ &< \frac{ig(t)}{n} \frac{7}{10}.\end{aligned}$$

Therefore, we obtain an expected drift conditional on D_2 of

$$\begin{aligned}\mathbb{E}[\Delta(t) \mid D_2] \geq -\mathbb{E}[-\Delta \mid \overline{E}] \Pr[\overline{E}] &> -\frac{7g(t)}{10en} \sum_{i=11}^{\infty} i(i-10)\frac{1}{(i-1)!} \\ &> -\frac{7g(t)}{10en}\left(4 \cdot 10^{-6}\right).\end{aligned}$$

□

We are now going to prove the upper bound by deriving the expected positive drift outweighing the negative drift given by Lemma 4.4. To do so, we observe that starting with a very big initial tree the algorithm will delete redundant leaves with a constant probability until most of the occurring literals are expressed. In this second stage the size of the tree is at most linear in n and the algorithm will insert literals, which do not occur in the tree at all, with a probability of at least linear in $1/n$ until all literals are expressed. In order to apply the second bound given by Lemma 4.4, we will split the second stage in two cases.

Case 1: We first consider the case $r(t) \geq v(t)$. Due to Lemma 4.2, Lemma 4.3 and Equation (1) we obtain

$$s(t) = r(t) + c^+(t) + c^-(t) \leq 4r(t) + v(t) \leq 5r(t),$$

thus the algorithm has a probability of at least $1/5$ for choosing a redundant leaf followed by choosing a deletion with probability $1/3$. Since the deletion of a redundant leaf without any additional operations does not change the fitness this contributes to the event E. Hence, we obtain for the event D_1

$$\mathbb{E}[\Delta(t) \mid D_1,\, E] \Pr[E] \geq \frac{1}{15}.$$

Additionally, the drift conditional on D_1 is always positive, which yields

$$\mathbb{E}[\Delta(t) \mid D_1] \geq \mathbb{E}[\Delta(t) \mid D_1,\, E] \Pr[E] \geq \frac{1}{15}.$$

The drift conditional on D_2 is given by Lemma 4.4. Overall, we get a constant drift in the case of $r(t) \geq v(t)$ due to the law of total expectation

$$\begin{aligned}\mathbb{E}[\Delta(t)] &\geq \mathbb{E}[\Delta(t) \mid D_1] \Pr[D_1] + \mathbb{E}[\Delta(t) \mid D_2] \Pr[D_2] \\ &\geq \frac{1}{15e} - \frac{1}{e}\left(1 - \frac{1}{e}\right)\left(4 \cdot 10^{-7}\right) \\ &\geq \frac{1}{e}\left(\frac{1}{15} - 4 \cdot 10^{-7}\right) \geq \frac{3}{50e}. \qquad (3)\end{aligned}$$

Case 2: Suppose $r(t) < v(t)$ and $s(t) \leq n/2$. In particular, we have for at least $n/2$ many $i \leq n$ that there is neither x_i nor $\overline{x}_i$ present in t. The probability to choose x_i is at least $n/4$ and the probability that the algorithm chooses an insertion is $1/3$. Since the location of the newly inserted literal is unimportant we obtain

$$\mathbb{E}[\Delta(t) \mid D_1] \Pr[D_1] \geq \frac{10}{12e}.$$

For the expected drift in the case D_2 holds we apply again the bound given by Lemma 4.4, which yields a constant drift analogously to Case 1

$$\mathbb{E}[\Delta(t)] \geq \frac{1}{e}\left(\frac{10}{12} - 4 \cdot 10^{-7}\right) > \frac{8}{10e}.$$

Case 3: Consider now the case that $r(t) < v(t)$ and $s(t) > n/2$. In particular, the tree can contain at most $5n$ leaves due to

$$s(t) \leq 4r(t) + v(t) < 5v(t) \leq 5n,$$

which enables us to bound the probability that an operation chooses a specific leaf v as

$$\frac{1}{5n} \leq \Pr[\text{choose leaf } v] \leq \frac{2}{n}.$$

Let A be the set of i, such that there is neither x_i nor $\overline{x}_i$ in t, and let B be the set of i, such that there is exactly one x_i and no $\overline{x}_i$ in t. Recall that $R(t)$ is the set of redundant leaves in t. For every i in A let A_i be the event that the algorithm adds x_i somewhere in t. For every j in $R(t)$ let $R_j(t)$ be the event, that the algorithm deletes j. Finally, let A' be the event that one of the A_i holds, and R' the event that one of the $R_j(t)$ holds.

