Run Time Bounds for Integer-Valued OneMax Functions

Jonathan Gadea Harder
jonathan.gadeaharder@hpi.de
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Timo Kötzing
timo.koetzing@hpi.de
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Xiaoyue Li
xiaoyue.li@hpi.de
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Aishwarya Radhakrishnan
aishwarya.radhakrishnan@hpi.de
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Janosch Ruff
janosch.ruff@hpi.de
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

ABSTRACT

While most theoretical run time analyses of discrete randomized search heuristics focus on finite search spaces, we consider the search space $\mathbb{Z}^n$. Understanding this search space is especially relevant for developing better algorithms for mixed-integer black box optimization (MI-BBO) problems.

We consider as a fitness functions the distance to the (unique) non-zero optimum a (based on the ℓ_1-metric) and study the (1+1) EA which mutates by applying a step-operator on each component that is determined to be varied. For changing by ± 1, we show that the expected optimization time is $\Theta(n \cdot (|a|_\infty + \log(|a|_H)))$. In particular, the time is linear in $|a|_\infty$, a measure of distance between starting point and target a. Employing a different step operator which chooses a step size from a distribution so heavy-tailed that the expectation is infinite, we get an optimization time of $O(n \cdot \log^2(|a|_1) \cdot (\log(\log(|a|_1)))^{1+\varepsilon})$.

Furthermore, we show that RLS with step size adaptation achieves an optimization time of $\Theta(n \cdot \log(|a|_1))$ and that ℓ_1-symmetric operators (as suggested by Rudolph'94) require a time at least linear in $|a|_1$.

We complement our findings with experimental results which show that asymptotically sub-optimal algorithms can be faster for smaller values of $|a|_\infty$.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**.

KEYWORDS

Evolutionary algorithms, integer optimization, run time analysis, theory.

ACM Reference Format:
Jonathan Gadea Harder, Timo Kötzing, Xiaoyue Li, Aishwarya Radhakrishnan, and Janosch Ruff. 2024. Run Time Bounds for Integer-Valued OneMax Functions. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3638529.3654091

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0494-9/24/07.
https://doi.org/10.1145/3638529.3654091

1 INTRODUCTION

Optimization problems are formalized as finding the optimal element x from a fixed search space $\mathcal{X}$ given some quality measure $f : \mathcal{X} \to \mathbb{R}$. In the theory of evolutionary search heuristics, the most commonly studied discrete search space is $\mathcal{X} = \{0,1\}^n$, the set of bit strings of fixed length. Other search spaces have been considered, such as permutations [3, 6], or multi-valued decision variables [5, 14] ($\mathcal{X} = \{0,\dots,r-1\}^n$). Note that all these search spaces are finite.

In this work we are interested in the infinite (but still discrete) search space $\mathbb{Z}^n$. This models a set of n decision variables with an infinite (totally and discretely ordered) domain. While many search problems can be usefully addressed by translating them into optimization problems using $\mathcal{X} = \{0,1\}^n$ or another of the before-mentioned search spaces, this is impossible in principle for an infinite search space. Furthermore, understanding them in their more natural formulation can lead to more efficient optimization algorithms.

In order to analyze heuristic search in this domain, we generalize fitness functions as well as heuristic search algorithms accordingly. Note that a generalization of the $\{0,1\}^n$ search space to $\{0,\dots,r-1\}^n$ was done in [5], and we follow a similar path in the generalization, but now with the added difficulty of an infinite search space.

As a first analysis, we consider the simple setting where, for a given $a \in \mathbb{Z}^n$, we have the fitness function

$$f_a : \mathbb{Z}^n \to \mathbb{Z}_{\geq 0}, x \mapsto \sum_{i=1}^{n} |x_i - a_i|.$$

Minimizing this function generalizes the well-known OneMax function (class), defined on the search space $\{0,1\}^n$, to the more general space of $\mathbb{Z}^n$.

As algorithms, we consider the (1+1) EA and RLS (Random Local Search), both suitably adjusted to deal with the search space $\mathbb{Z}^n$ as follows (see Section 2 for a detailed description of both algorithms). First, for finite search spaces, it is common to start the search with a uniformly random search point. For the infinite search space $\mathbb{Z}^n$ we make the decision to start deterministically with the all-0 string.

Since for our fitness functions only the position-wise differences of the starting point to the optimum matter, this choice does not restrict the meaningfulness of our results.

Second, as a variation operator, we consider to change either exactly one position (RLS) or each position independently with probability $1/n$ ((1+1) EA), just as in the common definitions of these algorithms. However, while changing a *bit* from $\{0,1\}$ leaves only one possible choice for the new value, changing a variable from $\mathbb{Z}$ leaves infinitely many choices for the new value. We consider two possible *step operators*, defining how to change a value from $\mathbb{Z}$. The first step operator is the ± 1 *operator*; it either increases or decreases the value by 1, decided uniformly at random. The second step operator is the *heavy-tailed operator*; it makes a uniform decision to either increase or decrease, but instead of deterministically changing by 1, it changes by a random number. In particular, we consider a distribution of the numbers that is unbounded and is so heavy-tailed that it does not have a finite expectation.

Note that [5] considered a *Harmonic operator* as a step operator for variables on $\{0,\dots,r-1\}$, which gives a step of size i a weight proportional to $1/i$. This cannot be directly extended to an operator on $\mathbb{Z}$, since the sequence $(1/i)_i$ is not summable. Instead, we take inspiration from [4], where very slowly decreasing yet summable sequences were considered, to define our heavy-tailed operator. In effect, a step size of 2^{i-2} has a probability proportional to $1/(i\log(i)^{1+\varepsilon})$. The goal of this very heavy tail of the distribution is to have as high as possible a chance to gain a constant fraction of the distance to the optimum, independent of the distance to the optimum. See Section 2 for details on the distribution.

Recently, heavy-tailed distributions were used to speed up optimization in various settings. In [8], the authors proposed to apply a heavy-tailed mutation operator for the first time. In particular, the number of variables to change was chosen from a heavy-tailed distribution (which for us still follows the traditional binomial distribution). This more explorative mutation operator was then shown to optimize so-called jump-functions faster. In contrast to our work, the heavy-tailed distributions in [8] have finite expectation. Since the publication of [8], further analyses have shown the use of such heavy-tailed distributions, for example for crossover and the $(1+(\lambda,\lambda))$-GA on OneMax [1] and jump-functions [2].

In our work, we consider the expected time of the given algorithms to find the optimum of a fitness function f_a. This time naturally depends on a (as well as n), specifically on its total weight $|a|_1 = \sum_{i=1}^n |a_i|$, its maximal weight $|a|_\infty = \max\{|a_i| \mid i \le n\}$ or its Hamming distance to the all-0 string $|a|_H = |\{i \mid a_i \neq 0, i \le n\}|$. Throughout our analyses we consider a to be non-zero.

In Section 3, we formally analyze the (1+1) EA with the ± 1 operator. Theorems 5 and 6 show that the expected optimization time is $\Theta(n\cdot(|a|_\infty + \log(|a|_H)))$ on any given fitness function f_a. While the linear dependence on the dimension n constitutes a good performance, the linear dependence on $|a|_\infty$ is rather slow. For comparison, note that there are $|a|_\infty^{\Theta(n)}$ target bit strings of roughly that size. Since the (1+1) EA gains about one bit of information with a comparison of two fitness values, a direct information-theoretic lower bound for finding the optimum is at $\Omega(n\cdot\log(|a|_\infty))$.

In Section 4 we turn to the (1+1) EA with the heavy-tailed operator. In Theorem 7 we show that the expected optimization time is $O(n\cdot\log^2(|a|_1)\cdot(\log(\log(|a|_1)))^{1+\varepsilon})$ for a given f_a. This is already much closer to the information-theoretic lower bound mentioned above.

We consider a version of RLS which adapts the step size during the search. This strategy was proven to be very efficient in [5] for the search space $\{0,\dots,r-1\}$ and we show that also here the algorithm achieves an expected optimization time of $\Theta(n\cdot\log(|a|_1))$. Note that, for all three operators, we derive central parts of our proofs by carefully adjusting analogous proofs from [5].

The only previous mention of the search space $\mathbb{Z}^n$ was in [14] where the authors suggest to use so-called ℓ_1-symmetric operators, which distribute the magnitude of change over all positions uniformly at random. We consider such operators in Section 6 and characterize their sampling behavior in Theorem 12. In the remainder of that section we show, for specific instances of algorithmic choices, that the optimization time of using any such operators is linear in $|a|_\infty$. Thus, we believe that the property of ℓ_1-symmetry is not desirable.

With our analyses of the search space $\mathbb{Z}^n$, we also aim at bridging the gap towards analyses of heuristic search on $\mathbb{R}^n$ (continuous optimization): If one is interested in approximating the optimum up to a distance of ε, one can discretize the search space accordingly, arriving at the search space $\mathbb{Z}^n$. In optimization problems on $\mathbb{R}^n$, one is frequently interested only in approximating the optimum, since finding it is typically impossible in principle. In Corollary **??**, we show a result about the (1+1) EA with the heavy-tailed operator approximating the optimum, showing that finding an approximation ratio of α scales with respect to α as $O(\log(1/\alpha))$.

In Section 7 we experimentally compare four algorithms with different mutation operators and show the scaling behavior of the run times for different settings.

The remainder of this paper is structured as follows. In Section 2 we introduce algorithms and notation. In Sections 3 and 4 we give our theoretical analyses of the (1+1) EA. Section 5 addresses the self-adjusting RLS and Section 6 gives the details of our results for distance symmetric operators. In Section 7 we present our experiments before we conclude in Section 8. Note that some proofs have been moved to the supplementary material due to space constraints.

2 PRELIMINARIES

In this section we define the (1+1) EA and random local search algorithms, along with the different step operators. At the end of this section we also state different drift theorems we use for our analysis.

For any $a \in \mathbb{Z}^n$, we let

$$|a|_1 = \sum_{i=1}^n |a_i|;$$
$$|a|_\infty = \max\{|a_i| \mid i \le n\};$$
$$|a|_H = |\{i \mid a_i \neq 0, i \le n\}|.$$

Also for $a \in \mathbb{Z}^n \setminus \{0^n\}$, we define

$$f_a : \mathbb{Z}^n \to \mathbb{R}, x \mapsto |a-x|_1.$$

Our class of target fitness functions is $\mathcal{F} = \{f_a \mid a \in \mathbb{Z}^n \setminus \{0^n\}\}$ to be minimized by evaluating the fitness function at any point as chosen by the algorithm.

We consider the following step operators which decide on the update of a mutation in a given component.

Definition 1 (±1-operator). *The* ±1-operator *takes an integer x as input and makes the following changes: With probability $1/2$ return $x+1$ and otherwise return $x-1$.*

For the next operator we need a definition. For $\varepsilon > 0$, we denote $c_\varepsilon = \sum_{i=2}^{\infty} \frac{1}{i \cdot (\log i)^{1+\varepsilon}}$ (note that this sum is finite [4]).

