Algorithmica
https://doi.org/10.1007/s00453-018-0451-4

Reoptimization Time Analysis of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints

Feng Shi[1] · **Martin Schirneck**[2] · **Tobias Friedrich**[2] · **Timo Kötzing**[2] · **Frank Neumann**[3]

Received: 31 August 2017 / Accepted: 3 May 2018

Abstract Rigorous runtime analysis is a major approach towards understanding evolutionary computing techniques, and in this area linear pseudo-Boolean objective functions play a central role. Having an additional linear constraint is then equivalent to the NP-hard Knapsack problem, certain classes thereof have been studied in recent works. In this article, we present a dynamic model of optimizing linear functions under uniform constraints. Starting from an optimal solution with respect to a given constraint bound, we investigate the runtimes that different evolutionary algorithms need to recompute an optimal solution when the constraint bound changes by a certain amount. The classical (1+1) EA and several population-based algorithms are designed for that purpose, and are shown to recompute efficiently. Furthermore, a variant of the $(1+(\lambda, \lambda))$ GA for the dynamic optimization problem is studied, whose performance is better when the change of the constraint bound is small.

Keywords Evolutionary algorithm · Runtime analysis · Reoptimization time · Dynamic constraint · Uniform constraint

A preliminary version of this work was presented at the *2017 Genetic and Evolutionary Computation Conference* (GECCO) [18].

✉ Feng Shi
fengshi@mail.csu.edu.cn

1 School of Information Science and Engineering, Central South University, Changsha, Hunan, People's Republic of China

2 Hasso Plattner Institute, Potsdam, Germany

3 School of Computer Science, The University of Adelaide, Adelaide, Australia

Published online: 10 May 2018

1 Introduction

Evolutionary computation has been widely used in practice to solve problems that arise in various domains in engineering and economics. To understand evolutionary computing techniques also on a theoretical basis, rigorous runtime analysis has become a major approach over the last 20 years [1,10,17]. In this area the class of linear pseudo-Boolean functions plays a crucial role. The first such analysis aimed at the classical (1+1) evolutionary algorithm, (1+1) EA, on the simplest linear function OneMax (maximize the number of 1-bits in a bit string) [15]. Since then the runtime of the (1+1) EA on the class of all linear functions has become a hot issue, and various proof techniques have been developed. While inferring the asymptotic runtime, a deep understanding of the internal mechanisms of the (1+1) EA has been gained. This has led to the famous $\Theta(n \log n)$ bound [6]. More detailed studies later pushed the insights even further and revealed the leading constants [5,12,19].

We are now interested in the behaviors of nature-inspired algorithms on constrained problems. That means, only search points satisfying a certain condition are considered as solutions to the problem. Various constraint handling methods for such algorithms are discussed in the literature (see the survey [14]). The analysis of the minimum spanning tree problem and minimum vertex covers suggests that good runtime bounds can be achieved by implementing the constraints as an additional objective in a then multi-objective fitness function [7,13,16].

In this article, we follow a similar approach. We examine the runtimes of several evolutionary algorithms on linear functions under constraints. It is known that the most general setting of linear constraints is equivalent to the NP-hard Knapsack problem [11]. Even on instances of the Knapsack problem that can be optimized using simple greedy heuristics, the runtime of the (1+1) EA was shown to be exponential [20]. Therefore, we restrict our attention to the subclass of uniform constraints, which restrict the Hamming weight of a feasible solution. Recently Friedrich et al. [8] investigated linear functions under uniform constraint in a static setting. There, a fixed bound B on the number of 1-bits in a feasible string is given. They studied the optimization behavior of the (1+1) EA starting from scratch. We extend these studies to a dynamic setting, where the constraint bound changes from B to some new value B^*. Our goal is to analyze the number of fitness evaluations that an evolutionary algorithm needs to *reoptimize* a solution that is optimal with respect to the old bound, into an optimal solution observing the new one. The main focus of this work is to give guarantees on the maximum reoptimization runtimes. These guarantees depend on the algorithm, the problem size, the constraint bounds as well as the extent of the change. A summary of the main results can be found in Table 1.

After fixing some notations and definitions in the preliminary Sect. 2, we start the analysis with the (1+1) EA in Sect. 3. The elitist selection mechanism and single-objective fitness enables it to reoptimize OneMax quickly. Conversely, it is very hard to make progress on general linear objective once the new cardinality bound B^* has been reached. Any improvement from there requires the (1+1) EA to "swap" certain bits, resulting in a long waiting time. To put the upper bound into perspective, we also prove that for certain settings the runtime's quadratic dependence on the problem size cannot be avoided. Aiming at the mentioned swap operation, we introduce the

Table 1 Overview of results

Profit function	(1+1) EA	MOEA	MOEA-S	MO $(\mu+(\lambda,\lambda))$ GA	
OneMax	$\mathrm{O}\Big(n\log\Big(\frac{n-B}{n-B^*}\Big)\Big)$	$\mathrm{O}\Big(nD\log\Big(\frac{n-B}{n-B^*}\Big)\Big)$	$\mathrm{O}\Big(n\log\Big(\frac{n-B}{n-B^*}\Big)\Big)$	$\mathrm{O}\Big(\min\{\sqrt{nD^3}, D^2\sqrt{\frac{n}{n-B^*}}\}\Big)$	if $B \le B^*$
	$\mathrm{O}\big(n\log\big(\frac{B}{B^*}\big)\big)$	$\mathrm{O}\big(nD\log\big(\frac{B}{B^*}\big)\big)$	$\mathrm{O}\big(n\log\big(\frac{B}{B^*}\big)\big)$	$\mathrm{O}\Big(\min\{\sqrt{nD^3}, D^2\sqrt{\frac{n}{B^*}}\}\Big)$	if $B > B^*$
Linear function	$\mathrm{O}\big(n^2\log(B^* w_{\max})\big)$	$\mathrm{O}(nD\log D)$	$\mathrm{O}(n\log D)$	$\mathrm{O}(nD)$	

Upper bounds on the expected reoptimization times of the (1+1) EA, the Multi-Objective Evolutionary Algorithm (MOEA), its variant with single bit flip (MOEA-S) and the multi-objective $(\mu+(\lambda,\lambda))$ Genetic Algorithm (MO $(\mu+(\lambda,\lambda))$ GA) on linear functions of length-n bit strings under dynamic uniform constraint. B denotes the old and B^* the new cardinality bound, $D = |B^* - B|$ their difference. Runtimes of the form $\mathrm{O}(n\log(B/B^*))$ are to be read as $\mathrm{O}(n\log B)$ if $B^* = 0$. For comparison, the (1+1) EA needs $\Omega(n)$ iterations to optimize OneMax under uniform constraint from scratch in the static setting (if B is not too close to 0, n or $n/2$) and $\Omega(n^2)$ for general linear profit functions [8]

Multi-Objective Evolutionary Algorithm, MOEA, in Sect. 4. This population-based EA stores individuals based on their distance to the new constraint. Namely, it maintains a solution for each Hamming weight between B and B^*. This way, the MOEA can forgo the swaps. However, the size of the population slows down the reoptimization process. To tackle this problem, an improved variant of the MOEA with single bit flip—the MOEA-S—is presented. It achieves a behavior similar to the MOEA with a population of only two individuals. The interplay of the two solutions is able to emulate the swaps, resulting in a quasi-linear waiting time for a fitness improvement. Finally, in Sect. 5, we examine the performance of a multi-objective variant of the $(1+(\lambda, \lambda))$ Genetic Algorithm, $(1+(\lambda, \lambda))$ GA, of Doerr et al. [4]. In contrast to many other GAs, the $(1+(\lambda, \lambda))$ GA does not use the crossover of candidate individuals to recombine good parts of solutions, but instead to repair malicious mutations. It has been shown that the $(1+(\lambda, \lambda))$ GA can optimize OneMax in linear time, when using an adaptive parameter setting [3]. We incorporate these ideas into our multi-objective setting with the Multi-Objective $(\mu+(\lambda, \lambda))$ Genetic Algorithm, MO $(\mu+(\lambda, \lambda))$ GA. Slightly at odds with the usual notation, the usage of μ in the name shall indicate that its population size may be larger than 1; however, this size is not fixed to a predefined value, instead it depends on the difference $|B - B^*|$.