Conditional on D_1 we observe for every event A_i a drift of 10. For each event $R_j(t)$ conditional on D_1 we observe a drift of 1 since the amount of redundant leaves decreases by exactly 1. Hence,

$$\mathbb{E}[\Delta(t) \mid A_i, D_1] = 10,$$
$$\mathbb{E}[\Delta(t) \mid R_j(t), D_1] = 1.$$

Regarding the probability for these events we observe that for A_i the algorithm chooses with probability $1/3$ to add a leaf and with probability $1/(2n)$ it chooses x_i for this. Furthermore, the position of the new leaf x_i is unimportant, hence

$$\Pr[A_i \mid D_1] \geq \frac{1}{6n}.$$

Regarding the probability of $R_j(t)$, with probability at least $1/(5n)$ the algorithm chooses the leaf j and with probability $1/3$ the algorithm deletes j. This yields

$$\Pr[R_j(t) \mid D_1] \geq \frac{1}{15n}.$$

In order to sum the events in A' and R', we need to bound the cardinality of the two sets A and $R(t)$. For this purpose we will need the above defined set B. First we note that the cardinality of B is at most $v(t)$. In addition

$$|A| + |R(t)| \geq r(t)$$

holds since $R(t)$ is the set of all redundant leaves. Furthermore, we observe that for any literal j, which is not in B or A, there has to exist at least one redundant leaf x_j or $\overline{x}_j$. Since every redundant leaf is included in $R(t)$ we obtain $|A| + |R(t)| + |B| \geq n$ and subsequently

$$|A| + |R(t)| \geq n - v(t). \tag{4}$$

Furthermore, due to (4) we deduce

$$\begin{aligned} s(t) - v(t) &\leq r(t) + c^+(t) + c^-(t) - v(t) \\ &\leq 4r(t) \leq 4(|A| + |R(t)|). \end{aligned} \tag{5}$$

This inequality (5) in conjunction with (4) yields

$$(10 + 4)(|A| + |R(t)|) \geq 10(n - v(t)) + s(t) - v(t) = g(t).$$

We obtain the expected drift conditional on the event D_1 as

$$\begin{aligned} \mathbb{E}[\Delta(t) \mid D_1] &\geq \mathbb{E}[\Delta(t) \mid (A' \vee R'), D_1] \Pr[A' \vee R' \mid D_1] \\ &= \sum_{i \in A} \mathbb{E}[\Delta(t) \mid A_i, D_1] \Pr[A_i, D_1] \\ &\quad + \sum_{j \in R(t)} \mathbb{E}[\Delta(t) \mid R_j(t), D_1] \Pr[R_j(t) \mid D_1] \\ &\geq |A| \frac{10}{6n} + |R(t)| \frac{1}{15n} \geq (|A| + |R(t)|) \frac{1}{15n} \\ &\geq \frac{g(t)}{15(10+4)n}. \end{aligned}$$

Concerning the expected drift conditional on D_2, the condition for the second bound given by Lemma 4.4 is satisfied in this case. Summarizing the events D_1 and D_2 we obtain the expected drift

$$\begin{aligned} \mathbb{E}[\Delta(t)] &\geq \mathbb{E}[\Delta(t) \mid D_1] \Pr[D_1] + \mathbb{E}[\Delta(t) \mid D_2] \Pr[D_2] \\ &\geq \frac{g(t)}{en} \left(\frac{1}{210} - \left(1 - \frac{1}{e}\right) \frac{7}{10} \cdot 4 \cdot 10^{-6} \right) \\ &> \frac{g(t)}{250en}. \end{aligned} \tag{6}$$

Summarizing the derived expected drifts (3) and (6), we observe a multiplicative drift in the case of

$$\frac{g(t)}{250en} \leq \frac{3}{50e},$$

which simplifies to $g(t) \leq 15n$. If $g(t) > 15n$, we observe a constant drift. This constant drift is at least $3/50e$ since the expected drift for Case 2 is always bigger than the one for Case 1.

