Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints[☆]

Tobias Friedrich[a,b], Timo Kötzing[a], J. A. Gregor Lagodzinski[a,*], Frank Neumann[b], Martin Schirneck[a]

[a] *Hasso Plattner Institute, University of Potsdam, Potsdam, Germany*
[b] *School of Computer Science, The University of Adelaide, Adelaide, Australia*

Abstract

Linear functions have gained great attention in the run time analysis of evolutionary computation methods. The corresponding investigations have provided many effective tools for analyzing more complex problems. So far, the runtime analysis of evolutionary algorithms has mainly focused on unconstrained problems, but problems occurring in applications frequently involve constraints. Therefore, there is a strong need to extend the methods for analyzing unconstrained problems to a setting involving constraints.

In this paper, we consider the behavior of the classical (1+1) evolutionary algorithm on linear functions under linear constraint. We show tight bounds in the case where the constraint is given by the OneMax function and the objective function is given by either the OneMax or the BinVal function. For the general case we present upper and lower bounds.

Keywords: run time analysis, evolutionary algorithm, knapsack, constraints
2010 MSC: 68-06, 68Q25, 68W20, 68W40

1. Introduction

Evolutionary algorithms have been used in a wide range of application domains such as water distribution network [37, 40], renewable energy [30], supply chain

[☆] A conference version of this article was presented at FOGA 2017 [13].
[*] Corresponding author
Email address: gregor.lagodzinski@hpi.de (J. A. Gregor Lagodzinski)

management [29], and software engineering [15, 26]. Their easy application and adaptation to a wide range of engineering problems qualifies them for research even without a deep algorithmic background.

Although evolutionary computation is very popular in a large variety of application domains, the theoretical understanding lags behind its practical success. Over the last 20 years a lot of progress in understanding evolutionary computing techniques has been achieved by studying the run time behavior of evolutionary algorithms that are simpler than the ones used in practice, but still capture the main aspects of the algorithms [1, 19, 33].

At the heart of these investigations have been studies of the behavior of the (1+1) evolutionary algorithm ((1+1) EA) on the class of linear pseudo-Boolean functions [9]. Initially, they considered OneMax [31] as the simplest linear function, and later generalized to the whole class of linear functions. It has been shown in [9] that the (1+1) EA optimizes them in expected time $\Theta(n \log n)$. Further works gave simpler proofs, and a more precise analysis revealed the leading constants and lower order terms hidden in the Θ-notation [6, 18, 39]. Linear functions have been explored also in dynamic [24] and stochastic settings [8, 14]. The insights gained this way provided strong tools that facilitated the analysis of other evolutionary computing techniques beyond simple EAs, such as particle swarm optimization [38], ant colony optimization [25], and estimation of distribution algorithms [5]. Most recently, generational genetic algorithms [3] and steady-stage genetic algorithms [4] have been added.

All the mentioned results on the analysis of evolutionary computation techniques for linear pseudo-Boolean functions are considering the problem without any constraints. However, problems occurring in the real-world are often constrained as resources are limited. It is quite surprising that the impact of adding a constraint to linear pseudo-Boolean functions has so far been mainly ignored in the literature. Investigating such problems allows us to build on existing techniques and extend them to the often more realistic setting incorporating constraints. From a technical perspective, this might point out difficulties where current techniques are not sufficient for the analysis of the constraint setting and

therefore set challenges in developing new appropriate methods.

Maximizing a linear function under a linear constraint means, in the most general case, that the search space is split by a hyperplane and only the points in one of the half spaces are allowed to serve as solutions. In the Boolean domain, this is equivalent to the well-known NP-hard knapsack problem. Beyond the worst case, the knapsack problem has been well-studied from an average case and smooth complexity perspective and it has been found that this problem can be solved in (expected) polynomial time for a wide range of these settings [2, 36].

Regarding evolutionary computation, it has been shown that the expected optimization time of the (1+1) EA on a specific deceptive knapsack instance is exponential [41]. To identify tractable instances, we investigate several subclasses of linear functions under linear constraints; most notably, when the constraint is given by OneMax, merely restricting the number of 1-bits in the string. We call this a uniform constraint. The goal of our investigations is to gain an understanding on the working principles of the (1+1) EA for these subclasses. The reader should note that all instances under investigation can be solved to optimality in polynomial time by deterministic (greedy) algorithms.

Our findings are summarized in Table 1. We start our study by considering the OneMax objective together with a uniform constraint of B (only strings with at most B 1-bits are feasible) and show that the (1+1) EA is able to find an optimal solution efficiently. Namely, it solves the problem in time $O(n \log n)$, but depending on B potentially much faster. Note that OneMax with uniform constraints has many global optima: any bit string with B 1-bits is optimal. We modify OneMax by increasing the weight of B bit positions to $1 + \varepsilon$, for a very small value of ε. This ensures that there is only one global optimum. We show that this function requires at most a quadratic number of fitness evaluations in expectation to be optimized. With a similar technique we show a run time bound of $O(n^2)$ for BinVal under any uniform constraint B. BinVal has only one global optimum, but the values of the different bits is now much more important. We sidestep the occurring issues with a careful drift analysis and show that the drift strongly depends on the number of non-optimal bits *outside*

the B positions of highest weight.

Investigating more general functions under uniform constraint, we show that a general upper bound of $O(n^3)$ holds for all linear objective functions and possible choices of B. Furthermore, we show that there exist linear functions and values of B such that an optimization time of $\Omega(n^2)$ cannot be avoided. We conjecture a general upper bound of $O(n^2)$ for all linear functions, independently of B. However, for now we content ourselves with showing this bound for BinVal and the $(1+\varepsilon)$-test function mentioned above.

We proceed in Section 2 by introducing the algorithm and the class of constrained optimization problems that is subject to our investigations. We consider uniform constraints in Section 3 and linear constraints in Section 4. Finally, we conclude in Section 5.

Constraint	Objective	Expected Optimization Time	
uniform	feasibility	$O(n \log(n/B))$	Lemma 7
	OneMax	$\Theta(\sqrt{n} + \lvert B - n/2 \rvert \log(n/B_{\min}))$	Theorem 9
	linear	$\Omega(n^2)$	Theorem 10
		$O(n^2 \log(B\, w_{\max}))$	Theorem 11
	$\sum_{i=1}^{B}(1+\varepsilon)x_i + \sum_{i=B+1}^{n} x_i$	$O(n^2)$	Theorem 12
	BinVal	$O(n^2)$	Theorem 13
linear	OneMax	$2^{\Omega(n)}$	Theorem 14

Table 1: Overview of Results. The expected optimization times of the (1+1) EA on linear functions of length-n bit strings under uniform or linear constraint B. The optimization time on OneMax under uniform constraint is never larger than in the unconstrained case and there are ranges of B in which it is significantly smaller. On the contrary, there is a (general) linear function and a uniform constraint B such that the optimization time is $\Omega(n^2)$. There is also a linear constraint such that the (1+1) EA needs exponential time even on OneMax as the objective.

For uniform constraints, B is an integer between 1 and n. $B_{\min} = \min(B, n-B)$, $0 < \varepsilon < 1/B$ is a positive real number, and $w_{\max}$ denotes the largest weight of the linear objective function. In the extreme case of $B = n$, $O(n \log(n/B))$ is to be read as $O(n)$. If $B_{\min} = 0$, $O(\log(n/B_{\min}))$ stands for $O(\log n)$.

2. Problems and Techniques

In this article we give bounds on the expected run time of the (1+1) evolutionary algorithm on pseudo-Boolean optimization problems whose collection of feasible solutions is restricted by a linear constraint. For the analysis we make extensive use of various tools commonly known as *drift analysis*.

2.1. Constrained Optimization Problems

We consider as search space the set $\{0,1\}^n$ of all bit strings $x = x_1 x_2 \dots x_n$ of fixed length n and mainly examine the class of linear functions

$$f(x) = \sum_{i=1}^{n} w_i x_i.$$

The weights w_i are positive reals, w.l.o.g. not smaller than 1; scaling them by $\min_{1 \le i \le n} w_i$ does not alter the acceptance behavior of the (1+1) EA (see Algorithm 1 below). We denote by $w_{\max} = \max_i w_i$ the maximal weight. The *feasible region* of the search space is given by a linear constraint

$$b(x) = \sum_{i=1}^{n} b_i x_i \le B$$

where the weights b_i are positive reals and B is a positive upper bound. We say the function f is under *uniform constraint* if all b_i are equal to 1; otherwise, we say that it is under *linear constraint*. An *optimal solution* is then a feasible search point of maximum f-value.

In order to analyze pseudo-Boolean functions, we fix some notation regarding bit strings. The number of 1-bits in x is denoted $|x|_1 = \sum_{i=1}^{n} x_i$, we sometimes also use the term *Hamming weight*; $|x|_0 = n - |x|_1$ is the number of 0-bits.

Algorithm 1: (1+1) EA

1 Choose $x \in \{0,1\}^n$ uniformly at random;
2 **while** *stopping criterion not met* **do**
3 $\quad y \leftarrow$ flip each bit of x independently with probability $1/n$;
4 $\quad$ **if** $z(y) \geq z(x)$ **then** $x \leftarrow y$;

In order to optimize f under the constraint $b(x) \leq B$ we employ the (1+1) EA, see Algorithm 1. Here, x denotes the best search point found so far. We use symbol x' for the (possibly mutated) offspring *after* selection; $x^{(0)}$ denotes the initial bit string drawn in Line 1. The constrained aspect of the optimization is handled by a penalty approach. Let $\mathbb{1}_{b>B}$ denote the characteristic function of the infeasible region. We define an auxiliary function z as

$$z(x) = f(x) - \mathbb{1}_{b>B}(x) \cdot (n\, w_{\max} + 1)(b(x) - B + 1).$$

This definition serves two purposes. First, $z(x)$ is negative iff x is infeasible. Second, while still in the infeasible region, the z-value increases when the extend of the constraint violation $b(x) - B$ is reduced. This way the search is guided towards the feasible region. The (1+1) EA accepts an offspring iff its z-value is not worse than that of its parent. In particular, Algorithm 1 will *never* adopt an infeasible solution in Line 4 after sampling the first feasible bit string.