Definition 2 (Heavy-tailed-operator). *For a given $\varepsilon > 0$, the* heavy-tailed operator *takes an integer x as input and makes the following changes: First sample a step size of 2^{I-2} using a random variable I which can take a value of any natural number $i \geq 2$ with $P(I = i) = \frac{1}{c_\varepsilon \cdot i \cdot (\log i)^{1+\varepsilon}}$. Then, with probability $\frac{1}{2}$ return $x + 2^{I-2}$ and otherwise return $x - 2^{I-2}$.*

We then consider the RLS and the (1+1) EA as given by Algorithm 1 and 2. Both start from the initial search point being the all-0 string. They then proceed in rounds, each of which consists of a *mutation* and a *selection* step. Throughout the whole optimization process, the algorithms maintain a single individual, which is always the most recently sampled best-so-far solution. The two algorithms differ only in the *mutation operator*. While *RLS* makes a step in exactly one position (chosen uniformly at random), the (1+1) EA makes, in each position, a step with probability $1/n$. We specify the termination criterion as the point in time when the search point has a fitness of 0.

Algorithm 1: The (1+1) EA minimizing a function $f : \mathbb{Z}^n \mapsto \mathbb{R}$ with a given step operator $\text{step} : \mathbb{Z} \to \mathbb{Z}$

```
Initialization: x ← 0^n;
Optimization: while f(x) ≠ 0 do
    for i from 1 to n do
        With probability 1/n set y_i = step(x_i) and set
        y_i = x_i otherwise;
    if f(y) ≤ f(x) then
        x ← y;
```

$RLS_{\alpha,\beta}$ maintains a search point $x \in \mathbb{Z}^n$ as well as a real-valued *velocity vector* $v \in [1, \infty]^n$; we use real values for the velocity to circumvent rounding problems. The initial search point is the all-0s string and the initial velocity is the all-1s string. In one iteration of the algorithm a position $i \in [n]$ is chosen uniformly at random. The entry x_i is replaced by $x_i - \lfloor v_i \rfloor$ with probability $1/2$ and by $x_i + \lfloor v_i \rfloor$ otherwise. The entries in positions $j \neq i$ are not subject to mutation. The resulting string y replaces x if its fitness is at least as good as the one of x, i.e. if $f(y) \leq f(x)$ holds. If the offspring y is strictly better than its parent x, i.e. if $f(y) < f(x)$, we increase the velocity v_i in the i-th component by multiplying it with the constant $\alpha > 1$; we decrease v_i to βv_i ($\beta < 1$) otherwise. The algorithm proceeds this way until we decide to stop it. To further lighten the notation, we say that the algorithm "*moves in the right direction*" or "*towards the target value*" if the distance to the target is actually decreased. Analogously, we speak otherwise of a step "*away from the target*" or "*in the wrong direction*".

Algorithm 2: $RLS_{\alpha,\beta}$ with self-adjusting step sizes minimizing a function $f : \mathbb{Z}^n \mapsto \mathbb{R}$

```
Initialization: x ← 0^n, v ← 1^n;
Optimization: while f(x) ≠ 0 do
    y ← x;
    Choose i ∈ [n] uniformly at random;
    With probability 1/2 set y_i = x_i − ⌊v_i⌋ and set
      y_i = x_i + ⌊v_i⌋ otherwise;
    if f(y) < f(x) then
        v_i ← α v_i;
    else
        v_i ← max{1, β v_i};
    if f(y) ≤ f(x) then
        x ← y;
```

A central tool in many of our proofs is drift analysis. This allows us to derive bounds on hitting by using bounds on the expected progress a process makes towards the target. We briefly collect here the tools we use.

Additive drift is the situation that the progress is bounded by any (constant) value. This quite common situation in run time analysis was the first framed into a drift theorem, namely the following one in [10].

Theorem 3 (Additive Drift Theorem [10]). *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers and let $(X^t)_{t \in \mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. 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 constant $\delta_1 > 0$ such that*

$$E[X^t - X^{t+1} \mid T > t] \geq \delta_1$$

holds. Then

$$E[T \mid X^0] \leq \frac{X^0}{\delta_1}.$$

If there exists a constant $\delta_2 > 0$ such that

$$E[X^t - X^{t+1} \mid T > t] \leq \delta_2$$

holds. Then

$$E[T \mid X^0] \geq \frac{X^0}{\delta_2}.$$

Multiplicative drift addresses the situation where progress is proportional to the distance from the target. For this situation, which is quite common in run time analysis we can use the following drift theorem [7].

Theorem 4 (Multiplicative Drift Theorem [11]). *Let $(X_t)_{t \in \mathbb{N}}$ be random variables over $\mathbb{R}$, $x_{\min} > 0$, and let $T = \min\{t \mid X_t < x_{\min}\}$. Furthermore, suppose that*

(a) $X_0 \geq x_{\min}$ and, for all $t \leq T$, it holds that $X_t \geq 0$, and that
(b) there is some value $\delta > 0$ such that, for all $t < T$, it holds that $X_t - E[X_{t+1} \mid X_0, \ldots, X_t] \geq \delta X_t$.

Then

$$E[T \mid X_0] \leq \frac{1 + \ln\left(\frac{X_0}{x_{\min}}\right)}{\delta}.$$

3 UNIT MUTATION STRENGTH

In this section, we regard the mutation operator that applies only ±1 changes to each component. We give a tight bound on the expected run time of the (1+1) EA with the ±1 operator. We start with a theorem which gives an upper bound on the expected optimization time of the $(1+1)EA$ optimizing any $f_a \in \mathscr{F}$ with the ±1 operator. Our proof idea is similar to the proof idea in [5, Theorem 12]. More details can be found in the supplementary material.

Theorem 5. *Let $f_a \in \mathscr{F}$. Then the expected optimization time of the (1+1) EA with ±1 operator starting with all 0 integer string on f_a is $O(n \cdot (|a|_\infty + \log(|a|_H)))$.*

We show in the following theorem that this upper bound on the (1+1) EA is sharp by proving the same asymptotic lower bound.

Theorem 6. *Let $f_a \in \mathscr{F}$. Then the expected optimization time of the (1+1) EA with ±1 operator starting with all 0 integer string on f_a is $\Omega(n \cdot (|a|_\infty + \log(|a|_H)))$.*

Proof. The lower bound $|a|_\infty \cdot n$ follows from looking at the position with a distance of $|a|_\infty$ to the target. The drift in the right direction is at most $\frac{1}{n}$, which is the probability of mutating the position. Therefore by the additive drift Theorem [3], we have a run time of $\Omega(n \cdot |a|_\infty)$. For the second part we prove that $P(T \geq (n-1)\log(|a|_H))$ is at least $\frac{1}{2}$. The probability that a particular index does not get modified in any of the t iterations is $\left(1 - \frac{1}{n}\right)^t$. The previous statement implies that the probability that it does get modified at least once is $1 - \left(1 - \frac{1}{n}\right)^t$. Therefore the probability that $|a|_H$ indices gets modified at least once in t iterations is $\left(1 - \left(1 - \frac{1}{n}\right)^t\right)^{|a|_H}$. This in turn implies that the probability that at least one of the $|a|_H$ indices does not get modified in t iterations is $1 - \left(1 - \left(1 - \frac{1}{n}\right)^t\right)^{|a|_H}$. If $t = (n-1)\ln(|a|_H)$, then

$$\begin{aligned}
1 - \left(1 - \left(1 - \frac{1}{n}\right)^t\right)^{|a|_H} &= 1 - \left(1 - \left(1 - \frac{1}{n}\right)^{(n-1)\ln|a|_H}\right)^{|a|_H} \\
&\geq 1 - \left(1 - \left(\frac{1}{e}\right)^{\ln|a|_H}\right)^{|a|_H} \\
&= 1 - \left(1 - \frac{1}{|a|_H}\right)^{|a|_H} \geq 1 - e^{-1} \geq \frac{1}{2}.
\end{aligned}$$

Now we have expected time

$$\begin{aligned}
E[T] &= \sum_{t=1}^{\infty} t \cdot P(T = t) \\
&= \sum_{t=1}^{\infty} P(T \geq t) \\
&\geq (n-1)\ln|a|_H \cdot P(T \geq (n-1)\ln|a|_H) \\
&\geq (n-1)\ln|a|_H \cdot \frac{1}{2} = \Omega(n \ln|a|_H).
\end{aligned}$$

□

4 HEAVY-TAILED MUTATION STRENGTH

In this section we discuss the behavior of the (1+1) EA with the heavy-tailed mutation operator on $f_a \in \mathscr{F}$. First, in the following theorem, we give an *upper* bound on the expected optimization time.

Theorem 7. *Let $f_a \in \mathscr{F}$. Then the expected optimization time of the (1+1) EA, starting with the all-0s integer string, with the* heavy-tailed operator *(see Definition 2) with parameter $\varepsilon > 0$ on f_a is $O(n \cdot \log^2|a|_1 \cdot (\log(\log|a|_1))^{1+\varepsilon})$.*

Proof. Our proof idea is similar to the proof idea in [5, Theorem 15]. We use multiplicative drift analysis in this proof.

Let x and x' be the integer strings at some arbitrary iteration t and $t+1$ when the (1+1) EA with heavy-tailed operator is optimizing f_a. Let the potential value X at time t be $f_a(x)$ and X' be the potential at time $t+1$. Then the initial potential value is $|a|_1$, since we start with all 0 integer string. Let $T = \min\{t \geq 0 \mid X = 0\}$. For any $t \geq 0$ and $i \in \{1, \ldots, n\}$, let $d_i = |a_i - x_i|$ and $d_i' = |a_i - x_i'|$.

For any $i \in \{1, \ldots, n\}$ and $j \in \{0, 1, \ldots, \lfloor \log d_i \rfloor\}$, let $A_{i,j}$ be the event that the mutation operator only modifies index i such that $|a_i - x_i| - |a_i - x_i'| = 2^j$ and does not cause any other changes. We define $j^* := j + 2$, then $P(A_{i,j}) \geq \frac{1}{2ec_\varepsilon n j^* (\log j^*)^{1+\varepsilon}}$. We get the previous bound on the probability because the probability to make a change of size 2^j (when $d_j > 2^j$) to a particular index in the right direction is $\frac{1}{2c_\varepsilon n j^* (\log j^*)^{1+\varepsilon}}$ and the probability to make exactly this change and no other changes to any other indices is at least $1/e$. Also note that while calculating the probability $P(A_{i,j})$, we did not consider the case that the mutation operator can overshoot, since this will only increase the probability.

$$\begin{aligned}
E[X - X'] &\geq \sum_{i=1}^{n} \sum_{j=0}^{\lfloor \log d_i \rfloor} E[X - X' \mid A_{i,j}] \cdot P(A_{i,j}) \\
&= \sum_{i=1}^{n} \sum_{j=0}^{\lfloor \log d_i \rfloor} 2^j \cdot P(A_{i,j}) \\
&\geq \sum_{i=1}^{n} \sum_{j=0}^{\lfloor \log d_i \rfloor} \frac{2^j}{2ec_\varepsilon n j^* (\log j^*)^{1+\varepsilon}} \\
&\geq \frac{1}{4ec_\varepsilon n} \sum_{i=1}^{n} \frac{d_i}{(\lfloor \log d_i \rfloor + 2)\,(\log(\lfloor \log d_i \rfloor + 2))^{1+\varepsilon}} \\
&\geq \frac{\sum_{i=1}^{n} d_i}{4ec_\varepsilon n \cdot (\log|a|_1 + 2) \cdot (\log(\log|a|_1 + 2))^{1+\varepsilon}} \\
&= \frac{X}{4ec_\varepsilon n \cdot (\log|a|_1 + 2) \cdot (\log(\log|a|_1 + 2))^{1+\varepsilon}}.
\end{aligned}$$