We now apply the variable drift theorem (Theorem 3.2) with $h(x) = \min\{3/(50e), 1x/(250en)\}$, $X_0 = T_{\text{init}} + 10n$ and $x_{min} = 1$, which yields

$$\begin{aligned} \mathbb{E}[T \mid g(t) = 0] &\leq \frac{1}{h(1)} + \int_1^{T_{\text{init}}+10n} \frac{1}{h(x)}\, dx \\ &= 250en + 250en \int_1^{15n} \frac{1}{x}\, dx + \frac{50e}{3} \int_{15n+1}^{T_{\text{init}}+10n} 1\, dx \\ &= 250en\,(1 + \log(15n)) + \frac{50e}{3}\,(T_{\text{init}} - 5n - 1) \\ &< 250en \log(15en) + \frac{50e}{3} T_{\text{init}}. \end{aligned}$$

This establishes the theorem.

5 RESULTS MAJORITY

In this section we show the following theorems.

THEOREM 5.1. *The (1+1) GP* without bloat control *(choosing $k = 1$ or $k = 1 + Pois(1)$) on MAJORITY takes $\Omega(T_{\text{init}} + n \log n)$ iterations in expectation.*

THEOREM 5.2. *The (1+1) GP* without bloat control *(choosing $k = 1$ or $k = 1 + Pois(1)$) on (weighted) MAJORITY takes* $O(T_{\text{init}} \log T_{\text{init}} + n \log^3 n)$ *iterations in expectation.*

5.1 Lower Bound

We are going to give a rough proof for the lower bound, which will leave out in-depth analysis of standard arguments but gives non-standard details.

Let T_{init} be large and t_0 be a GP-tree which contains T_{init} leaves labeled $\overline{x}_1$ and no other leaves. From a simple coupon collector's argument we get a lower bound of $\Omega(n \log n)$ for the run time to insert each x_i.

It remains to bound the time that the algorithm needs to express the literal x_1. To derive the second bound we observe, that the algorithm does in expectation 2 operations in each iteration. Hence, the algorithm needs in expectation $\Omega(T_{\text{init}})$ iterations to express the first literal, which yields the desired result for general $n \geq 1$.

Regarding the bound for the case $n = 1$ let t be a GP-tree, let $I_1(t)$ be the number of variables x_1 in t and $I_1'(t)$ be the number of variables $\overline{x}_1$ in t. We associate with t the potential function $g(t)$ by

$$g(t) = I_1'(t) - I_1(t).$$

In order to express the literal x_1, the potential $g(t)$ has to get non-positive at one point. In particular, starting with $g(t_0) = T_{\text{init}}$, the potential has to reach a value of at most $T_{\text{init}}^{2/3}$. Let τ denote the number of iterations until the algorithm encounters for the first time a GP-tree t with $g(t) \leq T_{\text{init}}^{2/3}$. We are going to bound the expected value of τ starting with t_0, since this will yield a lower bound for the expected number of iterations until the literal x_1 is expressed.

Let A be the event that in T_{init}^2 iterations the algorithm performs at least once more than $15 \ln(T_{\text{init}})$ operations in a single round. With high probability the algorithm will not encounter the event A. This yields

$$\mathbb{E}[\tau] = \mathbb{E}[\tau \mid A] \Pr[A] + \mathbb{E}[\tau \mid \overline{A}] \Pr[\overline{A}] \geq \mathbb{E}[\tau \mid \overline{A}] \, \frac{1}{2}.$$

It remains to bound the expected value of τ under the constraint of $\overline{A}$.

Let t' be the random variable describing the best-so-far solution in the iteration after t. We are going to bound the drift, i.e. the expected change $g(t) - g(t')$, which we denote by $\Delta(t)$. Let $g(t) = k - j$, where k is the number of variables $\overline{x}_1$ and j is the number of variables x_1. We observe that the variables *introduced* by an insertion or substitution only yield a drift of 0.

Let B be the event, that the algorithm chooses at least once a variable x_1 for a substitution or deletion in this iteration. The probability of B is at least the probability for the algorithm to do exactly one operation: a deletion or substitution of a variable x_1. Let $s(t)$ be the amount of leaves of t. We deduce

$$\Pr[B] \geq \frac{1}{e} \cdot \frac{2}{3} \cdot \frac{j}{s(t)}.$$

Furthermore, the expected negative drift of $g(t)$ can be bounded by this event B, which yields $\mathbb{E}[\Delta \mid B] = -1$.