The introduced fitness function uses a classical penalty approach where solutions are penalized with respect to their constraint violation. It should be noted that giving each infeasible solution the same large penalty would lead to large plateaus that are hard to escape from. In order to illustrate this, consider the case where B is small, i.e.‘ $B = 1$. This implies that a solution is only feasible if exactly one bit is set to one and all solutions with more than one bit set to 1 are infeasible. The set of infeasible solutions forms a large plateau and the set of feasible solutions almost forms a needle in the haystack. The classical needle in the haystack function called *Needle* differs from our setting by having exactly one optimal search with all other search points attaining the same non-optimal

fitness value. It is well known that the (1+1) EA has an expected optimization time of $\Omega(2^n)$ on *Needle* as the black box complexity of *Needle* is $\Omega(2^n)$ in the unrestricted black-model [7, 10]. We refer the reader to [20] for a more detailed discussions on runtime analysis of evolutionary algorithms on plateau functions.

We study the number of iterations the (1+1) EA needs until it samples an optimal solution for the first time. This is called the *optimization time* of the algorithm; we usually denote this random variable as T. The expected value of this variable $\mathrm{E}[T]$ is called the *expected optimization time*. We would like to point out that, unless explicitly stated otherwise, all asymptotic bounds are in terms of both n and B simultaneously. More formally, we compute the univariate asymptotics of the expected optimization time with respect to n for any positive, non-decreasing function $B = B(n)$.

During the analysis in the sections below we frequently bound an expected value by some conditional expectation. The justification is the following easy observation from the law of total expectation.

Lemma 1. *Suppose X is a discrete random variable taking values in $\mathbb{R}_0^+$ and $\mathcal{E}$ is an event in the same probability space such that $0 < \Pr[\mathcal{E}] < 1$. Then, $\mathrm{E}[X] \geq \mathrm{E}[X \mid \mathcal{E}] \Pr[\mathcal{E}]$. If additionally $X > 0$ implies $\mathcal{E}$, equality holds.*

Proof. The conditional expectation $\mathrm{E}[X \mid \neg\mathcal{E}]$ exists and is non-negative. Thus,

$$\mathrm{E}[X] = \mathrm{E}[X \mid \mathcal{E}] \Pr[\mathcal{E}] + \mathrm{E}[X \mid \neg\mathcal{E}] \Pr[\neg\mathcal{E}] \geq \mathrm{E}[X \mid \mathcal{E}] \Pr[\mathcal{E}].$$

If the second condition is met, then $\mathrm{E}[X \mid \neg\mathcal{E}] = 0$. □

2.2. Drift Analysis

In order to give bounds on the optimization time we apply drift analysis. For a general overview of the applications of drift analysis to evolutionary algorithms, including historical remarks, we direct the reader to [28]. The main idea of this approach is to define a *potential function* that maps the inner state of the algorithm to the real axis, allowing us to evaluate the expected progress between

consecutive rounds. Usually the potential is chosen such that its minimization corresponds to maximizing the fitness of the current solution. The potential reaching its minimal value is then equivalent to the bit string being optimal. Depending on the type of expected potential decrease, the *drift*, we apply one of several drift theorems with the following being arguably the most general.

Theorem 2 (Variable Drift Theorem [11, 21, 35])**.** *Let $(X^{(t)})_{t\geq 0}$ be sequence of random variables ranging over $S \cup \{0\}$, where $S = [s_{\min}, s_{\max}] \subsetneq \mathbb{R}^+$ is a positive interval. Furthermore, let T be the random variable denoting the first point in time $t \geq 0$ for which $X^{(t)} = 0$. Suppose that there exists a monotonically increasing function $h\colon [s_{\min}, s_{\max}] \to \mathbb{R}^+$ such that $1/h$ is integrable, and for all $t < T$ and $s \in S$,*

$$\mathrm{E}\Big[X^{(t)} - X^{(t+1)} \;\Big|\; X^{(t)} = s\Big] \geq h(s).$$

Then, for all $s_0 \in S$,

$$\mathrm{E}\Big[T \;\Big|\; X^{(0)} = s_0\Big] \leq \frac{s_{\min}}{h(s_{\min})} + \int_{s_{\min}}^{s_0} \frac{1}{h(s)}\,\mathrm{d}s.$$

If the drift function h satisfies additional conditions, tighter upper bounds as well as lower bounds can be asserted.

Theorem 3 (Additive Drift Theorem [17])**.** *Let $(X^{(t)})_{t\geq 0}$, S, $s_{\min}$, and T be as above. Suppose that there exists a positive real number $\delta > 0$ such that,*

$$\mathrm{E}[X^{(t)} - X^{(t+1)} \mid T > t] \geq \delta.$$

Then, for all $s_0 \in S$,

$$\mathrm{E}[T \mid X^{(0)} = s_0] \leq \frac{s_0 - s_{\min}}{\delta}.$$

If $\mathrm{E}[X^{(t)} - X^{(t+1)} \mid T > t] \leq \delta$, then $\mathrm{E}[T \mid X^{(0)} = s_0] \geq (s_0 - s_{\min})/\delta$.

Theorem 4 (Tail bound to the Additive Drift Theorem for lower bounds [23])**.** *In addition to the prerequisites of the second clause of Theorem 3, assume that*

the random variables $(X^{(t)})_{t\geq 0}$ each have finite expectation. Suppose that there exists a real number $\gamma > 0$ such that for all $t < T$,

$$|X^{(t)} - X^{(t+1)}| < \gamma.$$

Then, for all $s_0 \in S$ and every $r \geq 2(s_0 - s_{\min})/\delta$,

$$\Pr[T \geq r \mid X^{(0)} = s_0] \leq \exp\left(-\frac{r\delta^2}{8\gamma^2}\right).$$

Theorem 5 (Multiplicative Drift Theorem for upper bounds [6])**.** *Let $(X^{(t)})_{t\geq 0}$, S, $s_{\min}$, and T be as in Theorem 2. Suppose there exists a real number $\delta > 0$ such that for all $s \in S$ and $t < T$,*

$$\mathrm{E}[X^{(t)} - X^{(t+1)} \mid X^{(t)} = s] \geq \delta s.$$

Then, for all $s_0 \in S$,

$$\mathrm{E}[T \mid X^{(0)} = s_0] \leq \frac{\ln(s_0) - \ln(s_{\min}) + 1}{\delta}.$$

Theorem 6 (Multiplicative Drift Theorem for lower bounds [39])**.** *The random process $(X^{(t)})_{t\geq 0}$ now only takes values in S, excluding 0. Additionally, we require $s_{\min} = 1$ and $X^{(t+1)} \leq X^{(t)}$ for all t. The target value $s^* > 0$ is now arbitrary and T is the stopping time for the event $X^{(t)} \leq s^*$. Suppose there exist real numbers $1 \geq \delta, \beta > 0$ such that for all $s \in S$ and $t < T$,*

(i) $\mathrm{E}[X^{(t)} - X^{(t+1)} \mid X^{(t)} = s] \leq \delta s$;

(ii) $\Pr[X^{(t)} - X^{(t+1)} \geq \beta s \mid X^{(t)} = s \neq 1] \leq \beta\delta/\ln s$.

Then, for all $s_0 \in S$,

$$\mathrm{E}[T \mid X^{(0)} = s_0] \geq \frac{1-\beta}{1+\beta} \cdot \frac{\ln(s_0) - \ln(s^*)}{\delta}.$$

3. Uniform Constraint

We start with investigating uniform constraints, i.e., $b_i = 1$ for all $1 \leq i \leq n$. This effectively restricts the total number of 1-bits in a feasible solution. Hence, we assume the weight bound B to be an integer between 1 and n.

In the following lemma we derive a general bound on the time the (1+1) EA needs to sample a feasible solution on *any* pseudo-Boolean function under uniform constraint. This will ease the later analysis. We reduce the problem at hand to the well-known case of the (1+1) EA on the OneMax function, i.e.,

$$\textsc{OneMax}(x) = \sum_{i=1}^{n} x_i.$$

We would like to point out that in this article run time estimates of the form $\mathrm{O}(n \log(n/B))$ are to be read as $\mathrm{O}(n)$ if $B = n$.

Lemma 7. *Consider the* (1+1) EA *optimizing a linear function under uniform constraint B. Then, the expected number of iterations until a feasible solution is sampled for the first time is* $\mathrm{O}(n \log(n/B))$.

Proof. If the initial bit string is feasible, we are done. This includes the special case of $B = n$. In the infeasible range the (1+1) EA always prefers bit strings with fewer 1-bits. Hence, the optimization process is the same as that of the (1+1) EA on an unconstrained OneMax function (considered as a minimization problem), see e.g. [6, 39]. A mutant is adopted as offspring if and only if its Hamming weight is not larger than that of its parent. In particular, flipping a single 1-bit and nothing else improves on the potential and gives the following lower bound on the expected drift,

$$\mathrm{E}\big[\,|x|_1 - |x'|_1 \;\big|\; |x|_1 > B\big] \geq \frac{|x|_1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{|x|_1}{en}.$$

This bound is a multiplicative of the current potential. The maximum potential of any infeasible solution is n. Invoking the Multiplicative Drift Theorem for upper bounds (Theorem 5) with $\delta = 1/en$, $s_0 = n$, and $s_{\min} = B$ gives an

expected waiting time of

$$\mathrm{E}[T] \leq en\left(\ln\left(\frac{n}{B}\right)+1\right)$$

until the number of 1-bits is reduced below the cardinality constraint B. □

Above lemma can easily be extended to arbitrary non-linear (non-negative) pseudo-Boolean functions. Since for those functions the notion of $w_{\max}$ is undefined, the auxiliary function z of Algorithm 1 must be set such that infeasible solutions correspond to negative z-values.

As we will see in the next section, the bound established in Lemma 7 is not always tight. The importance of this result lies elsewhere. It will often allow us to assume that the optimization starts with a feasible solution without affecting the asymptotic run time.

3.1. OneMax

In the infeasible region of the search space any pseudo-Boolean function under uniform constraint behaves like a OneMax problem. To complement this, we now examine the optimization process of the (1+1) EA on OneMax as the objective function. The run time turns out to be heavily dependent on the relative size of the cardinality bound B in terms of the length n of the bit string. The next theorem shows that the time needed is never worse than in the unconstrained case. Moreover, OneMax can be maximized even in sub-linear time if B is close to $n/2$.