This article's results improve upon the conference version [18]. The bound on the reoptimization time of the MOEA on general linear functions has been reduced from $\mathrm{O}(nD^2)$ to $\mathrm{O}(nD \log D)$. For the MO $(\mu+(\lambda, \lambda))$ GA on OneMax, it is improved to $\mathrm{O}(\min\{\sqrt{n}\sqrt{D^3},\ D^2\sqrt{n/(n-B^*)}\})$ if $B \leq B^*$, and to $\mathrm{O}(\min\{\sqrt{n}\sqrt{D^3},\ D^2\sqrt{n/B^*}\})$ if $B > B^*$. We are also able to prove that the runtime of the same algorithm on linear profit functions is of order $\mathrm{O}(nD)$, instead of $\mathrm{O}(nD^2)$.

2 Preliminaries

2.1 Setting

In the paper, the behaviors of several evolutionary computing techniques on linear optimization problems with dynamic constraints are considered. The collection $\{0, 1\}^n$ of all bit strings with fixed length n serves as the search space. Given a sequence $\{w_1, w_2, \ldots, w_n\}$ of positive real weights, the *profit* of a search point $x = x_1 x_2 \ldots x_n \in \{0, 1\}^n$ is defined as

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

As a technical detail, we assume all w_i to be at least 1. Variable $w_{\max} = \max_i w_i$ denotes the maximum weight. The simplest linear profit function is

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

We would like to point out that throughout this work we distinguish between the profit and the fitness of an individual. The fitness function f is an implementation detail of the algorithm in use (cf. Sect. 2.2) while all algorithms aim to maximize the profit P.

The constrained aspect of the optimization problem is modeled by declaring a subset of the search space as the *feasible region*. Only feasible search points are treated as solutions to the problem. That is, all search points in the *infeasible region* are dismissed, even if their profit may be higher than that of any feasible solution. An *optimal solution* is then a feasible search point of maximum profit. It has been shown that the general class of "linear constraints" leads to exponential runtimes for many evolutionary algorithms even on simple problem instances [8,20]. We therefore content ourselves with *uniform constraints*, restricting the Hamming weight, i.e., the number of 1-bits, in a feasible solution. Let $|x|_1$ denote the Hamming weight of x, and conversely $|x|_0 = n - |x|_1$ the number of 0-bits in x. Let $0 \leq B \leq n$ be a non-negative integer—the *cardinality bound*. The general optimization problem of linear functions under uniform constraint is then given as

$$\begin{gathered} \max\ P(x) \\ \text{s.t.} |x|_1 \leq B. \end{gathered}$$

However, solving this problem is not our primary goal. Instead, we fix a second integer $0 \leq B^* \leq n$ and investigate the number of fitness evaluations until an algorithm samples an optimal solution to the problem with respect to the *new* cardinality bound B^* for the first time, starting from an optimal solution x_{orig} with respect to the *original* bound B. We refer to this setting as profit function P being under *dynamic uniform constraint*. The number of fitness evaluations needed is the *reoptimization time*, usually symbol T is used to denote this random variable. We call its expectation $E[T]$ the *expected reoptimization time*.

For each algorithm, we distinguish four cases in the analysis. We give a bound on the expected reoptimization time on general linear functions as well as on OneMax. Independently, the new bound B^* can be at least as large as B or strictly smaller. All runtime bounds are given in terms of the problem size n, the two cardinality bounds B, B^*, and their absolute difference $D = |B^* - B|$. Some asymptotic estimates display quotients of which the numerator or the denominator can become zero in extreme cases. If so, that quotient is to be understood as a constant strictly larger than 1.

2.2 Algorithms

Four nature-inspired algorithms are considered in the paper. Namely, the (1+1) *Evolutionary Algorithm*, the *Multi-Objective Evolutionary Algorithm*, its single bit flip variant, and the *Multi-Objective* $(\mu+(\lambda, \lambda))$ *Genetic Algorithm*. These evolutionary computing methods adopt different constraint handling strategies. They are reflected in the respective *fitness functions* and, in the case of multi-objective fitness, the notion of *dominance*.

We let the (1+1) EA (Algorithm 1) use the following fitness function $f_{(1+1)}$ suggested by Friedrich et al. in [8]. This single-objective function is defined as

$$f_{(1+1)}(x) = P(x) - (n\,w_{\max} + 1) \cdot \max\{0,\ |x|_1 - B^*\}.$$

This choice has two immediate consequences. First, the large penalty term $(n\,w_{\max} + 1) \cdot \max\{0, |x|_1 - B^*\}$ scales with the extent of the constraint violation. It thus guides the search of the algorithm towards the feasible region (given by the new constraint bound B^*). Second, once the algorithm samples the first feasible solution, its elitist selection bars it from ever again adopting an infeasible search point.

```
Algorithm 1: (1+1) EA
1 $x \leftarrow x_{\mathrm{orig}}$;
2 while stopping criterion not met do
3     $y \leftarrow$ flip each bit of $x$ independently with probability $1/n$;
4     if $f_{(1+1)}(y) \geq f_{(1+1)}(x)$ then
5         $x \leftarrow y$;
```

The Multi-Objective Evolutionary Algorithm (MOEA; see Algorithm 2) and its variant MOEA-S (Algorithm 3) use the same vector-valued fitness function $f_{\mathrm{MOEA}}(x) = (|x|_1, P(x))$, but different dominance relations between solutions $y, z \in \{0,1\}^n$. In the context of the MOEA, y *dominates* z whenever y provides at least the same profit as z with the same number of 1-bits,

$$y \succcurlyeq_{\mathrm{MOEA}} z \;\Leftrightarrow\; |y|_1 = |z|_1 \,\wedge\, P(y) \geq P(z).$$

This defines a preorder on the search space in which two strings are comparable if and only if they have the same Hamming weight. We say y *strongly dominates* z if y dominates z and has a strictly larger profit, written $y \succ_{\mathrm{MOEA}} z$. Note that two strings having the same weight and profit are indistinguishable with respect to $\succcurlyeq_{\mathrm{MOEA}}$.

```
Algorithm 2: MOEA; Assuming $B \leq B^*$.
1 $S \leftarrow \{x_{\mathrm{orig}}\}$;
2 while stopping criterion not met do
3     Choose $x \in S$ uniformly at random;
4     $y \leftarrow$ flip each bit in $x$ independently with probability $1/n$;
5     if $(B^* \geq |y|_1 \geq B) \wedge (\nexists w \in S\colon w \succ_{\mathrm{MOEA}} y)$ then
6         $S \leftarrow (S \cup \{y\}) \setminus \{z \in S \mid y \succcurlyeq_{\mathrm{MOEA}} z\}$;
```

The population set S of the MOEA is initialized with some fixed optimal solution x_{orig} which has exactly B bits set to 1. An offspring, denoted by y, is obtained via standard bit mutation of a string drawn from S uniformly at random. The algorithm checks whether y has Hamming weight between B and B^*, and is not already strongly dominated by another individual in S. If so, y is included in S and all solutions that are dominated by the new offspring are discarded (excluding y itself, of course). All individuals in S are thus pairwise incomparable. Note that the population can increase up to size $|B^* - B| + 1 = D + 1$.

The MOEA-S variant exhibits two major changes compared to the original MOEA. The first distinction is that the MOEA-S uses the single-bit flip operator, which is usually employed in randomized local search. The other one is the notion of dominance between solutions, written $\succcurlyeq_{\text{MOEA-S}}$. As mentioned before, $\succcurlyeq_{\text{MOEA-S}}$ uses the same fitness function f_{MOEA}, but the resulting orderings on the search space differ. Assume $B \leq B^*$ for the moment. Strings y and z such that *at most one* of the values $|y|_1$ and $|z|_1$ equal B^* or $B^* - 1$ are ordered lexicographically,

$$y \succcurlyeq_{\text{MOEA-S}} z \;\Leftrightarrow\; (|y|_1 > |z|_1) \vee (|y|_1 = |z|_1 \wedge P(y) \geq P(z)). \tag{1}$$

If *both* $|y|_1$ and $|z|_1$ are close to the new boundary B^*, i.e., if $|y|_1, |z|_1 \in \{B^*, B^* - 1\}$, we set

$$y \succcurlyeq_{\text{MOEA-S}} z \;\Leftrightarrow\; |y|_1 = |z|_1 \wedge P(y) \geq P(z). \tag{2}$$

As a result, two bit strings y and z are incomparable if and only if $|y|_1 = B^*$ and $|z|_1 = B^* - 1$ or vice versa, written $y \parallel_{\text{MOEA-S}} z$. Similar to the MOEA, the population S of the MOEA-S collects incomparable solutions during the reoptimization process, but now can have at most 2 elements at any given time. In case of $B > B^*$, we invert the dependency on the number of 1-bits in part (1) of the dominance definition to $|y|_1 < |z|_1$ since we now prefer solutions with smaller Hamming weight. Additionally, part (2) of the definition applies if $|y|_1, |z|_1 \in \{B^*, B^* + 1\}$ since we only care for solutions between the two bounds.