Regarding the positive drift, let C_i be the event, that in this iteration the algorithm chooses to do i operations, which are either substitutions or deletions of variables $\overline{x}_1$. We observe that each deletion of a variable $\overline{x}_1$ reduces $s(t)$ and $I_1'(t)$ by 1. Each substitution of a variable $\overline{x}_1$ reduces only $s(t)$ by 1. Therefore, we can bound the probability for a substitution by at most the probability of a deletion. Let p_i be the probability, that a Pois(1) distributed random variable is equal to i. This yields for $k < s(t)$

$$\Pr[C_i] \leq \frac{2}{3^i} \cdot p_{i-1} \cdot \frac{k!(s(t)-i)!}{s(t)!(k-i)!} \leq \frac{2}{3^i} \cdot p_{i-1} \cdot \frac{k}{2s(t)}.$$

Hence, we obtain the expected drift for $\overline{B}$

$$\mathbb{E}[\Delta(t) \mid \overline{B}] \Pr[\overline{B}] \leq \frac{k}{es(t)} \sum_{i=1}^{\infty} \frac{i}{3^i(i-1)!} = \frac{4k}{9e^{2/3}s(t)}.$$

Summarizing, we obtain by the law of total expectation

$$\mathbb{E}[\Delta(t)] \leq \frac{4k}{9e^{2/3}s(t)} - \frac{2j}{3es(t)} \leq \frac{2g(t)}{3es(t)}.$$

To bound the size $s(t)$ we observe that following a standard gambler's ruin argument within $o(T_{\text{init}}^{1.5})$ iterations the size will not shrink by a factor bigger than 1/2. Therefore, we obtain $s(t) \geq 1/2\, T_{\text{init}}$. Due to the step size bound of $15 \ln(T_{\text{init}}) < T_{\text{init}}^{2/3}$ we can apply Theorem 3.3. and derive

$$\mathbb{E}[\tau \mid \overline{A}, X_0 = T_{\text{init}}] \geq \frac{1 + \ln(T_{\text{init}}) - \ln(T_{\text{init}}^{1/2})}{\frac{2}{3eT_{\text{init}}} + \frac{(15\ln(T_{\text{init}}))^2}{T_{\text{init}}^{4/3} - (15\ln(T_{\text{init}}))^2}}.$$

Therefore, we obtain the desired result

$$\mathbb{E}[\tau] \geq \frac{3e\, T_{\text{init}} \ln(T_{\text{init}})}{8 + 12e} = \Omega(T_{\text{init}} \log T_{\text{init}}).$$

5.2 Upper Bound

We only give a very rough sketch of the proof since the proof is long and involved. The key ingredient is a bound on the bloat, i.e., on the speed with which the tree grows. More precisely, we will show that if $T_{\text{init}} \geq n \log^2 n$, then in $O(T_{\text{init}} \log T_{\text{init}})$ rounds, the size of the tree grows at most by a constant factor. Before we elaborate on the bloat, let us first sketch how this implies the upper bound. Consider any x_i that is not expressed, and let $V'(t_r, i) := \#\{\overline{x}_i\text{-literals}\} - \#\{x_i\text{-literals}\} \geq 1$. (For this outline we neglect the case that there are neither $\overline{x}_i$ nor x_i in the string.) Then the probability of deleting or relabelling a $\overline{x}_i$ is larger than deleting or relabelling a x_i, while they have the same probability to be inserted. Computing precisely, denoting t_r the GP-tree in round r, we get a drift

$$\mathbb{E}[V'(t_{r+1}, i) - V'(t_r, i) \mid V(t_r, i) = v] \leq -\frac{v}{3eT_{\max}} \tag{7}$$

for the $V'(t_r, i)$, where $T_{\max} = O(T_{\text{init}})$ is the maximal length of the string. Using a multiplicative drift theorem [DG13], after $O(T_{\text{init}} \log T_{\text{init}})$ rounds we have $V'(t_r, i) = 0$ with very high probability. By a union bound over all i, with high probability there is no i left after $O(T_{\text{init}} \log T_{\text{init}})$ rounds for which $V'(t_r, i) < 0$. This proves the theorem modulo the statement on the bloat.

In order to control the bloat, note that in expectation the offspring has the same size as the parent and the size of the tree does not change significantly by such unbiased fluctuations. However, in some situations longer offsprings are more likely to be accepted or shorter offsprings are more likely to be rejected. This results in a positive drift for the length, which we need to bound. Note that the biased drift is caused purely by the selection process. We will show that offsprings are rarely rejected and bound the drift of $|t_r|$ by (essentially) the probability that the offspring is rejected.