The analysis benefits extensively from symmetries inherent to the underlying random process. OneMax is invariant under permutations and the standard bit mutation operator of the (1+1) EA is indifferent towards the position and the value of the bits. Additionally, the number of 1-bits and the number of 0-bits in the initial solution are identically distributed and this distribution is symmetric around its mean value $n/2$.

We use the following lemma by Oliveto and Witt to bound the probability that a binomial variable deviates moderately from its expectation [34].

Lemma 8 (Lemma 6 in [34]). *Let X be the sum of n i.i.d. Bernoulli trials with success probability p and ℓ the minimum of pn and $(1-p)n$. Then,*

$$\Pr\left[X \geq \mathrm{E}[X] + \sqrt{\frac{\ell}{2}}\right] = \Omega(1).$$

Theorem 9. *Let $B_{\min}$ denote the minimum of B and $n-B$. The expected optimization time of the* (1+1) EA *on* OneMax *under uniform constraint B is*

$$\Theta\left(\sqrt{n} + \left|B - \frac{n}{2}\right| \log\left(\frac{n}{B_{\min}}\right)\right).$$

Similar to Lemma 7, the bound is to be read as $\Theta(n \log n)$ if $B = n$.

Proof. In this proof we identify the three main quantities that affect the run time of the (1+1) EA on constrained OneMax: the expected drift, the distance between the initial number of 1-bits and the cardinality bound B, and the distance from B to the central value $n/2$. Intuitively, the distance of the initial solution to B marks the ground we have to cover until we reach an optimal solution, and the drift is the speed we travel. The difference $|B-n/2|$ partitions the range of all possible values of B into regions corresponding to different asymptotic run times. In full detail, we prove the following.

$$\mathrm{E}[T] = \begin{cases} \Theta(\sqrt{n}), & \text{if } \left|B - \frac{n}{2}\right| < \sqrt{n}; \\ \Theta\left(\left|B - \frac{n}{2}\right|\right), & \text{if } \sqrt{n} \leq \left|B - \frac{n}{2}\right| < \frac{7}{16}\,n; \\ \Theta\left(n \log\left(\frac{n}{B_{\min}}\right)\right), & \text{otherwise.} \end{cases}$$

Observe that this indeed gives the claimed bound.

The first part of this proof is presented as a series of claims giving bounds on the expected drift and distance. While the drift can be inferred from standard arguments (Claim 1 below), the analysis of the influence of the two distance measures is more involved. The initial bit string has an expected Hamming weight of $n/2$. We show that the initial solution also has an expected lack/surplus of at least $\sqrt{n}$ 1-bits compared to any optimal solution (Claims 2 & 4). This

gap grows linearly when B moves further away from the center (Claim 3). The second part of the proof then derives bounds on the optimization time.

First, if the cardinality bound is set to the extreme value of $B = n$, the problem is equal to unconstrained OneMax (as a maximization problem). Hence, the well-known $\Theta(n \log n)$ bound carries over to this setting, see [9]. We assume $B < n$ in the following. Note that the upper bound holds independently of the value of B. We find a feasible solution in time $O(n \log n)$ (Lemma 7) and improve it until B bits are set to 1. This can be done in an additional phase of $O(n \log n)$ rounds by a Coupon Collector's argument. We now show that the optimization succeeds much faster if B is close to $n/2$.

For the drift analysis it is convenient to use either $|x|_1 = \textsc{OneMax}(x)$ itself or $|x|_0 = n - |x|_1$ as the potential function, depending on whether the current search string is feasible or not. Our first claim bounds the expected drift with respect to this potential. The results are well-known, we only state them here for completeness. A detailed discussion can be found in [6] and [39].

Claim 1. *While x is feasible, the expected drift is bounded by*

$$\frac{|x|_0}{en} \leq \mathrm{E}\big[\, |x|_0 - |x'|_0 \;\big|\; |x|_1 \leq B \big] \leq \frac{|x|_0}{n}.$$

Similarly, if x is infeasible, $\mathrm{E}[\, |x|_1 - |x'|_1 \mid |x|_1 > B] = \Theta(|x|_1/n)$. ◁

Next, we give estimates on the second of the above quantities, the distance from the initial number of 1-bits $|x^{(0)}|_1$ to the constraint B. Let

$$d_B(x^{(0)}) = |B - |x^{(0)}|_1|$$

denote this random variable. Furthermore, we use

$$B_{\mathrm{cen}} = |B - n/2|$$

for the absolute difference between B and the central value, the third quantity.

Claim 2. *The expected distance of the initial solution is bounded below by*

$$\mathrm{E}\Big[d_B(x^{(0)})\Big] = \Omega(\sqrt{n}).$$

Assume $B \leq n/2$ for the moment. Then, $|x^{(0)}|_1 \geq (n+\sqrt{n})/2$ is sufficient for $d_B(x^{(0)}) \geq \sqrt{n}/2$. The random weight $|x^{(0)}|_1$ is the sum of n i.i.d. equiprobable Bernoulli trials. An application of Lemma 8 with $\ell = n/2$ now gives

$$\Pr\left[|x^{(0)}|_1 \geq \frac{n}{2} + \frac{\sqrt{n}}{2}\right] = \Omega(1).$$

If $B > n/2$, we apply the same argument to the random variable $|x^{(0)}|_0$ (having the same distribution) and the event $|x^{(0)}|_0 \geq (n+\sqrt{n})/2$. By Lemma 1,

$$\mathrm{E}\Big[d_B(x^{(0)})\Big] \geq \Omega(1) \cdot \frac{\sqrt{n}}{2} = \Omega(\sqrt{n}). \qquad \lhd$$

For the next claim, recall that B_{cen} stands for $|B - n/2|$.

Claim 3. *The expected distance of the initial solution is bounded below by*

$$\mathrm{E}\Big[d_B(x^{(0)})\Big] \geq \frac{B_{\mathrm{cen}}}{2}.$$

Variable $|x^{(0)}|_1$ has $n/2$ also as its median. Hence, if $B \leq n/2$, with probability $\Pr[|x^{(0)}|_1 \geq n/2] \geq 1/2$ the distance is at least B_{cen}. The same holds if $B > n/2$, as $\Pr[|x^{(0)}|_1 \leq n/2] \geq 1/2$. The lower bound now follows from Lemma 1. $\lhd$

Claim 4. *The expected distance is bounded above by*

$$\mathrm{E}\Big[d_B(x^{(0)})\Big] \leq B_{\mathrm{cen}} + \frac{e}{2\pi}\sqrt{n}.$$

The triangle inequality yields $d_B(x^{(0)}) \leq B_{\mathrm{cen}} + \Big||x^{(0)}|_1 - n/2\Big|$. From the monotonicity and linearity of expectations we deduce

$$\mathrm{E}\Big[d_B(x^{(0)})\Big] \leq B_{\mathrm{cen}} + \mathrm{E}\Big[\Big||x^{(0)}|_1 - \frac{n}{2}\Big|\Big].$$

The latter expected value is known as the *mean deviation* of a binomially distributed random variable. For the special case of success probability $1/2$, it equals $\binom{n}{\lceil n/2 \rceil} \lceil n/2 \rceil 2^{-n}$, cf. e.g. [12]. Applying Stirling's approximation gives

$$\begin{aligned}\mathrm{E}\Big[\Big||x^{(0)}|_1 - \frac{n}{2}\Big|\Big] &= \frac{n}{2^{n+1}}\binom{n}{\frac{n}{2}} = \frac{n}{2^{n+1}} \cdot \frac{n!}{\left(\frac{n}{2}!\right)^2} \\ &\leq \frac{n}{2^{n+1}} \cdot \frac{e\sqrt{n}\left(\frac{n}{e}\right)^n}{\left(\sqrt{2\pi}\sqrt{\frac{n}{2}}\left(\frac{n}{2e}\right)^{\frac{n}{2}}\right)^2} = \frac{e}{2\pi}\sqrt{n}. \qquad \lhd\end{aligned}$$

Claims 2, 3 and 4 together show that if $B_{\mathrm{cen}} < \sqrt{n}$, the initial solution has distance $\mathrm{E}[d_B(x^{(0)})] = \Theta(\sqrt{n})$; otherwise, it is $\Theta(B_{\mathrm{cen}})$. The next claim states a useful technical property of the first feasible solution sampled during the optimization, as described in Lemma 7.

Claim 5. *If the optimization started in the infeasible region, the first feasible solution found by the* (1+1) EA *has Hamming weight at least* $B - \ln n$ *with probability superpolynomially close to* 1.

Consider the iteration in which the optimization process enters the feasible region. In order to jump from more than B bits set to 1 to less than $B - \ln n$, at least $\ln n$ bits must flip at once. We get the following bound on the probability,

$$\begin{aligned}\Pr\Big[|x'|_1 < B - \ln n \,\Big|\, |x|_1 > B\Big] &\leq \sum_{i=\ln n}^{n} \binom{n}{i}\frac{1}{n^i} \leq n\binom{n}{\ln n}\frac{1}{n^{\ln n}} \\ &\leq n\left(\frac{e}{\ln n}\right)^{\ln n} = \frac{1}{n^{\ln\ln n - 2}}. \qquad \lhd\end{aligned}$$

In the remainder of this proof we infer bounds on the expected optimization time from the claims above. We commence with proving an universal lower bound. Recall that T is the random variable denoting the number of rounds the (1+1) EA needs to sample an optimal solution for the first time. We consider its expected value $\mathrm{E}[T \mid d_B(x^{(0)})]$ conditional on the random distance of the initial solution to the bound B. Observe that this makes $\mathrm{E}[T \mid d_B(x^{(0)})]$ *itself* a random variable, and we have $\mathrm{E}[T] = \mathrm{E}[\,\mathrm{E}[T \mid d_B(x^{(0)})]\,]$. Claim 1 states that the drift during the whole optimization is at most 1, regardless of feasibility.

The Additive Drift Theorem (for lower bounds; Theorem 3) thus implies

$$\mathrm{E}\Big[T \;\Big|\; d_B(x^{(0)})\Big] \geq d_B(x^{(0)}).$$

We bound the expectation of the derived variable and, in turn, the expected optimization time with the help of Claim 2,

$$\mathrm{E}[T] = \mathrm{E}\Big[\mathrm{E}\Big[T \;\Big|\; d_B(x^{(0)})\Big]\Big] \geq \mathrm{E}\Big[d_B(x^{(0)})\Big] = \Omega(\sqrt{n}).$$

Note that this bound holds for any uniform constraint B.