The third to fourth inequality is because, for any real number $y > 0$, $2^{\log(y)} \geq y/2$. Since we have an initial potential value $|a|_1$ and a multiplicative drift value of $\frac{1}{4ec_\varepsilon n \cdot (\log|a|_1 + 2) \cdot (\log(\log|a|_1 + 2))^{1+\varepsilon}}$, by the multiplicative drift Theorem (4),

$$E[T] \leq 4ec_\varepsilon n \cdot (\log|a|_1 + 2) \cdot (\log(\log|a|_1 + 2))^{1+\varepsilon} \cdot (1 + \log|a|_1)$$

$$= O(n \cdot \log^2 |a|_1 \cdot (\log(\log |a|_1))^{1+\varepsilon}).$$

Thus we get the upper bound as claimed. □

5 SELF-ADJUSTING MUTATION RATES

In this section, we analyze self-adjusting mutation rates for the *RLS* algorithm and show how these can outperform the (1+1) EA with the static operators analyzed in the previous sections. The mutation strength for $RLS_{\alpha,\beta}$ is adjusted using the constants $1 < \alpha \leq 2$ and $1/2 < \beta < 1$ (see Algorithm 2 for further details on the algorithm). In Theorem 10, we give a tight bound on the expected run time of $RLS_{\alpha,\beta}$ for suitable α and β. We start by giving a lower bound on the expected run time in Lemma 8.

Lemma 8 (RLS lower bound). *Let $f_a \in \mathcal{F}$. For constants α, β the expected optimization time of $RLS_{\alpha,\beta}$ starting with all 0 integer string on f_a is $\Omega(n \cdot \log(|a|_1))$.*

Proof. A bound of $\Omega(n \log |a|_H)$ easily follows from a coupon collector argument: Since we need to change $|a|_H$ many entries and each one has a probability of $\frac{1}{n}$ of being changed in each iteration.

A bound of $\Omega(n \log |a|_\infty)$ follows from analyzing the entry j with the highest distance to the target. First observe that since the velocity at most doubles each time that an entry is selected and we start at 0, we need at least $\log(a_\infty) - 1$ changes for this entry. Let X_t be the random variable counting the number of changes on j. Let Y_t be another random variable with $Y_t := \log(a_\infty) - 1 - X_t$. Since $Y_0 = \log(|a|_\infty) - 1$ and we have an additive drift of at most $\frac{1}{n}$, the expected run time is of order $\Omega(n \cdot \log(|a|_\infty))$. We obtain this drift because the probability of changing the entry j is $\frac{1}{n}$ and we can only change it or not change it.

The lower bound of $\Omega(n \log |a|_1)$ is obtained by adding both lower bounds (this is asymptotically the same as taking the max) to get

$$n \log |a|_\infty + n \log |a|_H = n \log(|a|_\infty |a|_H) \geq n \log(|a|_1).$$

□

The proof of the upper bound in the following lemma is essentially the same as in [5, Theorem 17], only omitting parts that are not necessary for our setting. More details can be found in the supplementary material.

Lemma 9 (RLS upper bound). *Let $f_a \in \mathcal{F}$. For constants α, β satisfying $1 < \alpha \leq 2, 1/2 < \beta \leq 0.9, 2\alpha\beta - \beta - \alpha > 0, \alpha + \beta > 2$ and $\alpha^2\beta > 1$ the expected optimization time of $RLS_{\alpha,\beta}$ starting with all 0 integer string on f_a is $O(n \cdot \log(|a|_1))$.*

Combining both results we get a sharp run time result in the following theorem.

Theorem 10. *Let $f_a \in \mathcal{F}$. For constants α, β satisfying $1 < \alpha \leq 2, 1/2 < \beta \leq 0.9, 2\alpha\beta - \beta - \alpha > 0, \alpha + \beta > 2$ and $\alpha^2\beta > 1$ the expected optimization time of $RLS_{\alpha,\beta}$ starting with all 0 integer string on f_a is $\Theta(n \cdot \log(|a|_1))$.*

Example parameter values which satisfies the conditions in the theorem statement are $\alpha = 1.7$ and $\beta = 0.9$.

6 DISTANCE-SYMMETRIC OPERATOR

Symmetric random variables are of great significance in various fields of probability theory and computational optimization due to their remarkable properties and applications. For this paper, we are interested in exploring ℓ_1-symmetric random variables [9]. This concept was used in the context of evolutionary algorithms by Rudolph [14] who gave a specific ℓ_1-symmetric operator based on differences of geometric random variables.

Definition 11. *A discrete multivariate distribution X over $\mathbb{Z}^n$ is called ℓ_1-symmetric if for the probability to generate a specific point there exists a function $g : \mathbb{N} \mapsto [0, 1]$ such that*

$$\forall x \in \mathbb{Z}^n : Pr(X = x) = g(|x|_1).$$

Rudolph proposes an ℓ_1-symmetric operator and suggests its use for integer problems. The paper provides insights into how to control the mean deviation during the search process and presents empirical evaluations on five nonlinear integer problems, showcasing the effectiveness of the ℓ_1-symmetric operator. Here we will prove some general characteristics common to *all* ℓ_1-symmetric operators and also give a lower bound on the run time of (1+1) EA with any ℓ_1-symmetric operator.

Theorem 12. *Let M be a multivariate random variable over $\mathbb{Z}^n$. Then M is ℓ_1-symmetric if and only if there exists a distribution D over natural numbers such that the multivariate random variable $\overline{M}^D$ describing the result of the following procedure is identically distributed to M:*

1. *Draw $k \sim D$.*
2. *Uniformly place k balls into n bins, and let a_i be the number of balls in bin i after that.*
3. *Draw a sign $x_i \in \{-1, 1\}$ uniformly for each $i \leq n$.*
4. *Return the result $(x_1 a_1, \ldots, x_n a_n)$.*

Proof. To prove the theorem, we first establish two weaker lower bounds and then combine these to achieve the stated lower bound.

($\Rightarrow$) First, we prove the left to right implication. Let g be given as M is ℓ_1-symmetric and define, for all $k \in \mathbb{N}$, $C_k := \binom{n+k-1}{n-1}$ as the number of ways to partition k identical balls into n distinguishable bins. Let D be such that

$$\forall k : Pr(D = k) := Pr(|M|_1 = k) = C_k \cdot g(k)$$

Note that this is a valid probability distribution since $g(k)$ is, by definition, the uniform probability for M to generate any specific point with ℓ_1-norm of k and there are C_k points satisfying this criterion. This uniformity arises because each distribution of the balls over the bins defines a unique vector M_D, ensuring that each point is equally likely. We further see for all m

$$\begin{aligned}
Pr(\overline{M}^D = m) &= Pr(|\overline{M}^D|_1 = |m|_1 \wedge \overline{M}^D = m) \\
&= Pr(|\overline{M}^D|_1 = |m|_1) \cdot \frac{Pr(|\overline{M}^D|_1 = |m|_1 \wedge \overline{M}^D = m)}{Pr(|\overline{M}^D|_1 = |m|_1)} \\
&= Pr(|\overline{M}^D|_1 = |m|_1) \cdot Pr(\overline{M}^D = m \mid |\overline{M}^D|_1 = |m|_1) \\
&= C_{|m|_1} \cdot g(|m|_1) \cdot \frac{1}{C_{|m|_1}} = g(|m|_1) = Pr(M = m).
\end{aligned}$$

($\Leftarrow$) Now we prove the right to left implication. Let D be given such that M and $\overline{M}^D$ are identically distributed. We fix, for all $k \in \mathbb{N}$, an m_k with $|m_k|_1 = k$ and define

$$g(k) := Pr(\overline{M}^D = m_k) = Pr(M = m_k).$$

We will now show that, for all m with $|m|_1 = k$, $Pr(M = m) = g(k)$. The probability $Pr(M = m)$ can be decomposed as before into

$$Pr(|\overline{M}^D|_1 = k) Pr(\overline{M}^D = m \mid |\overline{M}^D|_1 = k).$$

By definition,

$$Pr(\overline{M}^D = m \mid |\overline{M}^D|_1 = k)$$

is the uniform distribution on possible targets with ℓ_1-norm k, so $Pr(\overline{M}^D = m \mid |\overline{M}^D|_1 = k) = Pr(\overline{M}^D = m_k \mid |\overline{M}^D|_1 = k)$ which concludes the proof. □

6.1 Unbiased Complexity

We now prove that, for certain worst case instances, the (1+1) EA employing *any* ℓ_1-symmetric operator in every iteration (potentially a different operator each iteration) will require a time linear in $|a|_1$. Note that this is exponentially worse than the RLS with self-adjusting mutation rates. This result has the very general flavor of unbiased black-box complexity results [12], since it makes no assumptions about the search other than that the operators are unbiased.

Theorem 13. *Let $r \in \mathbb{Z}$ and $a = (r, 0, \ldots, 0) \in \mathbb{Z}^n$. Then there is a constant c such that the (1+1) EA which, in each iteration, chooses any ℓ_1-symmetric operator to generate offspring, requires at least $\Omega(\min(r/n^3, 2^{cn}))$ iterations to optimize f_a.*

6.2 Progress-Maximizing Strategies

For the rest of the section we focus on algorithms that adjust the mutation strength with a certain success criterion in mind, such as the 1/5th-rule. We show lower bounds for such criteria.

As we discussed in Theorem 12, the choice of operator of the algorithm is equivalent to choosing a mutation strength k. We first consider choosing k such that the probability for the offspring x' to be accepted is maximal. Under this assumption, we choose a specific target function as in Theorem 13, such that the mutation strength is equal to 1 for the first step of the algorithm.

Lemma 14. *Let $n \in \mathbb{N}_{\geq 16}$, $r \in \mathbb{Z}$ and $a \in \mathbb{Z}^n$ such that $a_1 = r$ and for any $i \in \{2, \ldots, n\}$, $a_i = 0$. Let x' be the integer string representing the offspring at iteration $t = 1$ of the (1+1) EA optimizing f_a with an ℓ_1-symmetric operator. Moreover, let $k \in \mathbb{Z}^+$ be the mutation strength. Then, the probability that x' is accepted gets maximized by $k = 1$.*

Proof. Here an outline of the proof. Let p_k denote the probability of the event that the offspring x' is accepted conditioned on the event that the mutation strength k for the (1+1) EA optimizing f_a with ℓ_1-symmetric operator. In other words, we have to show that $p_1 > p_k$ for $k \geq 2$. Based on this notation we outline the proof of the statement in two steps. First, we show p_1 to be larger than smallish values for the mutation strength, namely $p_1 > p_2, p_3, p_4$. To that end, we derive a general formula for the probability p_k, i.e., the probability that the offspring is accepted for mutation strength k. Then, in a second step, we obtain a general upper bound for p_k and show that for $k \geq 5$ this upper bound to be smaller than p_1 under the mild assumption that n is large enough. Specifically, the statement holds for $n \geq 16$ as given as a hypothesis in our lemma.