Algorithm 3: MOEA-S; assuming $B \leq B^*$.

```
1 $S \leftarrow \{x_{\text{orig}}\}$;
2 while stopping criterion not met do
3     Choose $x \in S$ uniformly at random;
4     $y \leftarrow$ flip bit $x_i$ with $i \in \{0, \dots, n\}$ chosen uniformly at random;
5     if $\forall z \in S\colon y \parallel_{\text{MOEA-S}} z$ then
6         $S \leftarrow S \cup \{y\}$;
7     if $(B^* \geq |y|_1 \geq B) \wedge (\exists z \in S\colon y \succcurlyeq_{\text{MOEA-S}} z)$ then
8         $z \leftarrow y$;
```

The MO (μ+(λ, λ)) GA (Algorithm 4) combines features of the (1+(λ, λ)) GA [4] and the MOEA. Every iteration of the while-loop of the algorithm has three phases: mutation, crossover, and selection. During the mutation phase the algorithm first samples a search point $x \in S$ from the population uniformly and a number $0 \leq \ell \leq n$ (called *step size* in [4]) according to the binomial distribution $\mathrm{Bin}(n, p)$ with parameters n and p, the *mutation probability*. Then, λ mutants are generated by the operator $\mathtt{mutate}_\ell(x)$ which flips exactly ℓ bits in x chosen uniformly at random. This results in λ ℓ-Hamming neighbors of x chosen uniformly. Note that a mutation probability p greater than $1/n$ puts larger emphasis on the exploration aspect of the search, as compared to standard bit mutation.

First consider the case $B \leq B^*$. Among the ℓ-neighbors, those with Hamming weight strictly larger than $|x|_1$ would be preferred for crossover, if there are any. The flipping bits are chosen uniformly, hence we are not guaranteed such a neighbor. Notwithstanding, any flipping 0-bit potentially provides valuable information about a possible fitness improvement. Thus, we lower the requirement of an offspring to enter the crossover phase, only demanding that any 0-bit has been flipped in its creation. If $B > B^*$, the same idea applies to the 1-bits. This gives rise to the following definition. We call a mutant $z \in \{0, 1\}^n$ of x *valid* if $|x|_1 \leq B^*$ and some 0-bit was flipped by the mutation operator during its creation, or $|x|_1 \geq B^*$ and at least one 1-bit flipped. At the end of the mutation phase, some valid mutant x' is chosen at random for further processing; if there are none, we set $x' = x$.

The crossover operator $\mathtt{cross}_c(x, x')$ recombines the parent string x with the designated mutant x', aiming to repair potentially malicious mutations. Given some fixed *crossover probability* c, $\mathtt{cross}_c(x, x')$ creates a bit string by choosing, in every position $1 \leq i \leq n$, bit x'_i with probability c, and x_i otherwise. The operator is applied λ times to the same pair x, x'. If $B \leq B^*$, we are only interested in the offspring whose Hamming weights are exactly one larger than that of the parent string. The intuition is to only cautiously grow the population to avoid the connected slowdown in the early phases of the optimization. Consequently, the algorithm collects in set M those offspring whose Hamming weight equal $|x|_1 + 1$, provided that $B \leq B^*$ ($|x|_1 - 1$, if $B > B^*$). To rank the offspring in M, we use the same notion of dominance as that for the MOEA, i.e., $y \succcurlyeq_{\mathrm{MOEA}} z \Leftrightarrow |y|_1 = |z|_1 \wedge P(y) \geq P(z)$. At the end of the crossover phase, the algorithm draws some $\succcurlyeq_{\mathrm{MOEA}}$-maximal element $y' \in M$ to enter the selection phase. If $M = \emptyset$, it continues with $y' = x$. Observe that if the mutation phase fails to produce a valid mutant, all offspring are identical copies of x and the MO (μ+(λ, λ)) GA does not find any improvement in this round.

The selection phase finally checks whether the solution y' obtained in the earlier phases meets the cardinality constraint and is not strongly dominated by a solution previously in S with respect to $\succ_{\mathrm{MOEA}}$. If so, the population is updated in the usual way.

Algorithm 4: MO (μ+(λ, λ)) GA; assuming $B \leq B^*$. Concept from [4].

```
1  $S \leftarrow \{x_{\text{orig}}\}$;
2  while stopping criterion not met do
       /* Mutation phase.                                */
3      Choose $x \in S$ uniformly at random;
4      Choose $\ell$ according to $\text{Bin}(n, p)$;
5      for $i = 1$ to $\lambda$ do
6          $x^{(i)} \leftarrow \texttt{mutate}_\ell(x)$;
7      $V = \{x^{(i)} \mid x^{(i)} \text{ is valid}\}$;
8      if $V \neq \emptyset$ then
9          Choose $x' \in V$ uniformly at random;
10     else $x' \leftarrow x$;

       /* Crossover phase.                               */
11     for $i = 1$ to $\lambda$ do
12         $y^{(i)} \leftarrow \texttt{cross}_c(x, x')$;
13     $M = \{y^{(i)} \mid |y^{(i)}|_1 = |x|_1 + 1\}$ ;
14     if $M \neq \emptyset$ then
15         Choose a $\succcurlyeq_{\text{MOEA}}$-maximal $y' \in M$ uniformly at
           random;
16     else $y' \leftarrow x$;

       /* Selection phase.                               */
17     if $(B^* \geq |y'|_1 \geq B) \wedge (\nexists w \in S\colon w \succ_{\text{MOEA}} y')$ then
18         $S \leftarrow (S \cup \{y'\}) \setminus \{z \in S \mid y' \succcurlyeq_{\text{MOEA}} z\}$;
```

2.3 Tools

Drift analysis is our main tool to analyze the expected reoptimization times, cp. [5,9]. The inner state of the algorithm in question is mapped to a real number via a *potential function*. The change of the potential during the optimization process is interpreted as a random process (of the random decisions of the algorithm). The idea is to prove the existence of a *drift*, expected movement, towards a potential value corresponding to an optimal solution. The time needed to reach that value then equals the reoptimization time.

We primarily employ the Multiplicative Drift Theorem as introduced by Doerr et al. [5].

Theorem 1 (Multiplicative Drift Theorem [5]) *Let $S \subseteq \mathbb{R}^+$ be a finite set of positive numbers with minimum $s_{\min}$. $(X^{(t)})_{t\in\mathbb{N}}$ shall denote 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 real number $\delta > 0$ such that*

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

holds for all $s \in S$ with $\Pr[X^{(t)} = s] > 0$. Then, for all $s_0 \in S$ with $\Pr[X^{(t)} = s_0] > 0$, we have

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

As a technical detail, the target potential in our analysis may not exactly be 0, just any value smaller than a given $s_{\min} > 0$. It is obvious that Multiplicative Drift Theorem is still applicable in this case.

Additionally we use the Additive Drift Theorem by He and Yao [9], in particular its statement regarding lower bounds. The difference being whether the drift is a multiple of the current potential or constant. Again we choose a more flexible formulation, shooting for a target value $s_{\min}$ which might be different from 0.

Theorem 2 (Additive Drift Theorem [9]) *Let $S \subseteq \mathbb{R}^+$, $s_{\min}$, and $(X^{(t)})_{t\in\mathbb{N}}$ be as above. Let T denote the first point in time for which $X^{(t)} = s_{\min}$. Suppose that there exists a real number $\delta > 0$ such that*

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

holds for all $s \in S$ with $\Pr[X^{(t)} = s] > 0$. Then, for all $s_0 \in S$ with $\Pr[X^{(t)} = s_0] > 0$, we have

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

Conversely, if $E[X^{(t)} - X^{(t+1)} \mid X^{(t)} = s] \leq \delta$ for all such s, then $E[T \mid X^{(0)} = s_0] \geq (s_0 - s_{\min})/\delta$.