According to Chernoff bounds, the initial solution has at least $n/3$ and at most $2n/3$ bits set to 1 with probability $1 - 2^{-\Omega(n)}$. We have already seen an upper bound of $\mathrm{O}(n \log n)$ on the expected optimization time, independently of B. Hence, conditioning on the initial solution $x^{(0)}$ containing a linear number of both 1-bits and 0-bits affects the expected run time only by a sub-constant number of steps. We omit this condition in the notation below.

Assume $B_{\mathrm{cen}} < \sqrt{n}$. While the currently best search point is infeasible, its Hamming weight cannot increase. However, undershooting the target cardinality bound B by more than $\ln n$ is also unlikely (Claim 5). Conversely, the weight cannot decrease while the search point is feasible. In summary, we can safely assume that the maintained solution x observes $n/3 \leq |x|_1 \leq 2n/3$ during the whole optimization. By Claim 1, the expected drift is at least $1/3e$. The Additive Drift Theorem (the clause on upper bounds of Theorem 3) now yields

$$\mathrm{E}\Big[T \;\Big|\; d_B(x^{(0)})\Big] \leq 3e \cdot d_B(x^{(0)}).$$

Applying Claim 4 and the assumption $B_{\mathrm{cen}} < \sqrt{n}$ gives

$$\mathrm{E}[T] = \mathrm{E}\Big[\mathrm{E}\Big[T \;\Big|\; d_B(x^{(0)})\Big]\Big] \leq 3e\left(B_{\mathrm{cen}} + \frac{e}{2\pi}\sqrt{n}\right) = \mathrm{O}(\sqrt{n}).$$

If B_{cen} is between $\sqrt{n}$ and $7n/16$, the expected optimization time is in $\Theta(B_{\mathrm{cen}})$. The argument is the same as above, but utilizes Claim 3 in place

of Claim 2. The main observation is that the drift is still constant in this range of B. Note that now $B_{\text{cen}} + \sqrt{n} = \mathrm{O}(B_{\text{cen}})$.

Finally, we turn the discussion to the case in which the distance B_{cen} is larger than $7n/16$. The main difference is that the expected drift towards the can now become sub-constant during the optimization. If the bound B is close to 1, the (1+1) EA must remove almost all 1-bits to even find a feasible solution, let alone an optimal one. Conversely, if B is close to n, the number of 0-bits are almost eliminated. Both events result in a very small drift towards any optimum, cp. Claim 1. We use a multiplicative drift argument to handle this.

First, assume $B \geq 15n/16$, that is, $B_{\min} = \min(B, n - B) \leq n/16$. The initial assumption $B < n$ ensures that $B_{\min}$ is positive. By the same Chernoff argument as above, the initial solution is feasible (with probability exponentially close to 1). We use the number of 0-bits as the potential. In order to optimize OneMax, the (1+1) EA has to reduce the potential from at most n down to $B_{\min}$. The expected drift is at least $|x|_0/en$, the Multiplicative Drift Theorem for upper bounds (Theorem 5) thus yields

$$\mathrm{E}[T] \leq en \left(\ln\left(\frac{n}{B_{\min}}\right) + 1 \right) = \mathrm{O}\left(n \log\left(\frac{n}{B_{\min}}\right)\right).$$

To prove the lower bound, we want to apply Theorem 6. We have to check its various prerequisites. Again by Chernoff bounds, we can assume that the initial solution is not only feasible but has at least $B_{\min} + n/9$ bits set to 0. This implies that the number of 0-bits cannot increase during the optimization. We only measure the time until it falls below $B_{\min} + \ln n$. Suppose the current potential is $|x|_0 = s$, and define $\delta = 1/n$, $\beta = 1/2$. Then, the expected drift is at most $s/n = \delta s$ (Claim 1). Furthermore, large jumps are unlikely. More formally, in order to have a progress of at least $s/2 = \beta s$, between $s/2$ and s bits must flip simultaneously. The probability for such a mutation is at most

$$\sum_{i=s/2}^{s} \binom{n}{i} \frac{1}{n^i} \leq \sum_{i=s/2}^{s} \left(\frac{e}{i}\right)^i \leq \frac{s}{2} \left(\frac{2e}{s}\right)^{\frac{s}{2}}.$$

Since $n \geq s \geq B_{\min} + \ln n > \ln n$, we obtain

$$\Pr\left[|x|_0 - |x'|_0 \geq \frac{s}{2} \;\middle|\; |x|_0 = s\right] < \frac{\ln n}{2}\left(\frac{2e}{\ln n}\right)^{\frac{\ln n}{2}} \leq \frac{1}{2n\ln n} \leq \frac{\beta\delta}{\ln s}.$$

This results in the following lower bound on the expected optimization time,

$$\mathrm{E}[T] \geq \frac{1-\beta}{1+\beta} \cdot \frac{1}{\delta} \cdot \ln\left(\frac{B_{\min} + n/9}{B_{\min} + \ln n}\right) \geq \frac{n}{3}\,\ln\left(\frac{n/9}{B_{\min} + \ln n}\right)$$

Observe that $B_{\min} \leq n/16$ implies $B_{\min} + \ln n \leq \sqrt{n\,B_{\min}/9}$, provided that n is large enough. From there, we arrive at $\ln(B_{\min} + \ln n) \leq (\ln(n/9) + \ln B_{\min})/2$. Substituting this into our estimate yields

$$\mathrm{E}[T] \geq \frac{n}{3}\left(\ln\left(\frac{n}{9}\right) - \ln(B_{\min} + \ln n)\right) \geq \frac{n}{6}\left(\ln\left(\frac{n}{9}\right) - \ln B_{\min}\right),$$

which is $\Omega(n \log(n/B_{\min}))$, matching the upper bound.

The proof of the run time bounds in case of $B \leq n/16$ is similar, but somehow simpler. Now $B_{\min}$ is equal to B. For the analysis we invert the roles of 0- and 1-bits. With probability exponentially close to 1 the initial solution is *infeasible* and has a linear surplus of *1-bits*. A reduction below $B_{\min} + \ln n$ is necessary to optimize the bit string. By the same series of arguments as above, this needs an expected number of $\Omega(n \log(n/B))$ iterations. In order to derive the upper bound, we argue that the search for a feasible solution, which takes time $\mathrm{O}(n \log(n/B))$ in expectation (Lemma 7), dominates the run time. Once the (1+1) EA enters the feasible region, we only need to collect $\ln n$ additional 1-bits with probability superpolynomially close to 1, as proven in Claim 5. Since $B \leq n/16$, the currently best solution x has potential $|x|_0 \geq 15n/16$. Thus, we are again in the realm of constant drift. Joining the two phases gives

$$\mathrm{E}[T] = \mathrm{O}\left(n \log\left(\frac{n}{B}\right)\right) + \mathrm{O}(\log n) = \mathrm{O}\left(n \log\left(\frac{n}{B_{\min}}\right)\right). \qquad \square$$

3.2. Linear Functions

We now move to the more general case of linear objective functions under uniform constraints. The weights $w_i \geq 1$ can be chosen arbitrarily, whereas the weights b_i of the constraint are still all equal to 1. For OneMax the introduction of a cardinality bound never worsened the run time of the (1+1) EA. Contrary to that, uniform constraints increase the optimization time on linear functions in general. There exists a family of linear functions and corresponding bounds B such that their optimization needs at least quadratic time. The reason is that the collection of additional 1-bits stalls once the Hamming weight of the current solution equals B. From there, progress is only possible by swapping a 1-bit to a position with larger weight. This requires a simultaneous flip of both bits.

Theorem 10. *There is a linear function f and a bound B such that the optimization time of the* (1+1) EA *on f under uniform constraint B is in $\Omega(n^2)$, not only in expectation but even with high probability.*[1]

Proof. Let $\varepsilon > 0$ be an arbitrary positive real number, possibly even dependent on n. We set the bound B to $3n/4$ and define the objective f as

$$f(x) = \sum_{i=1}^{B} (1+\varepsilon)\, x_i + \sum_{j=B+1}^{n} x_j.$$

The slight weight increase in the first B bits results in f having a *unique* global optimum at $x^* = 1^{3n/4}0^{n/4}$. The main idea of this proof is to show that, during the optimization, the (1+1) EA likely samples a point with constant Hamming distance d_H from the optimum and exactly B 1-bits. Then, the only way to reach the optimum is to exchange a 0 in the first $3n/4$ bit positions, we call this the *first block*, with a 1 in the last $n/4$ bits, the *second block*. This event has at least a quadratic waiting time.

First, we prove that the (1+1) EA, prior to finding the optimum x^*, with

[1] We use the term *with high probability* for a success probability of at least $1 - 1/n^c$ for some constant $c > 0$.

high probability either samples a search point with Hamming distance between 4 and 8 from x^* or runs for $\Omega(n^2)$ iterations regardless. In a second step, we show that, given a feasible solution with constant distance, Algorithm 1 first finds a *non-optimal* bit string with exactly B 1-bits. A union bound over the polynomially small error probabilities for these events then implies the theorem.

By Chernoff bounds the initial solution has no more than $2n/3$ 1-bits with probability exponentially close to 1. We thus assume a linear Hamming distance from x^*. In order to maximize function f the (1+1) EA must decrease this distance below any positive constant. We claim that the algorithm does not jump directly from an individual with distance greater than 8 to one with distance less than 4. To this end, let $d > 8$ be the number of wrongly set bits of the current search point x. We pessimistically assume that every mutation decreasing the distance is accepted. For this malicious mutation to occur, at least $d - 3$ bits must flip simultaneously.

$$\begin{aligned}\Pr\big[d_H(x', x^*) < 4 \;\big|\; d_H(x, x^*) = d\big] &\leq \sum_{i=0}^{3} \binom{d}{d-i} \frac{1}{n^{d-i}} \\ &\leq 4\binom{d}{3}\frac{1}{n^{d-3}} \leq \frac{d^3}{n^{d-3}} = \mathrm{O}\Big(\frac{1}{n^6}\Big).\end{aligned}$$

The last estimate is due to the observation that the upper bound is maximal for $d = 9$. Hence, there exists a constant $c > 0$ such that this jump does not occur in the first cn^2 iterations with probability at least $1 - 1/n^4$.