Recall that the initial string before any offspring gets accepted is given by the all-0 integer string 0^n while the target function only differs in the first coordinate with $a_1 = r$. By Theorem 12, the ℓ_1-symmetric operator can be split into four steps. Let us fix the mutation k in order to compute p_k. Thus, the first step of the algorithm in Theorem 12 that picks the mutation strength is omitted and we assume the mutation strength to be k. We observe that, in the second step of the algorithm, adding a ball to the first bin potentially improves the fitness and putting a ball to any other bin makes it worse. Hence, a potential improvement towards the target function a is possible if and only if there are strictly more than half of the available balls to be be placed in the first bin (the number of balls are equal to k representing the mutation strength). Hence, if there are k balls to choose from in total, then there must be at least $m \geq \lfloor k/2 \rfloor + 1$ balls chosen to be placed in the first bin. Since the bin for a single ball is chosen uniformly at random and the number of bins is given by n, the probability of choosing the first bin for the j-th ball is given by $q = 1/n$. On the flip side, the probability for ball j to be not placed into the first bin is given by the complement rule with $p = 1 - 1/n$. Let Y be the random variable that counts the number of balls that are not placed in the first bin. Note that $Y \sim B(k, p)$ follows a binomial distribution. Thus, $Pr(Y = i) = \binom{k}{i}(1 - 1/n)^i (1/n)^{k-i}$. Recall that, given mutation strength k, at most $k - m \leq \lceil k/2 \rceil - 1$ balls can be placed in a bin other than the first one in order to enable a possible improvement via the offspring. Finally, in conjunction with with $Pr(Y = i)$, this gives

$$p_k = \frac{1}{2} \sum_{i=0}^{\lceil k/2 \rceil - 1} Pr(Y = i) = \frac{1}{2} \sum_{i=0}^{\lceil k/2 \rceil - 1} \binom{k}{i} (1 - 1/n)^i (1/n)^{k-i} \quad (1)$$

$$\leq \frac{1}{2} \sum_{i=0}^{\lceil k/2 \rceil - 1} \binom{k}{i} (1/n)^{k-i}, \quad (2)$$

whereby the $1/2$ in front of the sum stems from the fact that the sign $x_i \in \{-1, 1\}$ in the third step of the algorithm is chosen again uniformly at random. Thus, the probability of choosing the sign such that the offspring improves the parent string in the first position is given by $1/2$. In order to complete the first step of our proof it is left to show that $p_1 > p_2, p_3, p_4$. Recall that $n \geq 16$ by the hypothesis of our statement. Then, by plugging $k = 2, 3, 4$ into Equation 2 and applying Equation 1 to compute p_1, we indeed obtain

$$p_1 = \frac{1}{2n} > \frac{1}{2n^2} = p_2,$$
$$p_1 = \frac{1}{2n} > \frac{1}{2n^3} + \frac{3}{2n^2} \geq p_3,$$
$$p_1 = \frac{1}{2n} > \frac{1}{2n^4} + \frac{2}{n^3} \geq p_4,$$

assuming that $n \geq 4$, which is given by the hypothesis that $n \geq 16$.

Next, we turn to the second part of the proof, namely that for any $k \geq 5$ it also holds $p_1 > p_k$, again under the hypothesis that $n \geq 16$. Recall that p_k is the probability that the offspring is accepted

conditioned on the event that the mutation strength is equal to k. Based on Equation 2 we deduce,

$$\begin{aligned} p_k &\leq \frac{1}{2} \sum_{i=0}^{\lceil k/2 \rceil - 1} \binom{k}{i} (1/n)^{k-i} \\ &\leq \frac{1}{2n^{\lceil k/2 \rceil}} \sum_{i=0}^{\lceil k/2 \rceil - 1} \binom{k}{i} \\ &\leq \frac{2^k}{2n^{\lceil k/2 \rceil}} \leq \frac{2^k}{2n^{k/2}}, \end{aligned} \tag{3}$$

where in the first step we invoke that n^{i-k} is a function that is monotonically increasing in i while the last line follows by the identity of a binomial coefficient $\sum_{i=0}^{k} \binom{k}{i} = 2^k$. Recall that we assume that $k \geq 5$ and $n \geq 16$. Moreover, by previous calculations, we know that $p_1 = \frac{1}{2n}$ and $p_k \leq \frac{2^k}{2n^{k/2}}$. Finally, combining these facts and putting everything together now yields

$$p_k \leq \frac{2^k}{2n^{k/2}} = \frac{2^k}{2n^{k/4} n^{k/4}} \leq \frac{1}{2n^{k/4}} < \frac{1}{2n} = p_1,$$

where the second inequality follows since $n \geq 16$, while in the final inequality we use $k \geq 5$. Based on the argumentation of the outline of the proof, this finishes the second and final part of the proof. □

Building on top of this statement, we can iterate the result and find that, for our previous target function f_a, the same statement is true for any time step t, i.e, the mutation strength is chosen to be $k = 1$ in order to maximize the probability to accept the offspring.

Theorem 15. *Let $n \in \mathbb{N}_{\geq 16}$, $r \in \mathbb{Z}$ and $a \in \mathbb{Z}^n$ such that $a_1 = r$ and for any $i \in \{2, \ldots, n\}$, $a_i = 0$. Moreover, for any $t \in \mathbb{Z}^+$, let $k_t \in \mathbb{Z}^+$ be the mutation strength maximizing the probability that the offspring gets accepted, at any iteration step t of the (1+1) EA optimizing f_a with ℓ_1-symmetric operator. Then, at any iteration step $t \geq 1$, the mutation strength is given by $k_t = 1$.*

Proof. Intuitively, the statement holds since throughout all iterations only the integer x_1 can be modified, while all other integers remain 0. Thus, at any time step t, the problem is equivalent to the same problem as at time step $t = 1$ with the same respective starting and target string, except for a potentially different integer a_1. In order to formalize this idea for any $t \geq 1$, we proceed by induction.

Let us start with base ($t = 1$). We notice that the statement for $t = 1$ is immediately true by applying Lemma 14. For the induction step, suppose that $k_1, k_2, \ldots, k_{t-1} = 1$, i.e., the mutation strength before step t has always been equal to 1. Moreover, let $x' \in \mathbb{Z}^n$ be the integer string that is the accepted offspring after $t - 1$ iterations. Recall that the target function a differs from the initial all zero string 0^n only in the first coordinate. Thus, we observe that given the mutation strength is 1, a fitness improvement in the offspring is feasible if and only if the integer in the first position of the current string gets mutated (in the right direction). Since previous mutation strengths are all 1, we have for any $i \in \{2, \ldots, n\}$, $x'_i = 0$ and $-t + 1 \leq x'_1 \leq t - 1$.

Now, assume that $x' \neq a$ as otherwise we are done. W.l.o.g. let $r \in \mathbb{Z}^+$ (the other case follows by symmetry). It follows that $|x'_1| < r$ as x'_1 cannot be larger than r, using the induction hypothesis of the mutation strength $k_1, k_2, \ldots, k_{t-1} = 1$. Let $a' = a - x'$, i.e., the target string a except for the first position a_1 (the integer in the first position differs by the progress made in previous iterations). Then, notice that the problem at time step t is equivalent to target string a' with starting string 0^n. Invoking Lemma 14 the probability that the offspring gets accepted is maximized by $k_t = 1$. Hence, this equivalently holds for iteration t on the original string x'. Therefore, the (1+1) EA optimizing f_a with ℓ_1-symmetric operator picks $k_t = 1$ in order to maximize the probability of acceptance. □

This directly implies the following statement regarding a lower bound for the run time where we assume that the mutation strength that maximizes the desired probability is picked by an oracle.

Corollary 16. *Let $n \in \mathbb{N}_{\geq 16}$, $r \in \mathbb{Z}$ and $a \in \mathbb{Z}^n$ such that $a_1 = r$ and for any $i \in \{2, \ldots, n\}$, $a_i = 0$. Then the optimization time for the (1+1) EA optimizing f_a with ℓ_1-symmetric operator with mutation strength maximizing the probability of the offspring to be accepted is $\Omega(|a|_1)$.*

While most practical algorithms that adjust the mutation strength focus on success probability, it might be more desirable to adjust and maximize the *expected gain in fitness*. However, we can similarly show that this quantity is optimized by choosing a mutation strength of $k = 1$. The proof works analogously to the one given previously for Lemma 14 and the details can be found in the supplementary material.

Theorem 17. *Let $n \in \mathbb{N}_{\geq 64}$, $r \in \mathbb{Z}$ and $a \in \mathbb{Z}^n$ such that $a_1 = r$ and for any $i \in \{2, \ldots, n\}$, $a_i = 0$. Let x^t be the parent integer string at iteration t while the (1+1) EA with ℓ_1-symmetric operator is optimizing f_a. Moreover, let $k \in \mathbb{Z}^+$ be the mutation strength corresponding to ℓ_1-symmetric operator. Then, the expected gain, $E[f_a(x^{t-1}) - f_a(x^t)]$ at time step $t = 1$ is maximized by $k = 1$.*

For completeness we spell out the straightforward results regarding a lower bound for the run time if the expected gain is to be maximized. By an almost verbatim repeat of the proof of Theorem 15 we now get the equivalent results for the mutation strength that aims to maximize the expected gain.

Corollary 18. *Let $n \in \mathbb{N}_{\geq 64}$, $r \in \mathbb{Z}$ and $a \in \mathbb{Z}^n$ such that $a_1 = r$ and for any $i \in \{2, \ldots, n\}$, $a_i = 0$. Moreover, for $t \in \mathbb{Z}^+$, let $k_t \in \mathbb{Z}^+$ be the mutation strength maximizing the expected gain of the offspring, at any iteration step t of the (1+1) EA optimizing f_a with ℓ_1-symmetric operator. Then, at any iteration step $t \geq 1$, the mutation strength is given by $k_t = 1$.*

In similar fashion it now also follows the run time $\Omega(|a_1|)$.

Corollary 19. *Let $n \in \mathbb{N}_{\geq 64}$, $r \in \mathbb{Z}$ and $a \in \mathbb{Z}^n$ such that $a_1 = r$ and for any $i \in \{2, \ldots, n\}$, $a_i = 0$. Then the optimization time for the (1+1) EA optimizing f_a with ℓ_1-symmetric operator with mutation strength maximizing the expected gain of the offspring is $\Omega(|a|_1)$.*

7 EXPERIMENTAL ANALYSIS OF PERFORMANCE

In this section we provide an empirical analyses among different algorithms with different mutation operators. We compare the performances of (1+1) EA with the ±1 operator, (1+1) EA with the

heavy-tailed operator, the RLS with the self-adjusting operator and the (μ, λ) GA with ℓ_1-symmetric operator.

All our theoretical analyses are concerned with the unbounded integer search space, whereas, for practical reasons, we have to restrict our search space to be bounded. We bound the maximum value of the considered integer strings by a value r.