For an index $i \in [n]$ we say that i is *touched* by some mutation, if the mutation inserts, delete or changes a x_i or $\overline{x}_i$ variable or if it changes a variable into x_i or $\overline{x}_i$. We call a round an *i-round* if at least one of the mutations in this round touches i. First we give (without proof) some elementary properties of rounds that touch i.

Lemma 5.3. *There are constants $C, \delta > 0$ and $n_0 \in \mathbb{N}$ such that the following is true for every $n \geq n_0$, every GP-tree t, and every $\kappa \geq 2$. Let $i \in [n]$ and let k denote the number of mutations in the next round. Then:*

(1) $\Pr[k \geq \kappa] \leq e^{-\delta\kappa}$.
(2) $\Pr[k = 1 \mid i \text{ touched}] \geq \delta$.
(3) $\Pr[k \geq \kappa \mid i \text{ touched}] \leq e^{-\delta\kappa}$.
(4) $\mathbb{E}[k \mid i \text{ touched}] \leq C$.

Similar as before, for an expressed variable x_i we let $V(t_r, i) := \#\{x_i\text{-literals}\} - \#\{\overline{x}_i\text{-literals}\} \geq 0$. An important insight is that the offspring can only be rejected if there is some expressed x_i such that i is touched by at least $V(t_r, i) + 1$ mutations.[1] So we want to show that this does not happen frequently. The probability to touch i at least k times falls geometrically in k by Lemma 5.3, so in the following we will restrict to the most dominant case $V(t_r, i) = 0$.

Similar as before, we may bound the drift of $V(t_r, i)$ in rounds that touch i by

$$\mathbb{E}[V(t_{r+1}, i) - V(t_r, i) \mid V(t_r, i) = v, r \text{ is } i\text{-round}] \geq -\frac{Cvn}{T_{\text{init}}} \quad (8)$$

for a suitable constant $C > 0$. The factor n appears because we condition on r being an i-round, which happens with probability $\Theta(1/n)$.

Equation (8) tells us that the drift is negative, but relatively weak. We prove that under such circumstances, the expected return time to 0 is large. More precisely, it can be shown with martingale theory that the expected number of i-rounds to reach $V(t_r, i) = 0$ from any starting configuration is at least $\Omega(\sqrt{T_{\text{init}}/n})$. In particular, after $V(t_r, i)$ becomes positive for the first time, it needs in expectation $\Omega(\sqrt{T_{\text{init}}/n})$ i-rounds to return to 0. On the other hand, it only needs $O(1)$ i-rounds to leave 0 again. Hence, it is only in 0 in an expected $O(\sqrt{n/T_{\text{init}}})$-fraction of all i-rounds. Thus the drift of $|t_r|$ is also $O(\sqrt{n/T_{\text{init}}})$.

In particular, if $T_{\text{init}} \geq n \log^2 n$ then in $r_0 = O(T_{\text{init}} \log T_{\text{init}})$ rounds the drift increases the size of the GP-tree in expectation by at most $r_0\sqrt{n/T_{\text{init}}} = O(T_{\text{init}})$. Hence we expect the size to grow by at most a constant factor. With Markov's inequality the exact (rather technical) statement is the following.

Theorem 5.4. *Let $f = \omega(1)$ and assume $T_{\text{init}} \geq f(n) \cdot n \log^2 n$. Then there is $\varepsilon > 0$ such that, with probability at least $1 - 1/(2\sqrt{f(n)})$, within the next $r_0 := \varepsilon f(n) T_{\text{init}} \log T_{\text{init}}$ rounds the tree has never more than $T_{\max} := \frac{1}{4}\sqrt{f(n)}T_{\text{init}}$ leaves.*

Note that if $f(n)$ grows very slowly, Theorem 5.4 says roughly that after $\approx T_{\text{init}} \log T_{\text{init}}$ rounds the size of the tree is still $\approx T_{\text{init}}$. It is still technical to derive the upper bound in Theorem 5.2 from Theorem 5.4, but it is possible with the ideas sketched at the beginning of this section. This concludes the proof outline.