We now assume that Algorithm 1 just sampled a feasible solution x with $4 \leq d_H(x, x^*) \leq 8$ and continue the analysis from this point. If x has exactly B bits set to 1, the theorem follows immediately. The reason is as follows. Due to the Hamming distance, x can have at most 8 0-bits in the first block and at most 8 1-bits in the second block. Every mutation must flip at least one pair of these misplaced 0s and 1s simultaneously to improve on the fitness value. The probability for this to happen is at most $64/n^2$.

We are left with the case in which x has Hamming distance between 4 and 8, and strictly less than B 1-bits. Consider a run of the (1+1) EA for $\ln n$

steps. We want to show that during this phase the maintained solution collects B 1-bits in total but still does not reach x^*, with high probability. We employ drift analysis with the number of 1s as the potential. Observe that we have $|x|_1 \geq B - 8 = 3n/4 - 8$ due to the Hamming distance requirement. A standard argument provides a lower bound of $1/4e$ on the expected drift since x has more than $n/4$ bits set to 0 and flipping any of them is accepted as a fitness increase. Furthermore, no mutation increasing the number of 1s by more than 8 is accepted as this would violate the constraint. We apply the tail bound to the Additive Drift Theorem given in Theorem 4 with parameters $\delta = 1/4e$, $\gamma = 8$, and $r = \ln n$. This gives the following lower bound on the probability that the (1+1) EA removes the constant-size surplus of 0s within r rounds,

$$1 - \exp\left(-\frac{r\delta^2}{8\gamma^2}\right) = 1 - \exp(-\Omega(\log n)) = 1 - \frac{1}{n^{\Omega(1)}}.$$

We can safely assume that the Hamming distance to the optimum does not grow beyond 8 during these r iterations; otherwise, the same argument as above gives a quadratic lower bound on the run time. On the other hand, in order to reach x^* prematurely in this phase all $d \geq 4$ wrong bits must have to flip at least once. Finally, we have to prove that this does not happen with high probability. A specific bit position does not flip during r rounds with probability $(1 - 1/n)^r$. Hence, the d bits flip with probability $(1 - (1 - 1/n)^r)^d$. Using Bernoulli's inequality, the probability that the (1+1) EA does not sample the optimum during the r rounds is at least

$$1 - \left(1 - \left(1 - \frac{1}{n}\right)^r\right)^d \geq 1 - \left(\frac{r}{n}\right)^d \geq 1 - \left(\frac{\ln n}{n}\right)^4 \geq 1 - \frac{1}{n^2}.$$

This completes the proof. □

Next, we give a general upper bound on the run time of the (1+1) EA on arbitrary linear functions under uniform constraint. For this we use a method based on the expected gain in weight of bits, inspired by [32]. Recall that $w_{\max}$ denotes the maximal bit-weight of the objective function.

Theorem 11. *The expected optimization time of the* (1+1) EA *on any linear function under uniform constraint* B *is* $\mathrm{O}(n^2 \log(B\, w_{\max}))$.

Proof. It is sufficient to start the analysis with a feasible solution, by Lemma 7. In particular, the (1+1) EA will never sample an infeasible solution during the remainder of the optimization process. W.l.o.g. the weights of the objective function f are in descending order, i.e., $w_{\max} = w_1 \geq w_2 \geq \cdots \geq w_n$. The maximum value of f under uniform constraint B is thus $f_{\max} = \sum_{i=1}^{B} w_i$. To any feasible search point x we assign a potential value $g(x) = f_{\max} - f(x)$. Function g is non-negative and attains its minimum 0 just in case $f(x)$ is optimal.

We again refer to the first B bits as the first block and the last $n-B$ bits as the second block. Symbol $\overline{x_i}$ is used for the negation of the i-th bit. We define two auxiliary functions,

$$\mathrm{loss}(x) = \sum_{i=1}^{B} w_i\,(1 - x_i); \quad \mathrm{surplus}(x) = \sum_{j=B+1}^{n} w_j\, x_j.$$

Intuitively, $\mathrm{loss}(x)$ measures the total weight of the missing bits in the first block, while $\mathrm{surplus}(x)$ is the sum of weights of the superfluous bits in the second block. The potential function g can be reformulated as

$$g(x) = \mathrm{loss}(x) - \mathrm{surplus}(x).$$

Assume that the currently best solution x is non-optimal. Let $k \geq 1$ denote the number of 0-bits in the first block of x. We pessimistically assume that $|x|_1$ has already reached the cardinality bound B. In this case the expected drift in the above potential is minimal. (For solutions with Hamming weight strictly smaller than B a mutation flipping a single 0-bit is enough to improve on the fitness value.) Necessarily, there are also exactly k 1-bits in the second block. By $\mathcal{E}_{1,2}$ we denote the event that one 0 in the first block, one 1 in the second, and no other position flips in this round. Using Lemma 1, we bound the expected drift in g by the conditional drift under condition $\mathcal{E}_{1,2}$. Any of the

k 0s in the first block are equally likely to flip and together they make up for the whole value of $\text{loss}(x)$. Thus, the average potential decrease by flipping one of them is $\text{loss}(x)/k$. The same argument for the second block gives an average *increase* of the potential of $\text{surplus}(x)$. These two effects counteract each other. However, the resulting drift is still large enough as

$$\begin{aligned}\mathrm{E}[g(x) - g(x')] &\geq \mathrm{E}[g(x) - g(x') \mid \mathcal{E}_{1,2}] \cdot \Pr[\mathcal{E}_{1,2}] \\ &\geq \left(\frac{\text{loss}(x)}{k} - \frac{\text{surplus}(x)}{k}\right)\frac{k^2}{en^2} = g(x)\frac{k}{en^2} \geq \frac{g(x)}{en^2}.\end{aligned}$$

Since the potential is at most $f_{\max}$, Theorem 5 implies

$$\mathrm{E}[T] \leq en^2\,(\ln(f_{\max}) + 1) \leq en^2\,(\ln(B\,w_{\max}) + 1). \qquad \square$$

Consider the BinVal-function defined as $\textsc{BinVal}(x) = \sum_{i=1}^{n} 2^{n-i}x_i$. It serves as an extreme example of a linear function where every weight is strictly larger than the sum of all smaller weights. Further increasing the weight differences does not change the acceptance behavior of the (1+1) EA. Hence, we can assume $w_{\max} \leq 2^n$ for any linear function and Theorem 11 gives a worst-case expected optimization time of $\mathrm{O}(n^3)$. However, we suspect the log-factor appearing in above bound to be an artefact of the analysis and consequently conjecture that every linear function can be optimized in time $\mathrm{O}(n^2)$, for all choices of B. This is supported by the observation that very different classes of linear functions adhere to this bound. As a first example we return to the class of functions used in Theorem 10. The case of BinVal is treated in Section 3.3.

A clarifying remark may be advised here. The next two theorems show, for two classes of functions, a universal quadratic upper bound in n for any uniform constraint B. In contrast, Theorem 10 shows a quadratic lower bound only for specific choices of B. Although Theorems 10 & 12 share the same class of functions, this does not imply a tight bound with respect to both n and B.[2]

[2] $\Theta(n^2)$ for all B is easily refuted: the case $B = n$ has optimization time $\mathrm{O}(n \log n)$ [9].

Theorem 12. *Let B be an arbitrary integer between 1 and n, and $0 < \varepsilon < 1/B$ a real number. The optimization time of the* (1+1) EA *on the objective function*

$$f(x) = \sum_{i=1}^{B} (1+\varepsilon)\, x_i + \sum_{j=B+1}^{n} x_j$$

under uniform constraint B is $\mathrm{O}(n^2)$ in expectation.

Proof. Assume that the optimization starts in the feasible region. By Lemma 7, this does not affect the asymptotic bound. The key observation of this proof is that any mutation of a feasible solution that reduces the number of 1-bits is rejected, implying a behavior similar to that on OneMax. For the offspring to be accepted, the fitness cannot be worse than that of the parent. Hence, any net loss of 1-bits from the last $n - B$ bits (the second block) must be compensated by the same number of 1s in the first B positions (the first block). Note that due to $\varepsilon < 1/B$ fewer 1-bits do not suffice. Conversely, if a mutation reduces the number of 1s in the first block, only a strictly larger increase in the second block can make up for this since $\varepsilon > 0$.

Theorem 9 implies that Algorithm 1 samples a solution x with $|x|_1 = B$ in expected time $\mathrm{O}(n \log n)$. By the discussion above, the algorithm stays at the cardinality bound during the rest of the optimization. We define the potential function as the number of 0-bits in the first block, i.e., $g(x) = B - \sum_{i=1}^{B} x_i$. Since the string $x^* = 1^B 0^{n-B}$ is the unique maximum of f, $g(x) = 0$ is equivalent to optimality. While being at the cardinality bound, the *second* block must have exactly $g(x)$ bits set to 1. Focusing on mutations which flip a single 0-bit in the first block and a single 1-bit in the second block, we get

$$\mathrm{E}[g(x) - g(x') \mid |x|_1 = B] \geq \frac{g(x)^2}{n^2} \left(1 - \frac{1}{n}\right)^{n-2} \geq \frac{g(x)^2}{en^2}.$$

Let T' denote the random number of iterations until the (1+1) EA reaches the optimum x^* starting from a solution with B 1-bits. The Variable Drift Theorem (Theorem 2) applied to the drift function $h(s) = s^2/en^2$, starting

potential $s_0 = B$, and minimum $s_{\min} = 1$ yields

$$\mathrm{E}[T'] \leq en^2 + \int_1^B \frac{en^2}{s^2}\,\mathrm{d}s \leq en^2\left(1 + \int_1^\infty \frac{1}{s^2}\,\mathrm{d}s\right) = 2en^2.$$

Combining this with the bound on the time to find a solution with exactly B 1-bits gives the theorem. □

3.3. BinVal

The next theorem states a quadratic optimization time also for the BinVal function under uniform constraint. BinVal is defined by assigning to a bit string its binary value,

$$\textsc{BinVal}(x) = \sum_{i=1}^{n} 2^{n-i}x_i.$$

The weights on the various positions differ by orders of magnitude. As a consequence, the offspring has a higher BinVal-fitness if and only if the left-most flipping bit is a 0-bit, cf. [9].