We set the optimum as the all-r integer string and let the algorithm run until it finds the optimum and record the run time (number of function evaluations). For (1+1) EA with different mutation operators and $RLS_{\alpha,\beta}$, we start with the all-0 integer string. As for the algorithm with ℓ_1-symmetric operator, the author in [14] proposes the algorithm which fundamentally operates as a (μ, λ) genetic algorithm (GA). Subsequently, the authors of [15] have developed code that builds upon this algorithm. For replication purposes, we utilize the code made publicly available by the authors of [15] on GitHub.[1]

We choose the step size of r and the parameter setup for each algorithm as follows:

(1) (1+1) EA with ±1 operator: r from 10 till 150 with a step size of 10. Then we consider $r \in \{10^3, 10^4, 10^5\}$ to analyze the run time of the algorithm for exponentially increasing values of r.
(2) $RLS_{\alpha,\beta}$ with the self-adjusting operator and the (1+1) EA with the heavy-tailed operator: we consider $r = 10^k$, for $k \in \{1, \ldots, 9\}$. For RLS with the self-adjusting operator we set the parameter $\alpha = 2.0$ and $\beta = 0.5$. For the (1+1) EA with the heavy-tailed operator we set $\epsilon = 0.001$ and the step size is generated according to Definition 2.
(3) (μ, λ) GA with ℓ_1-symmetric operator: We set the $\mu = 4$, $\lambda = 28$, which is the same as [14]. Similar to the (1+1) EA with ±1 operator, we consider $r = 10^k$, for $k \in \{1, \ldots, 5\}$.

We present experimental results of all in Figure 1. All results are averaged over 20 independent runs. We attach the results of the statistical test for $n = 100$ in the appendix.

In Figure 1 we can see that the scaling behavior with respect to r is independent of the value of n.

Asymptotically, the results are as suggested by the theoretical results given in the prior sections. However, for small values of r, the scaling behavior is not yet the deciding factor. In particular, the ±1 operator is competitive as long as the optimum is not much more than $r = 10^2 = 100$ away in each component. For higher values of r, the constantly small step size is very much detrimental to efficient search.

An interesting finding is that the heavy-tailed operator can outperform the self-adjusting RLS, in spite of what the asymptotic bounds given in this paper suggest. For small values of r, a lot of time is wasted on attempting larger jumps, but for middle-ranged r these jumps start to pay off. In contrast, the self-adaptive RLS needs a warm-up phase adjusting its velocity value, meanwhile the heavy-tailed operator can make progress starting in the first iteration.

[1] https://github.com/jacobdenobel/algorithms/blob/main/algorithms/unbounded_integer_ea.py

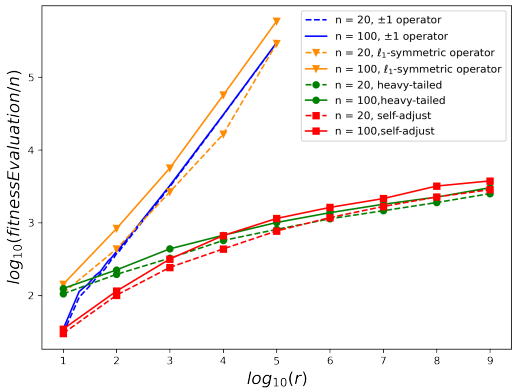

Figure 1: Comparison of all algorithms. We record the number of fitness evaluations as run time. Different colors together with markers stand for different mutation operators and different line styles stand for different values of n. Note that both axes are logarithmically scaled.

8 CONCLUSION

In this paper we addressed $\mathbb{Z}^n$ as a search space and lifted results for multi-valued decision variables to this search space. We further introduced another operator which uses a very heavy-tailed distribution to address the a priori unknown distance to the optimum (for bounded search spaces, some bound is known). Still, asymptotically optimal is a self-adjusting strategy already known from bounded search spaces. Furthermore, we gave good reasons why ℓ_1-symmetric operators are not competitive.

This work is one step towards tackling mixed-integer black box optimization problems. Other test functions need to follow, particularly test functions which have both continuous and discrete variables. A particular challenge is to give an interesting performance measure: while for discrete problems the focus is on the hitting time of finding the global optimum, continuous algorithms are typically evaluated in their ability to quickly approximate local optima.

REFERENCES

[1] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. 2022. Fast Mutation in Crossover-Based Algorithms. *Algorithmica* (2022), 1724–1761. https://doi.org/10.1007/s00453-022-00957-5
[2] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. 2023. Lazy Parameter Tuning and Control: Choosing All Parameters Randomly from a Power-Law Distribution. *Algorithmica* (2023), 442–484. https://doi.org/10.1007/s00453-023-01098-z
[3] Anh Viet Do, Mingyu Guo, Aneta Neumann, and Frank Neumann. 2021. Analysis of Evolutionary Diversity Optimisation for Permutation Problems *(GECCO '21)*. 574–582. https://doi.org/10.1145/3449639.3459313
[4] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2015. Solving Problems with Unknown Solution Length at (Almost) No Extra Cost *(GECCO '15)*. 831–838. https://doi.org/10.1145/2739480.2754681
[5] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2017. Static and Self-Adjusting Mutation Strengths for Multi-valued Decision Variables. *Algorithmica* 80 (2017). https://doi.org/10.1007/s00453-017-0341-1
[6] Benjamin Doerr, Yassine Ghannane, and Marouane Ibn Brahim. 2022. Towards a Stronger Theory for Permutation-Based Evolutionary Algorithms *(GECCO '22)*. 1390–1398. https://doi.org/10.1145/3512290.3528720

[7] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative Drift Analysis. *Algorithmica* 64, 4 (2012), 673–697. https://doi.org/10.1007/s00453-012-9622-x

[8] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. 2017. Fast Genetic Algorithms *(GECCO '17)*. 777–784. https://doi.org/10.1145/3071178.3071301

[9] K. W. Fang. 1990. *Symmetric Multivariate and Related Distributions* (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781351077040

[10] Jun He and Xin Yao. 2004. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing* 3 (2004), 21–35. https://doi.org/10.1023/B:NACO.0000023417.31393.c7

[11] Timo Kötzing and Martin S. Krejca. 2019. First-hitting times under drift. *Theoretical Computer Science* 796 (2019), 51–69.

[12] Per Kristian Lehre and Carsten Witt. 2012. Black-Box Search by Unbiased Variation. *Algorithmica* 64, 4 (2012), 623–642. https://doi.org/10.1007/S00453-012-9616-8

[13] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press.

[14] Günter Rudolph. 1994. An evolutionary algorithm for integer programming. In *Parallel Problem Solving from Nature — PPSN III*, Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer (Eds.). 139–148.

[15] André Thomaser, Jacob De Nobel, Diederick Vermetten, Furong Ye, Thomas Bäck, and Anna Kononova. 2023. When to be Discrete: Analyzing Algorithm Performance on Discretized Continuous Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '23)*. 856–863. https://doi.org/10.1145/3583131.3590410

A STATISTICAL TEST FOR RUN TIME COMPARISON

In this section we present the results of the statistical test. For each algorithm, we use one box plot to show the distribution of independent runs for $n = 100$. Each box is the first quartile ($Q1$) to the third quartile ($Q3$) of the group. The whiskers extend the box by 1.5 times the interquartile range (IQR). Dots are outliers which pass whiskers.

For the other set up $n = 20$, we get a similar box plot.

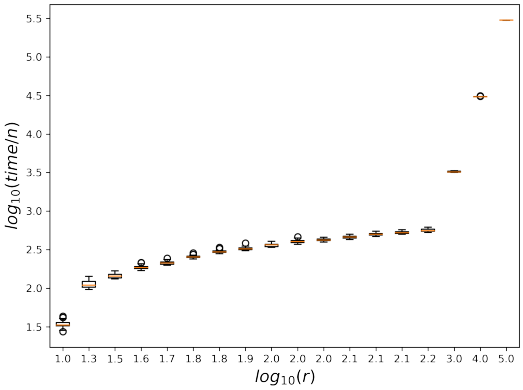

Figure 2: (1+1) EA with ±1 operator

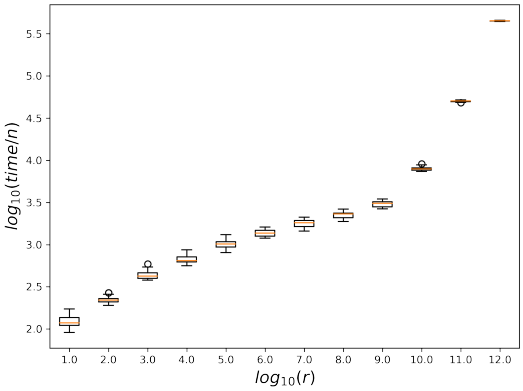

Figure 3: (1+1) EA with *heavy-tailed operator.*

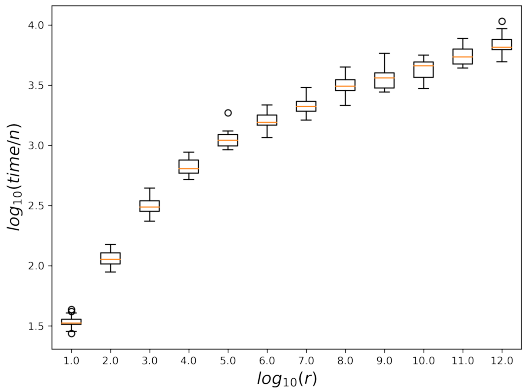

Figure 4: RLS with the self-adjusting operator.

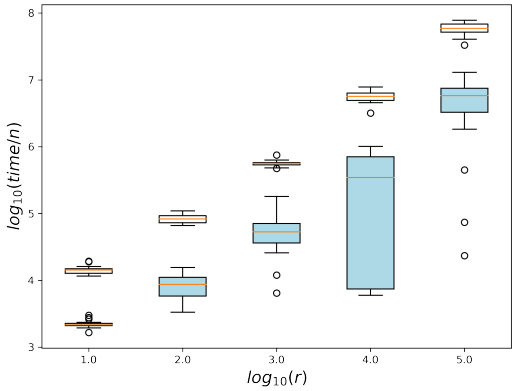

Figure 5: (μ, λ) GA with geometric step mutation operator. The blue box is for $n = 20$.