3 Analysis of the (1+1) EA

The runtime of the classical (1+1) EA on linear functions has been extensively studied in the literature, see e.g. [6,10,17]. In particular, a detailed discussion of its performance on constrained problems can be found in [8] and [20]. The ideas presented in this section are very similar to those used by Friedrich et al. [8] analyzing the

(1+1) EA under static constraints. We show that the same techniques are applicable in the dynamic setting. Recall that the fitness function $f_{(1+1)}$ guides the search of the algorithm towards the feasible region and only afterwards maximizes the profit.

Theorem 3 *The reoptimization time of the* (1+1) EA *on* OneMax *under dynamic uniform constraint is*

$$E[T] = \begin{cases} \mathrm{O}\left(n \log\left(\frac{n-B}{n-B^*}\right)\right), & \text{if } B \le B^*; \\ \mathrm{O}\left(n \log\left(\frac{B}{B^*}\right)\right), & \text{if } B > B^*. \end{cases}$$

Proof It is convenient for the analysis to use different potential functions depending on whether the bit string currently maintained by the (1+1) EA is feasible or not. In the former case we use $|x|_0$, and $|x|_1$ in the latter. We always aim to reduce the potential. Note that the target potential of an optimal string is not zero but $|x^*|_0 = n - B^*$ if $B \le B^*$ or $|x^*|_1 = B^*$ if $B > B^*$. If the current solution is feasible but non-optimal, flipping a single 0-bit (and nothing else) improves on the profit P and is thus accepted as a fitness-improving move. Conversely, the same holds for flipping a single 1-bit of an infeasible string, as this eases the constraint violation. A standard argument now shows that there is an expected drift in the potential of at least $|x|_0/en$ or $|x|_1/en$, respectively.

If $B \le B^*$, the initial solution x_{orig} is feasible and every infeasible solution would be discarded due to a negative fitness value. The Multiplicative Drift Theorem (Theorem 1) now gives an expected number of $\mathrm{O}(n \log(|x_{\text{orig}}|_0/|x^*|_0)) = \mathrm{O}(n \log((n - B)/(n - B^*)))$ generations to reach any optimal solution to the new problem with bound B^*. As the (1+1) EA uses a constant number of fitness evaluations per iteration, this is also an upper bound on the reoptimization time.

If $B > B^*$, the reasoning is similar but with the reoptimization starting in the infeasible region. The time needed to sample the first feasible solution, say z, is of order $\mathrm{O}(n \log(|x_{\text{orig}}|_1/B^*)) = \mathrm{O}(n \log(B/B^*))$. However, z itself does not need to be optimal. In the remainder of this proof, we show that the additional time needed to find the optimum does not worsen the asymptotic runtime. Namely, we claim that the (1+1) EA starting from z reaches an optimal solution x^* within $\mathrm{O}(n)$ iterations.

Let $U = B^* - |z|_1$ denote the random number of 1-bits by which we missed the optimality bound B^*. When z is sampled for the first time, it must originate from an infeasible solution. This can only happen if at least $U + 1$ bits flipped simultaneously. The probability of such a mutation is at most $\binom{n}{U+1}\frac{1}{n^{U+1}}$. As mentioned above, once we are in the feasible region there is an expected drift of $|x|_0/en$. This implies, again via Theorem 1, that the expected time to reach optimality conditional on U is bounded by

$$en \ln\left(\frac{|z|_0}{|x^*|_0}\right) = en \ln\left(\frac{n - B^* + U}{n - B^*}\right) = en \ln\left(1 + \frac{U}{n - B^*}\right) \le en \frac{U}{n - B^*} \le enU.$$

Here we use the estimate $\ln(1 + x) \le x$, which can easily be seen from the Taylor expansion. By the law of total expectation, the expected number of generations to reach any optimal solution x^* from z is thus at most

$$\sum_{u=0}^{B^*} \binom{n}{u+1} \frac{enu}{n^{u+1}} \le \sum_{u=0}^{B^*} \left(\frac{en}{u+1}\right)^{u+1} \frac{en(u+1)}{n^{u+1}}$$
$$= e^2 n \sum_{u=0}^{B^*} \left(\frac{e}{u+1}\right)^u \le e^2 n \sum_{u=0}^{\infty} \left(\frac{e}{u+1}\right)^u \le 4e^2 n.$$

□

We have seen in the proof of Theorem 3 that the drift of the two potential functions for OneMax are at least of order $\Omega(|x|_0/n)$ and $\Omega(|x|_1/n)$, in the respective regions. The following lemma by Witt [19] states that they are also at most of this order. We use this to give tight bounds on the expected reoptimization time for some values of B and B^*. Recall that $D = |B^* - B|$ denotes the absolute distance between the two cardinality bounds.

Lemma 4 (Lemma 6.7 in [19]) *Consider the* (1+1) EA *(with standard bit mutation) on* OneMax *as a maximization problem. Let x' denote the offspring of some parent string $x \in \{0, 1\}^n$ after selection. Then,*

$$E[|x|_0 - |x'|_0] \le \frac{|x|_0}{n} \left(1 - \frac{1}{n} + \frac{|x|_0}{n^2} \left(1 - \frac{1}{n}\right)^{n-|x|_0}\right) \le \frac{|x|_0}{n}.$$

Similarly, when considering the minimization variant, we have $E[|x|_1 - |x'|_1] \le |x|_1/n$.

Theorem 5 *Suppose either $B \le B^* = \varepsilon n$ or $B > B^* = \varepsilon n$ holds, where $0 < \varepsilon < 1$ is an arbitrary constant. Then, the expected reoptimization time of the* (1+1) EA *on* OneMax *under dynamic uniform constraint is* $\Theta(D)$

Proof The condition in the theorem states that during the whole reoptimization there is always a linear number of 0-bits present (if $B \le B^*$; otherwise, a linear number of 1-bits). By the discussion above as well as Lemma 4, there is a constant drift towards the new optimum, regardless of whether the current string is feasible or not. The Additive Drift Theorem (Theorem 2) now implies the claim. □

The famous tight bound $\Theta(n \log n)$ on the runtime of the classical (1+1) EA on OneMax indicates that the tight bound on the (1+1) EA (Algorithm 1) on OneMax under dynamic uniform constraint with $B = 0$ and $B^* = n$, is also $\Theta(D \log D)$. Combing the above theorem, it is not hard to see that the tight or lower bound on the runtime of the (1+1) EA on OneMax under dynamic uniform constraint is closely dependent on the relationship between B, B^*, and n. Thus we only discuss the tight or lower bound in some specific cases in the paper.

We now discuss the general case of reoptimizing a linear profit function. As compared to OneMax, it is possible to reduce the Hamming weight $|x|_1$ of a solution while at the same time improving on the profit $P(x)$. The constraint still only restricts the former. Symbol $w_{\max}$ stands for the maximum weight of the function P.

The proof of this theorem follows closely the one of Theorem 6 in [8].

Theorem 6 *The expected reoptimization time of the* (1+1) EA *on a linear profit function P under dynamic uniform constraint is*

$$E[T] = \mathrm{O}(n^2 \log(B^* w_{\max})).$$

Proof By Theorem 3, the claimed running time dominates the time to reach the feasible region. Consequently, we start the analysis with a feasible bit string. W.l.o.g. the weights of the profit function P are in descending order, i.e. $w_{\max} = w_1 \geq w_2 \geq \cdots \geq w_n$. Note that there might be multiple optimal solutions if the weights are not all distinct. To give an upper bound on the reoptimization time, we measure the time the (1+1) EA needs to sample $x^* = 1^{B^*}0^{n-B^*}$. To that end, let loss and surplus be two auxiliary functions defined as follows:

$$\mathrm{loss}(x) = \sum_{i=1}^{B^*} w_i \overline{x_i}; \quad \mathrm{surplus}(x) = \sum_{i=B^*+1}^{n} w_i x_i,$$

where $\overline{x_i}$ denotes the negation of bit x_i. Intuitively, loss(x) counts the bits in which x and the target x^* differ up to position B^* and weighted by the w_i. Similarly, surplus(x) counts the weighted dissimilarities starting from index $B^* + 1$. Let the potential of the solution x be $g(x) = \mathrm{loss}(x) - \mathrm{surplus}(x)$. Observe that for any feasible solution the potential is non-negative, it cannot increase during the optimization and is zero precisely for solution x^*.