6 CONCLUSION

We considered a simple mutational genetic programming algorithm, the (1+1) GP. We saw that, for the two simple problems ORDER and MAJORITY, optimization is efficient in spite of the possibility of bloat: except for logarithmic factors, all run times are linear. However, bloat and the variable length representations were not easily analyzed, but required rather deep insights into the optimization process and the growth of the GP-trees.

For optimization preferring smaller GP-trees we observed a very efficient optimization behavior: whenever there is a significant number of redundant leaves, these leaves are being pruned. Whenever only few redundant leaves are present, the algorithm easily increases the fitness of the GP-tree.

For optimization without consideration of the size of the GP-trees, we were able to show that the extent of bloat is not too excessive during the optimization process, meaning that the tree is only larger by multiplicative polylogarithmic factors. While such factors are not a major obstacle for a theoretical analysis, a solution which is not even linear in the optimal solution might not be desirable from a practical point of view. For actually getting small solutions, some kind bloat control should be used.

From our analysis we witnessed an interesting option for bloat control: by changing the probabilities such that deletions are more likely than insertions we would observe in the presented drift equations a bias towards shorter solutions. Overall, this would lead to faster optimization.

[1]Some borders cases are neglected in this statement.

REFERENCES

[DG13] Benjamin Doerr and Leslie Ann Goldberg. Adaptive drift analysis. *Algorithmica*, 65(1):224–250, 2013.
[DNO11] Greg Durrett, Frank Neumann, and Una-May O'Reilly. Computational complexity analysis of simple genetic programming on two problems modeling isolated program semantics. In *Proc. of FOGA'11*, pages 69–80, 2011.
[GO98] David E. Goldberg and Una-May O'Reilly. Where does the good stuff go, and why? How contextual semantics influences program structure in simple genetic programming. In *Proc. of EuroGP'98*, pages 16–36, 1998.
[HY04] Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.
[Joh10] Daniel Johannsen. *Random Combinatorial Structures and Randomized Search Heuristics*. PhD thesis, Universität des Saarlandes, 2010. Available online at http://scidok.sulb.uni-saarland.de/volltexte/2011/3529/pdf/Dissertation_3166_Joha_Dani_2010.pdf.
[KNS11] Timo Kötzing, Frank Neumann, and Reto Spöhel. PAC learning and genetic programming. In *Proc. of GECCO'11*, pages 2091–2096, 2011.
[KSNO12] Timo Kötzing, Andrew M. Sutton, Frank Neumann, and Una-May O'Reilly. The Max problem revisited: the importance of mutation in genetic programming. In *Proc. of GECCO'12*, pages 1333–1340, 2012.
[LP02] Sean Luke and Liviu Panait. Lexicographic parsimony pressure. In *Proc. of GECCO'02*, pages 829–836, 2002.
[MM14] Andrea Mambrini and Luca Manzoni. A comparison between geometric semantic GP and cartesian GP for Boolean functions learning. In *Proc. of GECCO'14*, pages 143–144, 2014.
[MMM13] Alberto Moraglio, Andrea Mambrini, and Luca Manzoni. Runtime analysis of mutation-based geometric semantic genetic programming on Boolean functions. In *Proc. of FOGA'13*, pages 119–132, 2013.
[MO16] Andrea Mambrini and Pietro Simone Oliveto. On the analysis of simple genetic programming for evolving Boolean functions. In *Proc. of EuroGP'16*, pages 99–114, 2016.
[Neu12] Frank Neumann. Computational complexity analysis of multi-objective genetic programming. In *Proc. of GECCO'12*, pages 799–806, 2012.
[NUW13] Anh Nguyen, Tommaso Urli, and Markus Wagner. Single- and multi-objective genetic programming: new bounds for weighted ORDER and MAJORITY. In *Proc. of FOGA'13*, pages 161–172, 2013.
[OO94] Una-May O'Reilly and Franz Oppacher. Program search with a hierarchical variable length representation: Genetic programming, simulated annealing and hill climbing. In *Proc. of PPSN'94*, pages 397–406. Springer-Verlag, 1994.
[O'R95] Una-May O'Reilly. *An Analysis of Genetic Programming*. PhD thesis, Carleton University, Ottawa, Canada, 1995.
[RS12] Jonathan E. Rowe and Dirk Sudholt. The choice of the offspring population size in the (1, λ) EA. In *Proc. of GECCO'12*, pages 1349–1356, 2012.
[Wit13] Carsten Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability and Computing*, 22(2):294–318, 2013.