Lemma 20. *[5, Lemma 13] Let $n \in \mathbb{N}$ be fixed, let q be a cost function on elements of $[n]$ and let c be a cost function on subsets of $[n]$. Furthermore, let a random variable S ranging over subsets of $[n]$ be given. Then we have*

$$\forall T \subseteq [n] : c(T) \leq \sum_{i \in T} q(i) \implies E[c(S)] \leq \sum_{i=1}^{n} q(i) P(i \in S) \quad (4)$$

and

$$\forall T \subseteq [n] : c(T) \geq \sum_{i \in T} q(i) \implies E[c(S)] \geq \sum_{i=1}^{n} q(i) P(i \in S) \quad (5)$$

Theorem 21. *Let $f_a \in \mathcal{F}$. Then the expected optimization time of the (1+1) EA with ±1 operator starting with all 0 integer string on f_a is $O(n \cdot (|a|_\infty + \log(|a|_H)))$.*

Proof. Our proof idea is similar to the proof idea in [5, Theorem 12]. We will use the multiplicative drift analysis (see Theorem 4) to

prove this theorem. For our analysis, we make use of two edge cases that both have a fitness of n; which would be only one entry being incorrect but n away from its target and the case that every entry in the target is 1. The former is hard to optimize since it takes long for the algorithm to successively change the incorrect position, while the latter can be solved very fast since there are multiple ways of progressing in each step. We exploit this by giving each position a weight exponential in the amount that is incorrect, and then sum over those weights. With any search point $x \in \Omega$ we associate a vector $d \in \mathrm{R}^n$ such that, for all $i \leq n, d_i = |a_i - x_i|$. Given some $\omega > 1$ we consider the potential

$$g(x) \coloneqq \sum_{i=1}^{n} (\omega^{d_i} - 1) \tag{6}$$

Let x_t be the integer string at iteration t when the (1+1) EA with ± 1 operator is optimizing f_a. Let $X_t = g(x_t)$ and let $T = \min\{t \geq 0 \mid X_t = 0\}$. Further let E_1 be the event that X_{t+1} is obtained by mutating exactly one position and let E_2 be the event that X_{t+1} is obtained by mutating at least two positions. Then $X_0 \leq |a|_H \cdot \omega^{|a|_\infty}$, since we start with all 0 integer string. We denote O as the set of already optimized positions and $A(S)$ as the set of accepted offspring by only manipulating $S \subseteq [n]$ positions. Now we bound the expected drift. Since $t < T$, at least one of the position is at least 1 distant far from the optimum and the probability to mutate this position in the right direction is $\frac{1}{2n}$ and the probability to not mutate any other positions is $\left(1 - \frac{1}{n}\right)^{n-1}$. Therefore,

$$\begin{aligned}
&E[X_t - X_{t+1} \mid t < T, E_1] \cdot P(E_1) \\
&\quad \geq \sum_{i \in [n] \setminus O} \frac{1}{2n} \left(1 - \frac{1}{n}\right)^{n-1} (\omega^{d_i} - 1) - (\omega^{d_i - 1} - 1) \\
&\quad \geq \frac{1}{2ne} \sum_{i \in [n] \setminus O} (\omega^{d_i} - 1) - (\omega^{d_i - 1} - 1) \\
&\quad = \frac{1}{2ne} \sum_{i \in [n] \setminus O}^{n} \left(1 - \frac{1}{\omega}\right) \cdot \omega^{d_i} \\
&\quad \geq \frac{1}{2ne} \sum_{i=1}^{n} \left(1 - \frac{1}{\omega}\right) \cdot \left(\omega^{d_i} - 1\right) \\
&\quad = \frac{\omega - 1}{2\omega ne} \sum_{i=1}^{n} (\omega^{d_i} - 1)
\end{aligned}$$

Let U be the set of all tuples $(y, i), y \in A(S)$ where the i-th positions worsens. As we only consider accepted mutations, we have that, for all $y \in A(S), \sum_{i \in S} d(y_i, a_i) - d_i \leq 0$. This implies that there are at least as many improving-pairs as there are worsening-pairs in $A(S) \times S$. For every (y, i) with $d_i \neq 0$ where bit position i changes in the wrong direction, there is a $(y', i) \in U$ with bit position i changing in the right direction and the remaining positions behaving the same. The change in potential for both pairs added is

$$\omega^{d_i + 1} - \omega^{d_i} + \omega^{d_i - 1} - \omega^{d_i} = \omega^{d_i} \frac{(\omega - 1)^2}{\omega}$$

Since there are in total at least as many improving-pairs as worsening-pairs, we can further map injectively each (y, i) with a correct position ($d_i = 0$) that changes in the wrong direction to another tuple (y', j) for $y' \in A(S)$ with some improving position j. We define a function

$$\overline{d}(y, i) \coloneqq |a_j - y'_j|.$$

The change in potential for both pairs added is

$$\begin{aligned}
\omega - 1 + \omega^{\overline{d}(y,i) - 1} - \omega^{\overline{d}(y,i)} &= \left(1 - \omega^{\overline{d}(y,i) - 1}\right) \cdot (\omega - 1) \\
&\leq 0 < \omega^{d_i} \frac{(\omega - 1)^2}{\omega}.
\end{aligned}$$

Let Y be the random variable describing the search point after one cycle of mutation and selection. The random variable Y is completely determined by choosing a set $S \subseteq [n]$ of bit positions to change in x and then, for each such position $i \in S$, choosing whether to change towards or away from the target. For each possible $S \subseteq [n]$, let $Y(S)$ be the random variable Y conditional on making changes exactly at the bit positions of S. Note that since we increase/decrease each index by 1 with the same probability, $Y(S)$ is the uniform distribution on $A(S)$. Further let

$$\begin{aligned}
c(S) &\coloneqq E[g(Y(S)) - g(x)] \\
&= \frac{1}{|A(S)|} \sum_{y \in A(S)} g(y) - g(x) \\
&= \frac{1}{|A(S)|} \sum_{y \in A(S)} \sum_{i=1}^{n} \left(\omega^{|y_i - a_i|} - \omega^{d_i}\right) \\
&\leq \frac{1}{|A(S)|} \sum_{(y,i) \in U} \left(\omega^{d_i + 1} - \omega^{d_i} + \omega^{\overline{d}(y,i) - 1} - \omega^{\overline{d}(y,i)}\right) \\
&\leq \frac{1}{2} \sum_{i \in S} \omega^{d_i} \frac{(\omega - 1)^2}{\omega}.
\end{aligned}$$

Using Lemma 20, we get that

$$E[g(Y) - g(x) | E_2] \leq \sum_{i=1}^{n} \frac{1}{n} \omega^{d_i} \frac{(\omega - 1)^2}{2\omega} = \frac{(\omega - 1)^2}{2\omega n} \sum_{i=1}^{n} \omega^{d_i}$$

We can use any $\omega > 1$ such that $\omega - 1 - e(\omega - 1)^2 > 0$ and set $c = (\omega - 1 - e(\omega - 1)^2)/e$. One can verify that for $\omega = 1.2$ we obtain $0.0912687 > 0$, so such ω exist. In total we get

$$\begin{aligned}
E[g(x) - g(Y)] &\geq \frac{\omega - 1}{2\omega ne} \sum_{i=1}^{n} \omega^{d_i} - \frac{(\omega - 1)^2}{2\omega n} \sum_{i=1}^{n} \omega^{d_i} \\
&= \frac{\omega - 1 - e(\omega - 1)^2}{2\omega ne} \sum_{i=1}^{n} \omega^{d_i} = \frac{c}{2\omega n} \sum_{i=1}^{n} \omega^{d_i} \geq \frac{c}{2\omega n} g(x).
\end{aligned}$$

Therefore by the multiplicative drift theorem (Theorem 4), we have

$$E[T] = O((|a|_\infty + \log(|a|_H)) \cdot n) = O(|a|_\infty \cdot n + n \cdot \log(|a|_H)).$$

□

Lemma 22 (RLS upper bound). *Let $f_a \in \mathscr{F}$. For constants α, β satisfying $1 < \alpha \leq 2, 1/2 < \beta \leq 0.9, 2\alpha\beta - \beta - \alpha > 0, \alpha + \beta > 2$ and $\alpha^2\beta > 1$ the expected optimization time of $RLS_{\alpha,\beta}$ starting with all 0 integer string on f_a is $O(n \cdot \log(|a|_1))$.*

Proof. To simplify the notation for a given search point x and the target integer string z and the chosen metric d, we let $d_i = d(x_i, z_i)$ for all $(i \leq n)$ be the distance vector of x to z. Thus, the goal is to reach a state in which the distance vector is $(0, ..., 0)$. We

now want to define a potential function in dependence on (d, v) (where of course d is dependent on x) such that it is 0 when d is $(0, ..., 0)$ and strictly positive for any $x \neq (0, ..., 0)$.

We use as potential function the following map $g : \mathbb{Z}^n \mapsto \mathbb{R}, (x, v) \mapsto \sum_{i=1}^{n} g_i(d_i, v_i)$ where $g_i(d_i, v_i) := 0$ for $d_i = 0$ and for $d_i \geq 1$

$$g_i(d_i, v_i) := d_i + \begin{cases} cd_i \max\{2v_i/d_i, d_i/(2v_i)\}, & \text{if } v_i \leq 2\beta d_i; \\ cd_i \max\{2v_i/d_i, d_i/(2v_i)\} + pd_i, & \text{otherwise.} \end{cases}$$

and c, p are (small) constants specified below. For further motivation on the potential see [5, Theorem 17].

Summarizing all the conditions needed below, we require that the constants α, β, c, p satisfy $1 < \alpha \leq 2, 1/2 < \beta \leq 0.9, 2\alpha\beta - \beta - \alpha > 0, \alpha + \beta > 2, \alpha^2\beta > 1, 8\alpha\beta c + 2p + 4c/\beta \leq 1/16, p > 8c\left(\frac{\alpha+\beta}{2} - 1\right)$, and $p > 4(\alpha - 1)c > 0$.

We can thus choose, for example, $\alpha = 1.7, \beta = 0.9, p = 0.01$, and $c = 0.001$.

Let $d \neq (0, ..., 0)$ and $v \in \mathbb{N}^n$. Let (d', v') be the state of Algorithm 2 started in (d, v) after one iteration (i.e., after a possible update of x and v). First we show that the expected difference in potential satisfies

$$E[g(d, v) - g(d', v')|d, v] \geq \frac{\delta}{n} g(d, v)$$

for some positive constant δ. Any fixed index i is chosen by Algorithm 2 for mutation with probability $1/n$; for all i, let A_i be the event that index i was chosen. We show that there is a constant δ such that, for all indices i with $d_i \neq 0$

$$E[g(d_i, v_i) - g(d_i', v_i') \mid d, v] \geq \delta g_i(d_i, v_i)$$

thus proving the claim using $P(A_i) = 1/n$.

We regard several cases, depending on how d_i and v_i relate.

Case 1: $v_i \leq d_i/8$.

First we observe that $\max\{2v_i/d_i, d_i/(2v_i)\} = d_i/(2v_i)$. The contribution of the i-th position to the current potential is thus

$$g_i(d_i, v_i) = d_i + cd_i^2/(2v_i).$$

With probability $1/2$ the algorithm decides to move in the right direction. In this case we make progress with respect to the fitness function and the velocity. That is, after the iteration we have $d_i' = d_i - \lfloor v_i \rfloor < d_i$ and $v_i' = \alpha v_i > v_i$.