The expected drift of g is minimal if solution x has Hamming weight B^*. Then, no single-bit flip can improve on the potential as it would be rejected by the (1+1) EA. Flipping a 0-bit violates the constraint and flipping a 1 would decrease the profit. If $|x|_1 = B^*$, the number of *missing* 1-bits up to position B^* equals the number of total 1s between index $B^* + 1$ and n. Let k denote this number. Consider the event $\mathcal{E}$ of flipping exactly one 0-bit in substring $x_1 x_2 \ldots x_{B^*}$, one 1 in $x_{B^*+1} \ldots x_n$ and nothing else. Conditional on $\mathcal{E}$, each of the k^2 mutations are equally likely and the average decrease in potential is loss$(x)/k$ for the flipping 0 and $-$surplus$(x)/k$ for the 1. The total expected drift of g can thus be bounded by

$$\begin{aligned} E[g(x) - g(x')] &\geq E[g(x) - g(x') \mid \mathscr{E}, |x|_1 = B^*]\Pr[\mathscr{E}] \\ &= \frac{\mathrm{loss}(x) - \mathrm{surplus}(x)}{k} \frac{k^2}{n^2} \left(1 - \frac{1}{n}\right)^{n-2} \geq \frac{g(x)}{en^2} \end{aligned}$$

The observation that the potential of any feasible solution is at most $\sum_{i=1}^{B^*} w_i \leq B^* w_{\max}$ and that the expected time to reduce this to zero is bounded by $en^2 \ln(B^* w_{\max})$ (by Theorem 1) completes the proof. $\square$

If $B \leq B^*$, we can derive a slightly better bound since we know the initial potential. W.l.o.g. we start with the initial solution $x_{\mathrm{orig}} = 1^B 0^{n-B}$. Its potential is $g(x_{\mathrm{orig}}) =$

$P(x^*) - P(x_{\text{orig}}) = \sum_{i=B+1}^{B^*} w_i \leq D w_{\max}$. This results in a runtime bound of $O(n^2 \log(D w_{\max}))$.

The reoptimization time of the (1+1) EA on general linear functions seems to be unreasonably large when compared to the $O(n \log n)$ bound [6] for the unconstrained setting. The reason is a different behavior at the cardinality bound B^*. In order to make progress there, it is necessary to swap the bits in the right place immediately instead of interleaving profit-increasing moves with those losing expendable 1-bits. To back up this assessment, the next result shows that there are instances on which a quadratic runtime cannot be avoided.

Theorem 7 *There is a linear profit function P and bounds B, B^* such that the reoptimization time of the* (1+1) EA *on P under dynamic uniform constraint is $\Omega(n^2)$, not only in expectation but even with high probability.*[1]

Proof Let $0 < \varepsilon < 1$ be a constant. We set $B = \varepsilon n$ and $B^* = B + 1$ as the constraints and choose the profit function

$$P(x) = \sum_{i=1}^{\varepsilon n} 3x_i + \frac{3}{2} x_{\varepsilon n+1} + \sum_{i=\varepsilon n+2}^{n} x_i.$$

The (unique) optimal solution with respect to the old bound B is thus $x_{\text{orig}} = 1^{\varepsilon n} 0^{(1-\varepsilon)n}$.

Starting from this solution, a mutation is accepted if and only if exactly one 0-bit is flipped or no bits at all. To see this, let c_1 denote the number of flipping 1-bits and c_0 that of the 0-bits. If $c_0 > c_1 + 1$, the offspring is discarded for violating the constraint $B^* = B + 1$. If $c_0 \leq c_1 + 1$ and not both values are 0, the best available mutation involves the bit at position $\varepsilon n + 1$ turning into a 1. The change in profit is then at most

$$P(x') - P(x) \leq -3c_1 + \frac{3}{2} + (c_0 - 1) \leq \frac{3}{2} - 2c_1.$$

The right-most member of the inequality is non-negative iff $c_1 = 0$.

The only optimal mutation is flipping $x_{\varepsilon n+1}$ and nothing else. Hitting this particular bit has probability

$$\frac{1}{e(1-\varepsilon)n} \leq \frac{1}{e\sqrt{n}}.$$

Hence, with high probability, the MOEA finds a non-optimal string of Hamming weight B^* first. The only improving move is then to flip the additional 1-bit and the 0 at position $\varepsilon n + 1$ simultaneously. This swap has quadratic waiting time, which implies the claim. □

[1] We use the term *with high probability* if there exists a constant $c > 0$ such that the probability is at least $1 - 1/n^c$.

4 Analysis of the MOEA Variants

In this section, we analyze the reoptimization times of the Multi-Objective Evolutionary Algorithm (MOEA) and its single-bit flip variant MOEA-S. As shown in Theorem 7, the (1+1) EA on general linear functions has to "swap" two specific bits for an improvement, once a non-optimal string with Hamming weight B^* has been found. To overcome this limitation and soften the effect of the new cardinality bound, the MOEA maintains a pool S of candidate solutions. The idea is to utilize multiple solutions in order to return to the incremental improvements typical for the unconstrained setting. On the other hand, a large population may slow down the reoptimization process. Recall that S contains a solution for each Hamming weight between B and B^* that occurred during the reoptimization process, which can be up to $D+1$. We first prove the adverse influence of the population size when working on very simple functions such as OneMax.

Theorem 8 *The expected reoptimization time of the* MOEA *on* OneMax *under dynamic uniform constraint is*

$$E[T] = \begin{cases} \mathrm{O}\left(nD\log\left(\frac{n-B}{n-B^*}\right)\right), & \text{if } B \leq B^*; \\ \mathrm{O}\left(nD\log\left(\frac{B}{B^*}\right)\right), & \text{if } B > B^*. \end{cases}$$

Proof First, note that the selection step in the MOEA (Algorithm 2) can be computed with a constant number of fitness evaluations per iteration when storing the vector $f_{\mathrm{MOEA}}(x)$ for each individual $x \in S$. It is thus enough to bound the expected number of generations needed for the reoptimization process.

We start with the discussion of case $B \leq B^*$. In order to employ drift analysis, we define the potential of the MOEA as $M = \min_{x \in S} |x|_0$. By choosing the (unique) member of S with the minimum number of 0-bits, M, and flipping exactly one of them, the potential is decreased by 1. Note that the value of M can never increase during the reoptimization process. Let M' denote the potential after one iteration starting from M. With the above argument we get a lower bound on the expected drift of

$$E[M - M'] \geq \frac{1}{|S|} \cdot \frac{M}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{M}{en(D+1)}.$$

The potential of the MOEA has an initial value $n - B$ for the initial population $\{x_{\mathrm{orig}}\}$, and is reduced down to $n - B^*$ once a solution x^* with $|x^*|_0 = n - B^*$ is included into the population. For the whole reoptimization process, the Multiplicative Drift Theorem implies that the MOEA needs an expected number of $\mathrm{O}(nD\log(\frac{n-B}{n-B^*}))$ generations.

In the case of $B > B^*$, the potential of the MOEA is defined as $M = \min_{x \in S} |x|_1$. The reasoning is the same as that for the case $B \leq B^*$ with the roles of 1-bits and 0-bits inverted. Note that we cannot undershoot the bound B^* as such solution would be outright rejected by the algorithm. □

The expected reoptimization time of the MOEA on OneMax under dynamic uniform constraint suffers from the potentially large population and the resulting longer waiting times. However, the next theorem shows that the population can speed up the reoptimization when running on general linear profit functions. In particular, this holds true when the difference $D = |B^* - B|$ is small.