Theorem 13. *For arbitrary values of B, the expected optimization time of the* (1+1) EA *on* BinVal *under uniform constraint B is in* $\mathrm{O}(n^2)$.

When proving Theorem 13 we will face the following situation: As explained above the offspring will be accepted if the left-most flipping bit was a 0-bit and the amount of 1-bits in the offspring is at most B. Therefore, if the amount of 1-bits in the parent is less than B and the left-most flipping bit is the only flipping 0-bit, the mutation will be accepted. However, if the amount of 1-bits in the parent is B, then we have to have at least one flipping 1-bit in order to have a feasible offspring. Exactly this situation will be reached asymptotically fast due to Lemma 7 when compared to the desired optimization time in $\mathrm{O}(n^2)$.

Inspired by the proof in [6] of the Linear Function problem we are going to study this situation. Employing for a bit string x a potential function $g(x)$ allows us to measure the expected progress made when comparing parent and offspring: the expected drift. The left-most flipping 0-bit will yield a positive

drift, however the condition of additional flipping 1-bits will yield a negative drift. This condition is necessary, as the drift under the event of exactly B 1-bits is a lower bound of the general drift. We condition the left-most flipping to be a 0-bit at position i. By studying the drift in the substring of bits with index larger than i we will measure the negative drift. After a careful study of both expressions we will observe that the positive drift outweighs the negative yielding a lower bound on the expected drift. Finally, we will derive the desired runtime by applying the Variable Drift Theorem 2 on said lower bound.

Proof. We assume to have a feasible solution x with at most B 1-bits for our analysis, since the time until the (1+1) EA samples one is asymptotically smaller than the desired optimization time in $O(n^2)$ due to Lemma 7. In order to apply a drift theorem we have to construct a suitable potential function for x. We observe that the optimal bit string for BinVal is 1^B0^{n-B} with B ones in the first entries followed by $n-B$ zeros in the last entries. Thus, the constraint B influences the optimization time. We take this into account by rewarding 1-bits only in the first B positions of the bit string. We do not regard the last $n-B$ positions at all in order to simplify the analysis.

We associate with a bit string x the potential function

$$g(x) = \sum_{i=1}^{B} 2\left(1-\frac{1}{n}\right)^i (1-x_i),$$

whose minimum of 0 is reached by the optimal bit string 1^B0^{n-B}. In fact, every other bit string minimizing $g(x)$ is a non-feasible solution due to the constraint of at most B 1-bits. We recall that for a best-so-far solution x the offspring of x under the (1+1) EA is denoted x'. By bounding the expected drift, i.e. the expected change $E[g(x)-g(x')]$, we are going to obtain the claimed runtime via the Variable Drift Theorem 2.

Due to the exponential behavior of BinVal the offspring x' will be rejected if the leftmost flipping bit is a 1-bit, whereas it may be accepted if the leftmost flipping bit is a 0-bit. Let L_0 be the set of indices $i \in [1, B]$, whose bit x_i

is 0. Further, let $\mathcal{A}$ be the event that the leftmost flipping bit of x is a 0-bit with index in $[1, B]$. $\mathcal{A}$ decomposes into the events $(\mathcal{A}_i)_{i \in L_0}$ fixing the leftmost flipping bit to be at position i. From the law of total expectation, we get

$$\mathrm{E}[g(x) - g(x')] = \mathrm{E}[g(x) - g(x') \mid \mathcal{A}] \Pr[\mathcal{A}] = \sum_{i \in L_0} \mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i] \Pr[\mathcal{A}_i].$$

For $i \in L_0$ the probability for $\mathcal{A}_i$ is at least the probability that only the i-th bit flips. This implies

$$\begin{aligned} \sum_{i \in L_0} \mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i] \Pr[\mathcal{A}_i] &\geq \sum_{i \in L_0} \mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i] \; \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \qquad (1) \\ &\geq \frac{1}{en} \sum_{i \in L_0} \mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i]. \end{aligned}$$

We will now focus on the expected drift under the condition $\mathcal{A}_i$ holds. If we have not yet reached the bound of B 1-bits we observe that already one flipping 0-bit contributes to the drift without affecting the probability of a flipping 1-bit. However, when we reach the bound every flipping 0-bit must be adjoined to at least one flipping 1-bit in order to obtain an accepted offspring. Hence, since every flipping 1-bit with index in $[1, B]$ reduces the drift, we pessimistically assume the case of B 1-bits from this point onwards.

Let X_i be the random variable denoting the number of flipping 1-bits with index *strictly* larger than i. Since the leftmost flipping bit must be a 0-bit, the offspring is accepted if and only if the condition $(X_i \geq 1)$ holds. Thus, Lemma 1 with the event $\mathcal{E} = (X_i \geq 1)$ yields

$$\mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i] = \mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i, \, (X_i \geq 1)] \Pr[X_i \geq 1]. \qquad (2)$$

Regarding the drift $\mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i, (X_i \geq 1)]$, flipping the 0-bit at position $i \leq B$ contributes a positive change of $2(1 - 1/n)^i$. We assume pessimistically that there are no further flipping 0-bits. The positive change is counteracted by a negative change, which is caused by the flipping 1-bits. We

denote by $x_{[i,j]}$ the substring of x constrained to the entries with indices in the interval $[i,j]$. The potential function constrained to the substring $x_{[i,j]}$ denoted by $g(x_{[i,j]})$ enables us to study the drift in $x_{[i,j]}$. The drift in the substring $x_{[i+1,B]}$ cannot be positive due to the condition $(X_i \geq 1)$ and we deduce

$$\begin{aligned} &\mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i,\, (X_i \geq 1)] \Pr[X_i \geq 1] \\ &\geq \left(2\left(1-\frac{1}{n}\right)^i + \mathrm{E}[g(x_{[i+1,B]}) - g(x'_{[i+1,B]}) \mid X_i \geq 1]\right) \Pr[X_i \geq 1]. \end{aligned} \tag{3}$$

We study the negative change in $[i+1,B]$ first. Let $l(i)$ denote the number of 1-bits with index in $[i+1,B]$. In order to apply Lemma 1 we observe the random variable $-(g(x_{[i+1,B]}) - g(x'_{[i+1,B]}))$ with the event $\mathcal{E} = (X_i \geq 1)$. As required this random variable assumes values in $\mathbb{R}_0^+$ and $(X_i \geq 1)$ has to hold in order for $-(g(x_{[i+1,B]}) - g(x'_{[i+1,B]}))$ to be positive. This yields

$$\begin{aligned} &\mathrm{E}[-(g(x_{[i+1,B]}) - g(x'_{[i+1,B]})) \mid X_i \geq 1] \Pr[X_i \geq 1] \\ &= \mathrm{E}[-(g(x_{[i+1,B]}) - g(x'_{[i+1,B]}))]. \end{aligned}$$

Due to the linearity of expectation this also holds for the random variable $g(x_{[i+1,B]}) - g(x'_{[i+1,B]})$. Regarding a lower bound we already assumed no flipping 0-bits in the constrained bit string. Each bit flips independently with probability $1/n$ and by the equality for a partial geometric series we deduce

$$\begin{aligned} \mathrm{E}[g(x_{[i+1,B]}) - g(x'_{[i+1,B]})] &\geq -\frac{1}{n} \sum_{j=i+1}^{i+l(i)} 2\left(1-\frac{1}{n}\right)^j \\ &= -2\left(\left(1-\frac{1}{n}\right)^{i+1} - \left(1-\frac{1}{n}\right)^{i+l(i)+1}\right). \end{aligned} \tag{4}$$

What is left is to derive an explicit expression for $\Pr[X_i \geq 1]$: the probability that at least one 1-bit with index larger than i flips. This probability depends on the number $l(i)$ as well as on the number of 1-bits with index in $[B+1,n]$ denoted by r. The probability that in one mutation step a single bit does not

flip is $1 - 1/n$, which yields

$$\Pr[X_i \geq 1] = 1 - \Pr[X_i = 0] = 1 - \left(1 - \frac{1}{n}\right)^{l(i)+r}. \tag{5}$$

We combine the derived expressions (2), (3), (4), and (5) to the bound

$$\mathrm{E}[g(x) - g(x') \mid A_i] \geq 2\left(1 - \frac{1}{n}\right)^{i}\left(\frac{1}{n} + \left(1 - \frac{1}{n}\right)^{l(i)+1} - \left(1 - \frac{1}{n}\right)^{l(i)+r}\right). \tag{6}$$

Since we have B 1-bits and have not yet reached the optimum, there has to be at least one 1-bit with index in $[B+1, n]$ implying $r \geq 1$. Therefore, we observe a positive drift. Let $f(r, l, n)$ be the last factor of (6). For the case $r = 1$ this factor is equal to $1/n$. In order to bound $f(r, l(i), n)$ for $r \geq 2$ we observe that the drift decreases if $l(i)$ increases. Since $l(i) \leq B - i$, this yields

$$\begin{aligned} f(r, l(i), n) &= \frac{1}{n} + \left(1 - \frac{1}{n}\right)^{l(i)}\left(1 - \frac{1}{n} - \left(1 - \frac{1}{n}\right)^{r}\right) \\ &\geq \frac{1}{n} + \left(1 - \frac{1}{n}\right)^{B-i}\left(1 - \frac{1}{n} - \left(1 - \frac{1}{n}\right)^{r}\right). \end{aligned}$$

For $r \geq 2$ this factor is positive, we apply the well-known bound $(1 + x)^n \leq 1/(1 - nx)$ (for $n \in \mathbb{N}$ and $x \in [-1, 0]$) and derive

$$\begin{aligned} f(r, l(i), n) &\geq \frac{1}{n} + \left(1 - \frac{1}{n}\right)^{B-i}\left(1 - \frac{1}{n} - \frac{1}{1 + r/n}\right) \\ &= \frac{1}{n} + \left(1 - \frac{1}{n}\right)^{B-i}\left(\frac{r}{r+n} - \frac{1}{n}\right) \\ &\geq \frac{1}{n} + \left(1 - \frac{1}{n}\right)^{B-i}\left(\frac{r}{4n}\right). \end{aligned} \tag{7}$$