To bound the progress in the second component of g_i, we observe that

$$\begin{aligned} cd_i' \max\{2\alpha v_i/d_i', d_i'/(2\alpha v_i)\} &= \max\{2c\alpha v_i, cd_i'^2/(2\alpha v_i)\} \\ &= cd_i'^2/(2\alpha v_i), \end{aligned}$$

where the second equality follows from $2\alpha v_i \leq d_i/2 < d_i'$. We thus obtain that for this case the difference in potential is at least

$$\begin{aligned} g_i(d_i, v_i) - g_i(d_i', v_i') &= d_i + cd_i^2/(2vi) - d_i' - cd_i'^2/(2\alpha v_i) \quad (7) \\ &\geq \frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2\alpha v_i}. \quad (8) \end{aligned}$$

With probability $1/2$ the algorithm decides to go in the wrong direction, then $d_i' > d_i$ holds and the new individual is discarded while the velocity v_i at position i is further decreased to $\max\{\beta v_i, 1\} \geq \beta v_i$. Hence the difference in potential for this case is at least

$$g_i(d_i, v_i) - g_i(d_i, \beta v_i) = \frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2\beta v_i}. \quad (9)$$

Combining (7) and (9), we thus obtain that the expected difference in potential is at least

$$\begin{aligned} &\frac{1}{2}\left(\frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2\alpha v_i} + \frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2\beta v_i}\right) = \frac{cd_i^2}{2v_i}\left(\frac{2\alpha\beta - \beta - \alpha}{2\alpha\beta}\right) \\ &= \left(\frac{2\alpha\beta - \beta - \alpha}{4\alpha\beta}\right)\left(\frac{cd_i^2}{2v_i} + \frac{cd_i^2}{2v_i}\right) \geq \left(\frac{2\alpha\beta - \beta - \alpha}{4\alpha\beta}\right)\left(4cd_i + \frac{cd_i^2}{2v_i}\right) \\ &\geq \left(\frac{2\alpha\beta - \beta - \alpha}{4\alpha\beta}\right)\min\{4c, 1\}\left(d_i + \frac{cd_i^2}{2v_i}\right) \\ &= \left(\frac{2\alpha\beta - \beta - \alpha}{4\alpha\beta}\right)\min\{4c, 1\} g_i(d_i, v_i) \end{aligned}$$

where in the third step we have used the requirement that $v_i \leq d_i/8$.

Case 2: $d_i/8 < v_i \leq 2\beta d_i$.

Now we are in a range of velocity which is well-suited to make progress. In fact, every step towards the optimum decreases the distance to the optimum by at least the minimum of $\lfloor d_i \rfloor /8$ (if v_i is close to $d_i/8$ and we hence do not overshoot the target) and $\lfloor (2 - 2\beta)d_i \rfloor$ (if $v_i = 2\beta d_i \geq d_i$ in which case we overshoot the target and the distance to it from d_i to at most $\lfloor 2\beta d_i \rfloor - d_i$). In case of moving towards the target value, the change in the first term of g_i is thus at least

$$\min\{\lfloor d_i/8 \rfloor, \lfloor (2 - 2\beta)d_i \rfloor\} = \lfloor d_i/8 \rfloor,$$

using $\beta \leq 0.9$. However, note that the decrease is at least 1 (since v_i is at least 1). Furthermore, we have, for all $z \geq 8, z/16 \leq \lfloor z/8 \rfloor$. Thus, we always have a decrease of at least $d_i/16$.

We now compute the change in the second term of g_i. Regard first the case that $\max\{2v_i'/d_i', d_i'/(2v_i')\} = 2v_i'/d_i'$. In this case, we pessimistically assume that the previous contribution of the second term in $g_i(d_i, v_i)$ was zero. This contribution increases to at most

$$2cv_i' + pd_i' \leq 2\alpha c v_i + pd_i' \leq 2\alpha c v_i + pd_i \leq (4\alpha\beta c + p)d_i. \quad (10)$$

On the other hand, $\max\{2v_i'/d_i', d_i'/(2v_i')\} = d_i'/(2v_i')$ and the previous contribution of the second term in $g_i(d_i, v_i)$ was $cd_i^2/(2v_i)$, then the contribution of this second term has been decreased to $c(d_i')^2/(2\alpha v_i) \leq cd_i^2/(2v_i)$. The change in contribution is thus positive in this case, and therefore in particular strictly larger than $-(4\alpha\beta c + p)d_i$. We finally need to regard the case that

$$\max\{2v_i'/d_i', d_i'/(2v_i')\} = d_i'/(2v_i')$$

and

$$\max\{2v_i/d_i, d_i/(2v_i)\} = 2v_i/d_i.$$

In this case the contribution in the second term of g_i increases by at most

$$\frac{cd_i'^2}{2\alpha v_i} \leq \frac{cd_i^2}{2\alpha(d_i/8)} \leq \frac{4cd_i}{\alpha} \leq 4\alpha\beta c d_i,$$

where the last step follows from $\alpha^2\beta \geq 1$.

Summarizing this discussion, we see that in case of stepping towards the target the change in progress satisfies

$$g_i(d_i, v_i) - g_i(d_i', v_i') \geq d_i(1/16 - (4\alpha\beta c + p)), \tag{11}$$

which is positive by our conditions on c and p.

Let us now regard the case of stepping away from the optimum, which happens with probability $1/2$ and the velocity is decreased to $\max\{\beta v_i, 1\}$. Assume first that $\max\{\beta v_i, 1\} = \beta v_i$. Then,

$$g_i(d_i, v_i) - g_i(d_i', v_i') = \max\{2cv_i, \frac{cd_i^2}{2v_i}\} - \max\{2c\beta v_i, \frac{cd_i^2}{2\beta v_i}\}. \tag{12}$$

If $\max\{2c\beta v_i, cd_i^2/(2\beta v_i)\} = cd_i^2/(2\beta v_i)$, then the term in (12) is at least $-cd_i^2/(2\beta v_i) \geq -4cd_i/\beta$ by our condition $d_i/8 \leq v_i$. Furthermore, if $\max\{2c\beta v_i, cd_i^2/(2\beta v_i)\} = 2c\beta v_i$, then (12) is strictly positive as can be seen by the following observation

$$\max\{2cv_i, \frac{cd_i^2}{2v_i}\} - 2c\beta v_i \geq 2cv_i - 2c\beta v_i > 0. \tag{13}$$

Putting everything together we thus obtain that for $d_i/8 \leq v_i \leq 2\beta d_i$

$$E[g_i(d_i, v_i) - g_i(d_i', v_i')] \leq \frac{d_i}{2}(1/16 - 2(4\alpha\beta c + p) - 4c/\beta) \tag{14}$$

which is positive if $8\alpha\beta c + 2p + 4c/\beta \leq 1/16$. Since $v_i = \Theta(d_i)$ this also shows that there is a positive constant δ such that $E[g_i(d_i, v_i) - g_i(d_i', v_i')] \geq \delta g_i(d_i, v_i)$.

We finally need to regard the case that $\max\{\beta v_i, 1\} = 1$. Intuitively, the cap can only make our situation better. This is formalized by the following computations. We need to bound

$$g_i(d_i, v_i) - g_i(d_i', v_i') = \max\{2cv_i, \frac{cd_i^2}{2v_i}\} - \max\{2c, \frac{cd_i^2}{2}\}. \tag{15}$$

As above we obtain positive drift for the case $\max\{2c, cd_i^2\} = 2c$ by observing that $\max\{2cv_i, \frac{cd_i^2}{2v_i}\} - 2c \geq 2cv_i - 2c \geq 0$ (using that $v_i \geq 1$). For the case $\max\{2c, cd_i^2\} = cd_i^2$ the term in (15) is at least $-cd_i^2 \geq -cd_i^2/(2\beta v_i) \geq -4cd_i/\beta$ as above. The same computation as above thus shows a positive multiplicative gain in g_i.

Case 3: $2\beta d_i < v_i < 2d_i$.

Under these conditions $g_i(d_i, v_i) = d_i + 2cv_i + pd_i$ holds.

As before, we first regard the case that the algorithm moves towards the target value. Since $\beta \geq 1/2$ it holds that $d_i \leq 2\beta d_i < v_i$ and the target value is thus overstepped. However, due to the requirement $v_i < 2d_i$, the distance of the offspring is strictly smaller than the previous distance. The velocity is hence increased to αv_i.

With probability $1/2$ the algorithm does a step away from the goal and thus the velocity is reduced to $v_i' = \max\{\beta v_i, 1\}$. Regard first the case that $v_i' = \beta v_i$. Then, due to $\beta v_i < 2\beta d_i$, the penalty term pd_i is no longer applied and the resulting potential at component i is thus $g_i(d_i', v_i') = d_i + 2c\beta v_i$.

Ignoring any possible gains in d_i, we therefore obtain that the expected difference in the potential is at least

$$2cv_i\left(1 - \frac{\alpha+\beta}{2}\right) + \frac{p}{2}d_i$$

Note that $1 - (\alpha+\beta)/2$ is negative, since we require $\alpha + \beta > 2$. Using $v_i \leq 2d_i$ we see that the drift is at least

$$4cd_i\left(1 - \frac{\alpha+\beta}{2}\right) + \frac{p}{2}d_i = d_i\left(\frac{p}{2} - 4c\left(\frac{\alpha+\beta}{2} - 1\right)\right).$$

Since $p > 8c(\frac{\alpha+\beta}{2} - 1)$ this expression is positive. Furthermore, we have $g_i(d_i, v_i) = \Theta(d_i)$, yielding the desired multiplicative drift.

For $v_i' = 1$ we first observe that $v_i' = 1 \leq d_i \leq 2\beta d_i$ and the penalty term pd_i is thus not in force. Furthermore, we have $\beta d_i < \beta v_i \leq 1$ and thus $d_i \leq 1/\beta \leq 2$, showing that $\max\{2/d_i, d_i/2\} \leq \max\{2, 1\} = 2$. We obtain

$$E[g_i(d_i, v_i) - g(d_i', v_i') \geq \frac{p}{2}d_i - cv_i(\alpha - 1) \geq \frac{p}{2}d_i - 2(\alpha - 1)cd_i,$$

which is positive for $p/2 - 2(\alpha - 1)c > 0$.

Case 4: $v_i = 2d_i$.

Steps away from the target are not accepted, thus regardless of whether or not we move towards or away from the target, the fitness does not decrease; therefore, the velocity is decreased to βv_i (note that $v_i \geq 2$ and hence $\beta v_i \geq 1$). The previous contribution of the i-th component to $g(x)$ being $d_i + 2cv_i + pd_i = d_i(1 + 4c + p)$, and the new potential at the i-th component being $d_i(1 + 4\beta c)$, we obtain

$$E[g_i(d_i, v_i) - g_i(d_i', v_i')] = d_i(4c + p - 4\beta c),$$

which is strictly positive and linear in $g_i(d_i, v_i)$.

Case 5: $2d_i < v_i$.

Steps towards the optimum are now also not accepted, since they overstep the optimum by too much. Therefore, we always decrease the velocity to $\max\{\beta v_i, 1\} = \beta v_i$ (note that $v_i > 2$ and thus $\beta v_i > 1$) and the gain in potential is

$$2cv_i + pd_i - (2c\beta v_i + pd_i) = 2cv_i(1 - \beta) > 4(1 - \beta)cd_i$$

showing that we have the multiplicative drift as desired.