Theorem 9 *The reoptimization time of the* MOEA *on linear functions under dynamic uniform constraint is*

$$E[T] = \mathrm{O}(nD \log D).$$

Proof Again, we present our method in the case of $B \le B^*$, the other one follows almost identically. To bound the reoptimization time we track so-called "candidates" x in the population. Given a feasible solution x, we say it is a *candidate* if there exists an optimal solution x^* such that $x_i^* = 1$ whenever $x_i = 1$. That is, we can create an optimal solution from a candidate by flipping only 0-bits. Observe that x_{orig} is a candidate of Hamming weight B, and any candidate of weight B^* is necessarily optimal. By the definition of the partial ordering $\succcurlyeq_{\mathrm{MOEA}}$, a candidate in the population S can only be replaced by another candidate with the same weight but higher profit. Let h be the maximum Hamming weight among all candidates in S and let $x^{(h)} \in S$ denote its representative. We define $G = B^* - h$ to be the potential of the MOEA. The reasoning is as follows. Among the $n - h$ 0-bits in $x^{(h)}$ there are at least $B^* - h$ (of highest weight) such that flipping them creates a candidate of higher Hamming weight.

We can now use a similar argument as in the proof of Theorem 8. Choosing $x^{(h)}$ for mutation and flipping exactly one of those designated 0s decreases the potential by one. This results in an expected drift of at least

$$\frac{1}{|S|} \cdot \frac{B^* - h}{en} \ge \frac{G}{en(D+1)}.$$

The claim now follows from Theorem 1 and the observation that the inital potential is exactly $D = B^* - |x_{\mathrm{orig}}|_1$.

For the other case in which $B > B^*$, the notion of a candidate is reversed. Solution x is a *candidate* if there is an optimum x^* such that $x_i = 0$ implies $x_i^* = 0$. Using the above reasoning on the potential $G = h - B^*$ yields the same expected reoptimization time. □

The MOEA-S variant uses the single-bit flip operator instead of the standard bit mutation. Also, in order to reduce the population size and avoid long waiting times, the dominance relation $\succcurlyeq_{\text{MOEA-S}}$ of the MOEA-S is different from $\succcurlyeq_{\mathrm{MOEA}}$. Two strings are now incomparable only if they have Hamming weights B^* and $B^* - 1$, respectively (assuming $B \le B^*$; otherwise the weights are B^* and $B^* + 1$). In the following lemma we investigate the update mechanism of the MOEA-S for solutions in S.

Lemma 10 *As long as no solution of Hamming weight B^* has been sampled by the* MOEA-S, *the population size is* 1. *If S has two members, their Hamming distance is* 1.

Proof For the first part, recall that the population is initialized with a single bit string x_{orig}. While the Hamming *weight* of the currently best solution is between B and $B^* - 1$

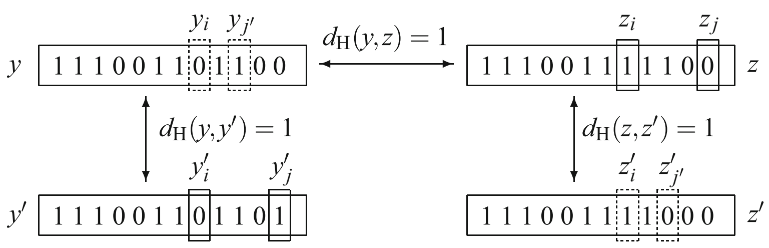

Fig. 1 An illustration of the proof of Lemma 10. Symbol d_H denotes the Hamming distance. If population $\{y, z\}$ is updated, then the new set is either $\{y, y'\}$ or $\{z, z'\}$, in which the two solutions maintain distance 1. The updates effectively swap the bits z_i, z_j or $y_i, y_{j'}$, respectively

(B and $B^* + 1$ if $B > B^*$), a mutation is accepted iff it flips a single 0-bit (1-bit). By the definition of $\succcurlyeq_{\text{MOEA-S}}$, the offspring then replaces its parent. The condition in line 5 of Algorithm 3 is not met prior to S holding a weight-B^* solution and another 0-bit flips (a 1-bit flips).

For the second part, consider the iteration in which said condition is fulfilled for the first time. Due to the single-bit flip the initial two incomparable solutions have Hamming *distance* 1. Suppose the difference is at position i. Let z denote the solution with the larger number of 1-bits and y the one with fewer 1s. Necessarily, $z_i = 1$ and $y_i = 0$. Flipping a 0 in z or a 1 in y is always discarded. Either it violates the upper bound of B^* in line 7 of the algorithm (lower bound if $B > B^*$) or it creates an offspring that is already dominated. A flip at position i transforms the two solutions into each other. There are only two ways for the population to reach a new state, see also Fig. 1. The first is to flip a 0-bit in y at index $j \neq i$. The resulting offspring y' now has Hamming weight one larger than y and is designated to replace z (if it yields at least the same profit). String y' is almost identical to z, the only difference is that $z_i = 1$ and $z_j = 0$ are now swapped. The other possible update is to replace y by flipping a 1-bit at position j' in z and thus swapping $y_i = 0$ and $y_{j'} = 1$. Both updates preserve the distance. □

Theorem 11 *The reoptimization time of the* MOEA-S *on* OneMax *under dynamic uniform constraint is*

$$E[T] = \begin{cases} O\left(n \log\left(\frac{n-B}{n-B^*}\right)\right), & \textit{if } B \leq B^*; \\ O\left(n \log\left(\frac{B}{B^*}\right)\right), & \textit{if } B > B^*. \end{cases}$$

Proof Lemma 10 shows that the MOEA-S on OneMax behaves like random local search. It is now easy to see that the expected optimization times differ from those in Theorem 3 only by a constant factor. □

Theorem 12 *The reoptimization time of the MOEA-S on linear functions under dynamic uniform constraint is*

$$E[T] = O(n \log D).$$

Proof For this proof we introduce some notation to address consecutive ranges of bits in solutions $x \in \{0,1\}^n$. Let i and j, $i \leq j$, be two indices, $x_{[i,j]} = x_i x_{i+1} \dots x_j$ stands for the substring of x from position i to j. The special cases of $x_{[1,n]} = x$ and $x_{[i,i]} = x_i$ are included. We call $x_{[1,B^*]}$ the *first block* of x and $x_{[B^*+1,n]}$ the *second block*.

W.l.o.g., the weights of the profit function P are ordered non-increasingly and we have $x_{\text{orig}} = 1^B 0^{n-B}$ as the initial solution. We pessimistically assume that the optimum $x^* = 1^{B^*} 0^{n-B^*}$ is unique, i.e., $w_{B^*} > w_{B^*+1}$. We start the analysis at the first point in time at which population S contains two search points. We claim that Algorithm 3 then needs $O(n \log D)$ generations in expectation to optimize the solution with Hamming weight B^*. Observe that this is indeed enough to establish the results as the starting point can be reached within another phase of $O(n \log D)$ iterations (Theorem 11).

It remains to prove the claim. First, assume $B \leq B^*$. Let y denote the member of S with Hamming weight $B^* - 1$ and z the one with Hamming weight B^*. We define the potential of the MOEA-S to be

$$G = 2B^* - 1 - |y_{[1,B^*]}|_1 - |z_{[1,B^*]}|_1.$$

Intuitively, it measures the number of *missing* 1-bits in the first block of the target solution z, but also considers the state of y. It is straightforward to check that the potential is non-negative and that $G = 0$ implies $|z_{[1,B^*]}|_1 = B^*$, that is, optimality. Conversely, when the MOEA-S samples an optimal solution for the first time, the potential drops to 0. To see this, recall from Lemma 10 that z can only be updated by flipping a 0-bit in y. Afterwards, the two solutions differ exactly in the previously flipping position i. If $i > B^*$, there is a bit set to 1 in the second block $z_{[B^*+1,n]}$, a contradiction to the optimality of z. Hence, $i \leq B^*$, which gives $|y_{[1,B^*]}|_1 = B^* - 1$ and $|z_{[1,B^*]}|_1 = B^*$.

We now examine the update behavior of the solutions in S with respect to the potential. To this end, we only need to consider the cases in which the position i of the defect (i.e., $y_i = 0$ and $z_i = 1$) and the new flipping position j are in different blocks. Suppose $i \leq B^* < j$, thus $w_i > w_j$. Flipping a 0-bit in z would be discarded; however, flipping $z_j = 1$ is the same as trying to swap a 1-bit into the first block of y. This is accepted since the weight difference ensures a profit gain $P(z') > P(y)$. The swap decreases the potential by 1. If string y were to be mutated, a 0-bit needs to be flipped. But this swap of a 0 into the first block of z is discarded for reducing the profit. The case $j \leq B^* < i$ is symmetric, using $w_j > w_i$.