Regarding the last inequality, the expression holds if $r/(r + n) \geq (r + 4)/4n$, which is equivalent to $r(3n - r - 4) \geq 4n$. By the partial derivative of the left side with respect to r we observe for $n \geq 5$ a positive growth due to $r \leq n$. Therefore, the left side is minimal if r is minimal. Since we assumed $2 \leq r$, we

obtain $6n - 6 \geq 4n$, which holds for $n \geq 3$ justifying the last inequality in (7)

For the case $r \geq 2$ we obtain the drift for the event A_i by (7), $B \leq n-1$ and the usual bound $(1-1/n)^{n-1} \leq e^{-1}$

$$\begin{aligned}\mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i] &\geq 2\left(1-\frac{1}{n}\right)^i f(r, l(i), n) \\ &\geq \frac{2}{n}\left(1-\frac{1}{n}\right)^i + \left(1-\frac{1}{n}\right)^B \left(\frac{r}{2n}\right) \geq \frac{r}{2en}.\end{aligned}$$

This drift turns out to be valid for the case $r = 1$ as well. Here, we observe that $i \leq n-1$ since $B \leq n-1$. More precisely, if the leftmost flipping 0-bit is the last bit for the case of exactly B 1-bits, the mutation will never be accepted. We observe for $r = 1$,

$$\begin{aligned}\mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i] &\geq 2\left(1-\frac{1}{n}\right)^i f(1, l(i), n) = \frac{2}{n}\left(1-\frac{1}{n}\right)^i \\ &\geq \frac{2}{en} \geq \frac{r}{2en}.\end{aligned}$$

With the expected drift under the event $\mathcal{A}_i$ at hand, we now turn towards deriving the total drift. In order to obtain a bound on the total drift, due to (1) we need to sum up the $\mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i]$ with indices in L_0 – the set of indices in $[1, B]$ whose bit is a 0-bit. Since we derived a bound for $\mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i]$ not conditional on the index i, it suffices to deduce the cardinality of L_0. We assumed to be in the case with B 1-bits, hence each 1-bit with index in $[B+1, n]$ yields a 0-bit with index in $[1, B]$. Thus, the cardinality of L_0 is r and we obtain

$$\begin{aligned}\mathrm{E}[g(x) - g(x')] &\geq \frac{1}{en}\sum_{i \in L_0} \mathrm{E}[g(x) - g(x') \mid \mathcal{A}_i] \geq \frac{1}{en}\sum_{i=1}^{r} \frac{r}{2en} \\ &= \frac{r^2}{2e^2n^2}.\end{aligned}$$

We expressed the drift in terms of r. However, the Variable Drift Theorem 2 requires a bound depending on the current potential $g(x)$. We observe that $g(x)$ and r correlate by $g(x) = \sum_{i \in L_0} 2\,(1-1/n)^i \leq 2|L_0| = 2r$. Therefore, the

expected drift is

$$\mathrm{E}[g(x) - g(x')] \geq \frac{r^2}{2e^2n^2} \geq \frac{g(x)^2}{8e^2n^2} =: h(g(x)).$$

We apply the Variable Drift Theorem 2 on function h with the initial potential $s_0 \leq 2B$ and $s_{min} = 1$ to obtain [3]

$$\mathrm{E}[T] \leq \frac{s_{\min}}{h(s_{\min})} + \int_{s_{\min}}^{s_0} \frac{1}{h(s)}\,\mathrm{d}s = 8e^2n^2\left(1 + \int_1^{2B} \frac{1}{s^2}\,\mathrm{d}s\right) \leq 16e^2n^2.$$

This establishes the theorem. □

4. Linear Constraint

In this section we investigate linear functions under general linear constraints. That means that we can choose the weights $b_i > 0$ of the constraint function arbitrarily. The resulting optimization problem is capable of encoding the NP-complete KnapSack problem, cf. [22], thus we do not expect a polynomial run time of the (1+1) EA. In fact, we show that already the optimization time (the number of iterations until optimality) is exponential in general.

He, Mitavskiy, and Zhou have shown the existence of trap-like problem instances fitting the general KnapSack formulation which cannot be solved efficiently by simple evolutionary computation methods [16]. The algorithms get trapped in a local optimum that has a large Hamming distance to any globally optimal solution. The class of instances investigated in [16] consists of $n-1$ items having weight and profit 1 and a single item having a large weight and profit. In contrast, we show that even in the case of OneMax as the objective function–assigning the same profit to all objects–there are instances the (1+1) EA cannot optimize efficiently. Note that this case can easily be solved by a greedy algorithm.

[3] We note that the Drift depends on the current potential quadratically. Hence, we cannot apply the Multiplicative Drift Theorem 5.

Theorem 14. *There is a linear function b and a real $B > 0$ such that the optimization time of the* (1+1) EA *on* OneMax *under linear constraint $b(x) \leq B$ is exponential, not only in expectation but even with overwhelming probability.*[4]

Proof. We define the constraint function $b(x)$ as

$$b(x) = \sum_{i=1}^{2n/3} n\,x_i + \sum_{i=2n/3+1}^{n} (n+1)\,x_i$$

together with the bound $B = 2n^2/3$. This choice ensures that every string x with Hamming weight $|x|_1 < 2n/3$ is feasible, while the others are infeasible. The *sole exception* is the optimal (feasible) solution $x^* = 1^{2n/3}0^{n/3}$. In other words, the collection of strings with exactly $2n/3 - 1$ bits set to 1 form a large plateau of equal, sub-optimal fitness. We condition the following analysis on the initial solution being feasible, which happens with overwhelming probability due to Chernoff bounds. Hence, the (1+1) EA never adopts an infeasible search point.

We prove an *unbiasedness* property [27] of the underlying random process. Informally, as long as the optimum has not been found, the probability of a bit string to be sampled in round t depends only on the number of the bits set to 1, not on their position. To state this more formally, we need some additional notation highlighting the effect of selection in the optimization. Let $(X^{(t)})_{t\geq 0}$ be the series of random variables with values in $\{0,1\}^n$ denoting the search points adopted by the (1+1) EA *after* the selection step in round t. Further, define $Y^{(t)}$ as the offspring (of individual $X^{(t-1)}$) created in round $t > 0$ *before* any selection takes place. For a permutation $\pi\colon \{1,\dots n\} \rightarrowtail\!\!\!\!\!\rightarrow \{1,\dots n\}$, let $\pi(x) = x_{\pi(1)}x_{\pi(2)}\dots x_{\pi(n)}$ be the string obtained from x by deranging its positions according to π.

While the optimum x^* has not yet been found, the search behaves like the (1+1) EA on constrained OneMax with a cardinality bound of $2n/3$, cp. Theorem 9. We claim that the random process $(Y^{(t)})_t$ is unbiased towards the

[4] We use the term *with overwhelming probability* for a success probability of at least $1 - 1/2^{cn}$ for some constant $c > 0$.

position of the 1-bits. In full detail, we want to prove that, for any $t > 0$, permutation π, and bit string $y \in \{0,1\}^n$,

$$\Pr[Y^{(t)} = y \mid X^{(t-1)} \neq x^*] = \Pr[Y^{(t)} = \pi(y) \mid X^{(t-1)} \neq x^*].$$

We do this by induction over t. Lehre and Witt [27] have characterized the standard bit mutation of the (1+1) EA as an *unary, unbiased variation operator*. In particular, it is invariant under permutation,

$$\Pr[Y^{(t)} = y \mid X^{(t-1)} = x] = \Pr[Y^{(t)} = \pi(y) \mid X^{(t-1)} = \pi(x)].$$

The initial solution $X^{(0)}$ is uniformly distributed over $\{0,1\}^n$, that means, $\Pr[X^{(0)} = x] = \Pr[X^{(0)} = \pi(x)]$. Together with the unbiasedness condition, this proves the claim for $t = 1$.

If $t > 1$, the search point $X^{(t-1)}$ is equal to the offspring $Y^{(t^*)}$, where $t^* < t$ is the the round in which it has been selected. By the induction hypothesis the claim holds for t^*. The selection itself is also unbiased as the objective function OneMax (with the additional rejection rule) is invariant under bit permutation. Hence, we can express $\Pr[Y^{(t)} = y \mid X^{(t-1)} \neq x^*]$ as the following sum of conditional probabilities. Hereby, observe that the condition $X^{(t-1)} \neq x^*$ is implicitly fulfilled for all strings x such that $|x|_1 < 2n/3$.

$$\begin{aligned}
&\Pr[Y^{(t)} = y \mid X^{(t-1)} \neq x^*] \\
&\quad = \sum_{x\colon |x|_1 < \frac{2}{3}n} \Pr[Y^{(t)} = y \mid X^{(t-1)} = x] \Pr[X^{(t-1)} = x] \\
&\quad = \sum_{x\colon |x|_1 < \frac{2}{3}n} \Pr[Y^{(t)} = \pi(y) \mid X^{(t-1)} = \pi(x)] \Pr[X^{(t-1)} = \pi(x)] \\
&\quad = \Pr[Y^{(t)} = \pi(y) \mid X^{(t-1)} \neq x^*].
\end{aligned}$$

There are $\binom{n}{n/3}$ bit strings with Hamming weight exactly $2n/3$, but only one of them is optimal. The above claim implies that for any $t > 0$ the probability

of finding x^* in round t is at most

$$\Pr[Y^{(t)} = x^* \mid X^{(t-1)} \neq x^*] \leq 1 \Big/ \binom{n}{n/3} \leq 3^{-\frac{n}{3}},$$

which yields an exponential waiting time. Moreover, a simple union bound over t shows that for a suitably small constant $c > 0$ the probability of finding the optimum within the first 2^{cn} iterations is exponentially small. □

5. Conclusion

Studying the run time behavior of linear functions has provided many new tools for analyzing evolutionary computing techniques and set the basis for run time studies for more complex problems. With this paper, we have contributed to the area of run time analysis of evolutionary computing by studying classes of linear functions under a linear constraint. This setting, in its most general sense, is equivalent to the KnapSack problem. Central to the area of run time analysis is the linear function OneMax. We have focused our attention on problem classes where the objective function or the constraint is given by OneMax as well.

Our theoretical investigations show that the (1+1) EA can handle uniform constraints efficiently, but fails for more general constraints even on OneMax. The latter includes instances that are easily solved by a greedy algorithm. The constraint handling we employed directs the search within the infeasible region towards the feasible region by adding a penalty dependent on the distance to the constraint. However, the search within the feasible region is not guided by any knowledge about the constraint. Therefore, it is interesting to investigate whether additional information can help direct the search such that OneMax with non-uniform constraint can be handled efficiently.