Together with the observation that the initial potential is of order at most

$$\sum_{i=1}^{n} d_i^2 \leq \left(\sum_{i=1}^{n} d_i\right)^2 = |a|_1^2$$

plugged into the multiplicative drift theorem (Theorem 4) proves the desired overall expected run time of $O(n \log(|a|_1))$. □

Theorem 23. *Let $n \in \mathbb{N}_{\geq 64}$, $r \in \mathbb{Z}$ and $a \in \mathbb{Z}^n$ such that $a_1 = r$ and for any $i \in \{2, \cdots, n\}$, $a_i = 0$. Let x^t be the parent integer string at iteration t while (1+1) EA with ℓ_1-symmetric operator is optimizing f_a. Moreover, let $k \in \mathbb{Z}^+$ be the mutation strength corresponding to ℓ_1-symmetric operator. Then, the expected gain, $E[f_a(x^{t-1}) - f_a(x^t)]$ at time step $t = 1$ is maximized by $k = 1$.*

Proof. This proof works analogously to the one given previously for Lemma 14. Throughout the proof, let $X \in \mathbb{R}_{\geq 0}$ be the random variable that is the potential gain at time step $t = 1$ (negative gains are discarded by the algorithm). We wish to compute the expected gain conditioned on the event that the mutation strength is given by $M = k$. More formally, by the conditional expectation of X, $\mathbb{E}[X|M = k] = \sum_{x \in \mathbb{N}} x \cdot Pr(X = x|M = k)$. Then, the statement of the lemma is equivalent to $\mathbb{E}[X|M = 1] > \mathbb{E}[X|M = k]$ for $k \geq 2$.

Recall that p_k is the probability that the offspring is accepted, given the mutation strength k. In a first step we use p_k as stated in Equation (1) in order to derive the expected gain if the mutation

strength is equal to k. Then, for small values of k (for $k \leq 6$), we show that the expectation is largest if $k = 1$. The second step uses a rough upper bound to estimate $\mathbb{E}[X|M = k]$ which is strong enough to show that for $k \geq 7$, the upper bound is smaller than the expected gain given mutation strength $k = 1$.

Now, as previously pointed out, let us start by deriving an upper bound on $\mathbb{E}[X|M = k]$ via p_k. Notice that the maximal support of the random variable X is given by k if conditioned on the event that the mutation strength is $M = k$. Thus, with this observation in conjunction with p_k being the sum over probability mass where the gain is positive, we get by applying Equation (2),

$$\mathbb{E}[X|M = k] \leq k \cdot p_k \leq \frac{k}{2} \sum_{i=0}^{\lceil k/2 \rceil - 1} \binom{k}{i} (1/n)^{k-i}. \tag{16}$$

On the flip side, since the gain X is a non-negative random variable (as negative gains are discarded), the support of X conditioned on $M = 1$ is $\{0, 1\}$ and consequently $\mathbb{E}[X|M = 1] = p_k = \frac{1}{2n}$. Thus, recalling $n \geq 64$, elementary calculations via Equation (16) now reveal,

$$\begin{aligned}
\mathbb{E}[X|M = 2] &\leq 2p_2 = \frac{1}{n^2} < \frac{1}{2n} = p_1 = \mathbb{E}[X|M = 1],\\
\mathbb{E}[X|M = 3] &\leq 3p_3 = \frac{3}{2n^3} + \frac{9}{2n^2} < \frac{1}{2n} = p_1 = \mathbb{E}[X|M = 1],\\
\mathbb{E}[X|M = 4] &\leq 4p_4 = \frac{2}{n^4} + \frac{8}{n^3} < \frac{1}{2n} = p_1 = \mathbb{E}[X|M = 1],\\
\mathbb{E}[X|M = 5] &\leq 5p_5 = \frac{5}{2n^5} + \frac{25}{2n^4} + \frac{25}{n^3} < \frac{1}{2n} = p_1 = \mathbb{E}[X|M = 1],\\
\mathbb{E}[X|M = 6] &\leq 6p_6 = \frac{3}{n^6} + \frac{18}{n^5} + \frac{45}{n^4} < \frac{1}{2n} = p_1 = \mathbb{E}[X|M = 1].
\end{aligned}$$

Finally, it is left to show that similar inequalities also hold for larger k. Again, using the upper bound $\mathbb{E}[X|M = k] \leq k \cdot p_k$, in conjunction with calculations as carried out in Equation (3), yields,

$$\mathbb{E}[X|M = k] \leq k \cdot p_k \leq \frac{k2^k}{2n^{k/2}} < \frac{2^{2k}}{2n^{k/2}}.$$

To wrap things up, we get for $k \geq 7$ and using $n \geq 64$ (which entails $n^{k/6} \geq 2^k$),

$$\mathbb{E}[X|M = k] < \frac{2^{2k}}{2n^{k/6}n^{k/3}} \leq \frac{2^k}{2n^{k/3}} \leq \frac{1}{2n^{k/6}} < \frac{1}{2n} = \mathbb{E}[X|M = 1].$$

Thus, the expected gain is indeed maximized by mutation strength $M = 1$. □

Theorem 24. *Let $r \in \mathbb{Z}$ and $a = (r, 0, \ldots, 0) \in \mathbb{Z}^n$. Then there is a constant c such that the (1+1) EA which, in each iteration, chooses any ℓ_1-symmetric operator to generate offspring, requires at least $\Omega(\min(r/n^3, 2^{cn}))$ iterations to optimize f_a.*

Proof. We use the following version of a Chernoff bound [13, Theorem 4.4]. Let X be the sum of n identically distributed Bernoulli random variables and let μ be the expectation of X. Then we have, for $\delta > 0$,

$$Pr\left(|X - \mu| \geq \delta\mu\right) \leq 2\exp\left(-\delta^2\mu/3\right). \tag{CB}$$

In order to prove the theorem, for all $t \in \mathbb{N}$, let x_t be the current individual of the (1+1) EA after t iterations and let

$$X_t = |r - x_1| - 3\sum_{i=2}^{n} |x_i|.$$

We see that $X_0 = r$ and the algorithm has found the optimum if and only if $X_t = 0$; furthermore, negative values of X_t are not possible. We set up to prove a bound on the expected hitting time of 0 by using an additive drift theorem (Theorem 3).

By Theorem 12, the choice of the algorithm is equivalent to choosing a mutation strength k in any given iteration (potentially dependent on the current search point) and then deterministically drawing uniformly from the set of all offspring at ℓ_1-distance of k.

Let t be given and suppose $X_t \neq 0$. We reason conditional on X_t. Let $k \in \mathbb{Z}^+$ be the mutation strength as chosen by the algorithm in iteration $t + 1$.

Case 1: $k \leq n^3$.
Since the maximum potential gain when changing by at most k is $4k$, the total gain is at most $4n^3$.

Case 2: $k > n^3$.
In expectation, the mutation strength for any specific position is k/n. Using the Chernoff bound (CB), we see that the probability for a specific position to have a mutation strength outside of $k/n \pm \sqrt{6k\ln(nk)}/n$, using $\delta = \sqrt{6\ln(nk)/k}$, is at most

$$2\exp(-2\ln(nk)) = 2/(nk)^2.$$

Let A be the event that all of the positions have a mutation strength inside of $k/n \pm \sqrt{6k\ln(nk)}/n$. We get

$$E((X_t - X_{t+1})1[\overline{A}]) \leq E\Delta \Pr\left\{\overline{A}\right\} \leq 4k \cdot 2/(nk)^2 = O(1).$$

Thus, we now want to estimate $E(X_t - X_{t+1})1[A]$. Let now j be the number of positions orienting away from the optimum. Then, for the offspring, the loss in fitness is at least $j(k/n - \sqrt{6k\ln(nk)}/n)$ and the gain is at most $(n - j)(k/n + \sqrt{6k\ln(nk)}/n)$. Thus, the overall loss in fitness is at least

$$\begin{aligned}
&j(k/n - \sqrt{6k\ln(nk)}/n) - (n - j)(k/n + \sqrt{6k\ln(nk)}/n)\\
&\qquad = (2j - n)k/n - \sqrt{6k\ln(nk)}.
\end{aligned}$$

If $2j - n \geq 1$, this is positive (using $k > n^3$), the offspring is thus not viable. This shows that at most half the bit positions orient away from the target.

Case 2.1: At least $n/100$ of the bit positions are at most $99k/(100n)$ away from the target.
Thus, moving any of these positions a distance of $k/n \pm \sqrt{6k\ln(nk)}/n$ towards the target gains at most

$$98k/(100n) + \sqrt{6k\ln(nk)}/n.$$

Since moving away from the target still looses the full amount, a straightforward application of a Chernoff bound shows that the offspring is not viable with probability $1 - O(2^{cn})$, for some c. In case of accepted offspring, we pessimistically assume a full potential gain of $\Theta(r)$.

Case 2.2: Less than $n/100$ of the bit positions are at most $99k/(100n)$ away from the target.
We call positions that are at least $99k/(100n)$ away from the target *large*.

We let Z be the set of all mutation-strings which, when added to the current bit string, will lead to an accepted offspring. Let $Z_0 \subseteq Z$ be that subset of mutation-strings which lower the potential (and are accepted). Note that the total increase in distance from the optimum in the bits 2 to n has to be offset by the gain in bit 1, which is at most $k/n + \sqrt{6k\ln(nk)}/n$. Thus, the maximum loss in potential is $4(k/n + \sqrt{6k\ln(nk)}/n)$.

Fix any $z \in Z_0$. Then, in z, at most $n/2$ positions of the $99n/100$ large positions are moving towards the target (otherwise there is no loss in potential). Thus, in z, at least $49n/100$ large positions move away from the target. We let $N(z)$ be the set of all those elements from Z which can be obtained from z by orienting exactly 3 large positions, which orient away from the target in z, towards the target. Using an element of $N(z)$ for mutation leads to an increase in potential of at least

$$\begin{aligned} &9(k/n - \sqrt{6k\ln(nk)}/n) - 4(k/n + \sqrt{6k\ln(nk)}/n) \\ &= 5k/n - 13\sqrt{6k\ln(nk)}/n \geq 0. \end{aligned}$$

For each $z \in Z_0$, there are at least $(49n/100) \cdot (49n/100 - 1) \cdot (49n/100 - 2)/6$ many elements in $N(z)$. Let $Z_1 = \bigcup_{z \in Z_0} N(z)$. We consider an element $y \in Z_1$ and estimate the number of $z \in Z_0$ such that $y \in N(z)$. In y, there are at most $n/2 + 3$ large positions oriented towards the target (otherwise, any originating z would not decrease the potential and thus not be in Z_0). Hence, there are at most $(n/2+3) \cdot (n/2+2) \cdot (n/2+1)/6$ many $z \in Z_0$ such that $y \in N(z)$. This shows that, in this case, the potential gain is at most 0: Only elements from Z_0 decrease the potential, by $4k/n(1 + o(1))$. Each $z \in Z_0$ has $(49/(100n))^3(1 \pm o(1))$ many other elements that increases the potential by $5k/n(1 + o(1))$; these other elements are shared between at most $(50/(100n))^3(1 \pm o(1))$ many elements from Z_0. Thus, the weighted share is

$$\frac{(49/(100n))^3(1 \pm o(1))}{(50/(100n))^3(1 \pm o(1))} > 0.9.$$

Since $4 < 5 \cdot 0.9$, we get an average decrease of the potential of at most 0 in this case.

Overall, the potential decreases by $O(\max(n^3, r/2^{-cn}))$. Thus, an application of the additive drift theorem (Theorem 3) finishes the proof. □