Following the above analysis, the potential G never increases during the reoptimization. In the worst case all the 1-bits that were collected while S only had a single element, fell in the second block. Then, the number of 1-bits in the first block did not change from the initial point x_{orig} and we get $G \leq 2B^* - 1 - 2B = 2D - 1$. We now compute the expected drift in the potential. Let p denote the probability that in the current round the position i in which y and z differ is in the first block, $i \leq B^*$. If so, the potential is decreased by 1 if z is selected for mutation and a 1-bit in its *second* block is flipped. There are $|z_{[B^*+1,n]}|_1 = B^* - |z_{[1,B^*]}|_1$ many of them. If instead $i > B^*$ holds, the potential is decreased whenever one of the $B^* - |y_{[1,B^*]}|_1$ 0-bits in the *first* block of y flips. Putting it all together, we have

$$E[G - G'] = p\,\frac{B^* - |z_{[1,B^*]}|_1}{2n} + (1-p)\,\frac{B^* - |y_{[1,B^*]}|_1}{2n} \geq \frac{G}{4n}.$$

An application of the Multiplicative Drift Theorem proves the claimed bound of $\mathrm{O}(n \log D)$ for this case.

For $B > B^*$, the reasoning is almost the same. Only that now the number of 1-bits does indeed decrease during the reoptimization. In the worst case all deleted 1s are in the first block of length B^*. To reach solution z with Hamming weight $B^* + 1$, $D - 1$ successful flips are necessary; for y with $|y|_1 = B^*$, D flips are necessary. Thus, the initial potential is $G \leq 2B^* - 1 - (B^* - D) - (B^* - D - 1) = 2D$. Once the algorithm reaches the state of two solutions, the behavior is identical. □

5 Analysis of the MO $(\mu+(\lambda, \lambda))$ GA

Doerr, Doerr and Ebel introduced the $(1+(\lambda, \lambda))$ Genetic Algorithm [4]. It employs a special mutation operator and a non-uniform crossover in strong interconnection. As opposed to many other GAs, the crossover is not intended to combine favorable parts of the parent solutions, but to repair malicious mutations. The $(1+(\lambda, \lambda))$ GA can optimize OneMax in linear time, when using an adaptive parameter setting. In every round, λ—the number of mutants—is set to $\lceil \sqrt{n/(n - |x|_1)}\,\rceil$, where x is the currently best solution. This way, an improving move can be found within $\mathrm{O}(1)$ iterations (i.e., $\mathrm{O}(\lambda)$ fitness evaluations), which is significantly faster than the (1+1) EA. For comparison, the single-bit flip operator (random local search) needs time in $\mathrm{O}(\lambda^2)$. Thus this interplay can be seen as an upgraded RLS.

We combine features of the MOEA and the $(1+(\lambda, \lambda))$ GA to the Multi-Objective $(\mu+(\lambda, \lambda))$ Genetic Algorithm (Algorithm 4). The use of variable μ shall indicate that the size of the population maintained by the algorithm may be larger than 1. However, we would like to point out that μ is not a parameter as the size is not fixed to any pre-defined value. The MO $(\mu+(\lambda, \lambda))$ GA implements the same mutation/crossover-scheme as the $(1+(\lambda, \lambda))$ GA in the shape of operators $\mathtt{mutate}_\ell$ and $\mathtt{cross}_c$. Additionally, it is adapted to the multi-objective setting in that the algorithm maintains a population of incomparable solutions. We use the same notion of dominance as for the MOEA. Similar to the approach of the MOEA, we only consider new solutions for selection if they have Hamming weight exactly one larger than that of their parent (exactly one less, for $B > B^*$).

5.1 OneMax with Dynamic Uniform Constraint

We start the analysis of MO $(\mu+(\lambda, \lambda))$ GA on the simplest objective function OneMax. We show that the probability of an iteration of the while-loop to achieve an improvement can be lower bounded by a constant, given that the mutation probability p, crossover probability c, and number of mutants λ are set correctly. The following two lemmata are an adaption of the related results in [4] to the MO (μ+(λ, λ)) GA. The proofs are included for completeness.

Lemma 13 *Starting with a solution x of* OneMax *under dynamic uniform constraint with $|x|_1 = A$ ($B \leq A < B^*$), the probability of an iteration of the while-loop in the MO (μ+(λ, λ)) GA to get a solution y^* with $|y^*|_1 = A + 1$ is greater than C, where $C > 0$ is a constant, if $p = \lambda/n$, $c = 1/\lambda$, and $\lambda = \lceil \sqrt{n/(n - |x|_1)}\, \rceil$.*

Proof We start by analyzing the probability without considering the adaptive setting way of parameters p, c, and λ. In order to get a solution y^*, whose Hamming weight is exactly one larger than that of x, the MOEA has to consider the probability $\Omega\,(|x|_0/n)$ of the standard bit mutation operator that flips exactly one 0-bit in x. However, for the MO (μ+(λ, λ)) GA, the requirement of its mutation phase can be lowered, only demanding that there is a flip from 0 to 1 in the creation of the solution x' (x' is obtained by the mutation phase, to attend the crossover phase). That is because the crossover operator used in the crossover phase is able to find the resulting bit of the flip from 0 to 1, and ignore the resulting bits of all the other flips in x'. Summarizing in a sentence, to get such a solution y^*, it is a necessary condition that there is an index j ($1 \leq j \leq n$) such that $x'_j = 1$ and $x_j = 0$ (since $B \leq B^*$), i.e., the solution x' obtained by the mutation phase is a valid offspring of x.

Observe that the solution x' is not a valid offspring of x if and only if none of the λ mutants that are generated by the operator $\texttt{mutate}_\ell(x)$ on x is a valid offspring of x. An offspring $x^{(i)} = \texttt{mutate}_\ell(x)$ of x is not a valid offspring of x if and only if all the ℓ bits flipping in its creation are 1-bits. Thus, the probability for $\texttt{mutate}_\ell(x)$ to get an invalid offspring of x is

$$\prod_{t=0}^{\ell-1} \frac{|x|_1 - t}{n}.$$

It is an easy conclusion that the probability that none of the λ offspring is a valid offspring of x is

$$\left(\prod_{t=0}^{\ell-1} \frac{|x|_1 - t}{n}\right)^{\lambda} \leq \left(\frac{|x|_1}{n}\right)^{\ell\lambda},$$

and the probability that x' is a valid offspring of x, is at least

$$1 - \left(\frac{|x|_1}{n}\right)^{\ell\lambda}.$$

The following discussion is based on the assumption that the solution x' obtained by the mutation phase is a valid solution of x. Since the mutation operator $\texttt{mutate}_\ell(x)$ flips exactly ℓ bits in x, we only need to consider the ℓ positions where x and x' are different when considering the crossover operator (position j is one of the ℓ positions, where $x'_j = 1$ and $x_j = 0$). For $y^{(i)} = \texttt{cross}_c(x, x')$, the probability that $y^{(i)}$ chooses the x'_j as its j-th bit and the other $\ell-1$ bits in x as its bits in the corresponding positions is $c(1-c)^{\ell-1}$, indicating that $y^{(i)}$ has Hamming weight $|x|_1+1$ with probability not less than $c(1-c)^{\ell-1}$. Observe that the solution y' is not with Hamming weight $|x|_1+1$ if and only if none of the λ offspring that are generated by the operator $\texttt{cross}_c(x, x')$ is with Hamming weight $|x|_1+1$. Thus the probability $|y'|_1 = |x|_1 + 1$ is at least

$$1-\left(1-c\,(1-c)^{\ell-1}\right)^{\lambda}.$$

Combining the probability that the mutation phase gets a valid offspring x' of x, the probability to get a solution y^* with Hamming weight $A+1$ within an iteration of the while-loop is at least

$$\left(1-\left(\frac{|x|_1}{n}\right)^{\ell\lambda}\right)\left(1-\left(1-c(1-c)^{\ell-1}\right)^{\lambda}\right).$$

Now we incorporate the adaptive parameter setting, dependent on the current solution x. We get a lower bound on the probability to reach a solution y^* having Hamming weight $A+1$ within one iteration of the while-loop. First of all, we consider the case $\lambda = \lceil\sqrt{n/(n-|x|_1)}\,\rceil = 1$, i.e., $0 \le |x|_1 < \frac{3n}{4}$. The parameter ℓ is set as 1 by the binomial distribution $\mathrm{Bin}(n, p)$, where $p = \lambda/n = 1/n$, with probability $\Omega(1/e)$. And the probability for the mutation operator $\texttt{mutate}_\ell(x)$ to flip a 0-bit in x is $\Omega(1)$. Using the fact that the parameter c is set to $1/\lambda = 1$ in this case, the claim now follows.