Acknowledgments

The authors would like to thank the anonymous reviewers as well as the guest editors, Pietro S. Oliveto and Andrew M. Sutton, for their constructive feedback.

We are also thankful for the fruitful discussions and the valuable hints of the participants of FOGA 2017 and, in particular, Dirk Sudholt.

The research leading to this article has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 618091 (SAGE) and from the Australian Research Council under grant agreements DP140103400 and DP160102401.

References

[1] Auger, A., & Doerr, B. (2011). *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. Singapore: World Scientific Publishing.

[2] Beier, R. (2005). *Probabilistic Analysis of Discrete Optimization Problems*. Ph.D. thesis Universität des Saarlandes. URL: `http://scidok.sulb.uni-saarland.de/volltexte/2005/464`.

[3] Corus, D., Dang, D. C., Eremeev, A. V., & Lehre, P. K. (y. TBD). Level-Based Analysis of Genetic Algorithms and Other Search Processes. *IEEE Transactions on Evolutionary Computation*, (p. TBD). doi:`10.1109/TEVC.2017.2753538`. Ahead of print.

[4] Corus, D., & Oliveto, P. S. (y. TBD). Standard Steady State Genetic Algorithms Can Hillclimb Faster than Mutation-only Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, (p. TBD). doi:`10.1109/TEVC.2017.2745715`. Ahead of print.

[5] Dang, D., & Lehre, P. K. (2015). Simplified Runtime Analysis of Estimation of Distribution Algorithms. In *Proceedings of the 2015 Genetic and Evolutionary Conference (GECCO)* (pp. 513–518). doi:`10.1145/2739480.2754814`.

[6] Doerr, B., Johannsen, D., & Winzen, C. (2012). Multiplicative Drift Analysis. *Algorithmica*, *64*, 673–697. doi:`10.1007/s00453-012-9622-x`.

[7] Doerr, C. (2018). Complexity theory for discrete black-box optimization heuristics. *CoRR*, *abs/1801.02037*. URL: http://arxiv.org/abs/1801.02037. arXiv:1801.02037.

[8] Droste, S. (2004). Analysis of the (1+1) EA for a Noisy OneMax. In *Proceedings of the 2004 Genetic and Evolutionary Conference (GECCO)* (pp. 1088–1099). doi:10.1007/978-3-540-24854-5_107.

[9] Droste, S., Jansen, T., & Wegener, I. (2002). On the Analysis of the (1+1) Evolutionary Algorithm. *Theoretical Computer Science*, *276*, 51–81. doi:10.1016/S0304-3975(01)00182-7.

[10] Droste, S., Jansen, T., & Wegener, I. (2006). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory Comput. Syst.*, *39*, 525–544. URL: https://doi.org/10.1007/s00224-004-1177-z. doi:10.1007/s00224-004-1177-z.

[11] Feldmann, M., & Kötzing, T. (2013). Optimizing Expected Path Lengths with Ant Colony Optimization Using Fitness Proportional Update. In *Proceedings of the 12th Workshop on Foundations of Genetic Algorithms (FOGA)* (pp. 65–74). doi:10.1145/2460239.2460246.

[12] Frame, J. S. (1945). Mean Deviation of the Binomial Distribution. *The American Mathematical Monthly*, *52*, 377–379. doi:10.2307/2304638.

[13] Friedrich, T., Kötzing, T., Lagodzinski, J. G., Neumann, F., & Schirneck, M. (2017). Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints. In *Proceedings of the 14th Workshop on Foundations of Genetic Algorithms (FOGA)* (pp. 45–54). doi:10.1145/3040718.3040728.

[14] Gießen, C., & Kötzing, T. (2016). Robustness of Populations in Stochastic Environments. *Algorithmica*, *75*, 462–489. doi:10.1007/s00453-015-0072-0.

[15] Harman, M. (2011). Software Engineering Meets Evolutionary Computation. *IEEE Computer*, *44*, 31–39. doi:10.1109/MC.2011.263.

[16] He, J., Mitavskiy, B., & Zhou, Y. (2014). A Theoretical Assessment of Solution Quality in Evolutionary Algorithms for the Knapsack Problem. In *Proceedings of the 2014 Congress on Evolutionary Computation (CEC)* (pp. 141–148). doi:10.1109/CEC.2014.6900442.

[17] He, J., & Yao, X. (2004). A Study of Drift Analysis for Estimating Computation Time of Evolutionary Algorithms. *Natural Computing*, *3*, 21–35. doi:10.1023/B:NACO.0000023417.31393.c7.

[18] Hwang, H.-K., Panholzer, A., Rolin, N., Tsai, T.-H., & Chen, W.-M. (y. TBD). Probabilistic Analysis of the (1+1)-Evolutionary Algorithm. *Evolutionary Computation*, (p. TBD). doi:10.1162/evco_a_00212. Ahead of print.

[19] Jansen, T. (2013). *Analyzing Evolutionary Algorithms - The Computer Science Perspective*. Natural Computing Series. Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-17339-4.

[20] Jansen, T., & Wegener, I. (2001). Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Trans. Evolutionary Computation*, *5*, 589–599. URL: https://doi.org/10.1109/4235.974841. doi:10.1109/4235.974841.

[21] Johannsen, D. (2010). *Random Combinatorial Structures and Randomized Search Heuristics*. Ph.D. thesis Universität des Saarlandes. URL: http://scidok.sulb.uni-saarland.de/volltexte/2011/3529.

[22] Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems*. Berlin, Heidelberg: Springer. doi:10.1007/978-3-540-24777-7.

[23] Kötzing, T. (2016). Concentration of First Hitting Times Under Additive Drift. *Algorithmica*, *75*, 490–506. doi:10.1007/s00453-015-0048-0.

[24] Kötzing, T., Lissovoi, A., & Witt, C. (2015). (1+1) EA on Generalized Dynamic OneMax. In *Proceedings of the 13th Workshop on Foundations of Genetic Algorithms (FOGA)* (pp. 40–51). doi:10.1145/2725494.2725502.

[25] Kötzing, T., Neumann, F., Sudholt, D., & Wagner, M. (2011). Simple Max-Min Ant Systems and the Optimization of Linear Pseudo-Boolean Functions. In *Proceedings of the 11th Workshop on Foundations of Genetic Algorithms (FOGA)* (pp. 209–218). doi:10.1145/1967654.1967673.

[26] Le Goues, C., Nguyen, T., Forrest, S., & Weimer, W. (2012). GenProg: A Generic Method for Automatic Software Repair. *IEEE Transaction on Software Engineering*, *38*, 54–72. doi:10.1109/TSE.2011.104.

[27] Lehre, P. K., & Witt, C. (2012). Black-Box Search by Unbiased Variation. *Algorithmica*, *64*, 623–642. doi:10.1007/s00453-012-9616-8.

[28] Lengler, J. (2017). Drift Analysis. *CoRR*, *abs/1712.00964*. URL: http://arxiv.org/abs/1712.00964. arXiv:1712.00964.

[29] Li, X., Bonyadi, M. R., Michalewicz, Z., & Barone, L. (2013). Solving a Real-world Wheat Blending Problem Using a Hybrid Evolutionary Algorithm. In *Proceedings of the 2013 Congress on Evolutionary Computation (CEC)* (pp. 2665–2671). doi:10.1109/CEC.2013.6557891.

[30] Lückehe, D., Wagner, M., & Kramer, O. (2015). On Evolutionary Approaches to Wind Turbine Placement with Geo-Constraints. In *Proceedings of the 2015 Genetic and Evolutionary Conference (GECCO)* (pp. 1223–1230). doi:10.1145/2739480.2754690.

[31] Mühlenbein, H. (1992). How Genetic Algorithms Really Work: Mutation and Hillclimbing. In *Proceedings of the Second Conference on Parallel Problem Solving from Nature (PPSN)* (pp. 15–26).

[32] Neumann, F., & Wegener, I. (2007). Randomized Local Search, Evolutionary Algorithms, and the Minimum Spanning Tree Problem. *Theoretical*

Computer Science, *378*, 32–40. URL: https://doi.org/10.1016/j.tcs.2006.11.002. doi:10.1016/j.tcs.2006.11.002.

[33] Neumann, F., & Witt, C. (2010). *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-16544-3.

[34] Oliveto, P. S., & Witt, C. (2015). Improved Time Complexity Analysis of the Simple Genetic Algorithm. *Theoretical Computer Science*, *605*, 21–41. doi:10.1016/j.tcs.2015.01.002.

[35] Rowe, J. E., & Sudholt, D. (2012). The Choice of the Offspring Population Size in the (1,λ) EA. In *Proceedings of the 2012 Genetic and Evolutionary Conference (GECCO)* (pp. 1349–1356). doi:10.1145/2330163.2330350.

[36] Spielman, D. A., & Teng, S. (2009). Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice. *Communications of the ACM*, *52*, 76–84. doi:10.1145/1562764.1562785.

[37] Stokes, C. S., Simpson, A. R., & Maier, H. R. (2015). A Computational Software Tool for the Minimization of Costs and Greenhouse Gas Emissions Associated with Water Distribution Systems. *Environmental Modelling and Software*, *69*, 452–467. doi:10.1016/j.envsoft.2014.11.004.

[38] Sudholt, D., & Witt, C. (2010). Runtime Analysis of a Binary Particle Swarm Optimizer. *Theoretical Computer Science*, *411*, 2084–2100. doi:10.1016/j.tcs.2010.03.002.

[39] Witt, C. (2013). Tight Bounds on the Optimization Time of a Randomized Search Heuristic on Linear Functions. *Combinatorics, Probability and Computing*, *22*, 294–318. doi:10.1017/S0963548312000600.

[40] Zheng, F., Simpson, A. R., & Zecchin, A. C. (2015). Improving the Efficiency of Multi-objective Evolutionary Algorithms Through Decomposition: An Application to Water Distribution Network Design. *Environmen-*

tal Modelling and Software, *69*, 240–252. doi:10.1016/j.envsoft.2014.08.022.

[41] Zhou, Y., & He, J. (2007). A Runtime Analysis of Evolutionary Algorithms for Constrained Optimization Problems. *IEEE Transactions on Evolutionary Computation*, *11*, 608–619. doi:10.1109/TEVC.2006.888929.