Thus we only consider the case $\lambda \ge 2$ in the following discussion. Denote by L the random variable sampled by the binomial distribution $\mathrm{Bin}(n, p)$, and K the success to get a solution y^* with Hamming weight $A+1$ within an iteration of the while-loop. We have

$$\Pr[K] \ \ge \sum_{\ell=\lceil\lambda/4\rceil}^{7\lambda/4} \Pr[K|L=\ell]\cdot\Pr[L=\ell],$$

where $\Pr[K|L=\ell] \ge (1-(\frac{|x|_1}{n})^{\ell\lambda})(1-(1-c(1-c)^{\ell-1})^{\lambda})$.

Since $c = 1/\lambda$, and that only the values $\ell \in [\lambda/4, 7\lambda/4]$ are considered, we can get

$$\left(1-c\,(1-c)^{\ell-1}\right)^{\lambda} \ \le\ \left(1-\frac{1}{\lambda}\left(1-\frac{1}{\lambda}\right)^{\frac{7\lambda}{4}}\right)^{\lambda} \ \le\ \left(1-\frac{1}{8\sqrt{2}\lambda}\right)^{\lambda} \ \le\ e^{-\frac{1}{8\sqrt{2}}}.$$

The second inequality above always holds because of the fact that $(1-1/a)^a \ge 1/4$ for any $a \ge 2$. For the term $1-(\frac{|x|_1}{n})^{\ell\lambda}$, we have

$$1-\left(\frac{|x|_1}{n}\right)^{\ell\lambda} \geq 1-\left(\frac{|x|_1}{n}\right)^{\frac{\lambda^2}{4}} \geq 1-\left(1-\frac{n-|x|_1}{n}\right)^{\frac{\sqrt{n/(n-|x|_1)}^2}{4}}$$
$$\geq 1-\left(1-\frac{n-|x|_1}{n}\right)^{\frac{n}{n-|x|_1}\cdot\frac{1}{4}} \geq 1-e^{-\frac{1}{4}}.$$

Thus $\Pr[K|L=\ell]$ is greater than a positive constant $\alpha=(1-e^{-\frac{1}{8\sqrt{2}}})(1-e^{-\frac{1}{4}})$, and

$$\Pr[K] \geq \sum_{\ell=\lceil\lambda/4\rceil}^{7\lambda/4} \Pr[K|L=\ell]\cdot\Pr[L=\ell] \geq \alpha\cdot\sum\nolimits_{\ell=\lceil\lambda/4\rceil}^{7\lambda/4}\Pr[L=\ell].$$

When n is sufficiently large, the Chebyshev's inequality [2] can be used to estimate the fraction of values that are more than a certain distance from the mean $np=\lambda$ of the binomial distribution $\text{Bin}(n,p)$. Specifically, no more than $1/k^2$ of the binomial distribution's values can be more than k standard deviations away from the mean (the standard deviation of the binomial distribution $\text{Bin}(n,p)$ is $\sqrt{np(1-p)}$). Consider the case $k=1.05$, thus more than 9% of values drawn from $\text{Bin}(n,p)$ are within 1.05 standard deviations ($1.05\sqrt{np(1-p)}$) from the mean $np=\lambda$, and

$$\sum_{\ell=\lceil\lambda/4\rceil}^{7\lambda/4}\Pr[L=\ell] \geq \sum_{\ell=\lceil\lambda-1.05\sqrt{np(1-p)}\rceil}^{\lambda+1.05\sqrt{np(1-p)}}\Pr[L=\ell]\geq 0.09\ .$$

The first inequality holds because $1.05\sqrt{np(1-p)}=1.05\sqrt{\lambda(1-\frac{\lambda}{n})}\leq 1.05\sqrt{\lambda}\leq\frac{3\lambda}{4}$ for any $\lambda\geq 2$. □

Lemma 14 *Starting with a solution x of* ONEMAX *under dynamic uniform constraint with $|x|_1=A$ ($B\geq A>B^*$), the probability of an iteration of the while-loop in the MO (μ+(λ,λ)) GA to get a solution y^* with $|y^*|_1=A-1$ is greater than C, where $C>0$ is a constant, if $p=\lambda/n$, $c=1/\lambda$, and $\lambda=\lceil\sqrt{n/|x|_1}\,\rceil$.*

Proof The reasoning runs in a similar way to that for Lemma 13. Since the case $B>B^*$ is considered in the lemma, the roles of 1-bits and 0-bits in the necessary condition to get a solution y^* with Hamming weight $A-1$ are reversed. Specifically, to get a solution y^* whose Hamming weight is exactly one less than that of x, it is a necessary condition that there is a position j ($1\leq j\leq n$) in the solution x' obtained by the mutation phase such that $x'_j=0$ and $x_j=1$, i.e., x' is a valid offspring of x.

The probability to get a valid offspring of x at the end of the mutation phase is at least

$$1-\left(\prod_{t=0}^{\ell-1}\frac{n-|x|_1-t}{n}\right)^{\lambda} \geq 1-\left(\frac{n-|x|_1}{n}\right)^{\ell\lambda}.$$

Based on the supposition that the solution x' obtained by the mutation phase is a valid offspring of x, the solution y' obtained by the crossover phase has Hamming weight $A-1$ with probability not less than

$$1-\left(1-c\,(1-c)^{\ell-1}\right)^{\lambda}.$$

As a result, an iteration of the while-loop can get a solution y^* where $|y^*|_1 = A-1$, with probability not less than

$$\left(1-\left(\frac{n-|x|_1}{n}\right)^{\ell\lambda}\right)\left(1-\left(1-c\,(1-c)^{\ell-1}\right)^{\lambda}\right). \tag{3}$$

The remaining analysis of the probability (3) incorporating the adaptive parameter setting is almost the same as that given in Lemma 13. Summarizing the above analysis, the probability to get a solution y^* having Hamming weight $A-1$ within an iteration of the while-loop in the MO $(\mu+(\lambda,\lambda))$ GA, can be lower bounded by a positive constant as well. □

The above two lemmata show that an iteration of the while-loop in the MO $(\mu+(\lambda,\lambda))$ GA can get an improvement—an increment or decrement of the Hamming weight, respectively—of the considered solution x with a probability greater than a positive constant C. This requires to set the parameters of the MO $(\mu+(\lambda,\lambda))$ GA adaptively in every round. In particular, the parameter λ is set depending on the selected solution $x \in S$.

Theorem 15 *The expected reoptimization time of the MO $(\mu+(\lambda,\lambda))$ GA on* OneMax *under dynamic uniform constraint is*

$$E[T] = \begin{cases} \mathrm{O}\left(\min\{\sqrt{nD^3},\ D^2\sqrt{n/(n-B^*)}\}\right), & \text{if } B \le B^*; \\ \mathrm{O}\left(\min\{\sqrt{nD^3},\ D^2\sqrt{n/B^*}\}\right), & \text{if } B > B^*. \end{cases}$$

Hereby, the parameters adapt to the solution x chosen for mutation:

$$\lambda = \begin{cases} \lceil\sqrt{n/(n-|x|_1)}\,\rceil, & \text{if } B \le B^*; \\ \lceil\sqrt{n/|x|_1}\,\rceil, & \text{if } B > B^*, \end{cases}$$

the mutation probability is $p=\lambda/n$ and the crossover probability is $c = 1/\lambda$.

Proof Note that when $B \le B^*$, it is unnecessary to consider the case that the denominator $n-|x|_1$ of the fraction in λ equals 0 since the considered solution x would be an optimal solution, and the algorithm would stop. Similar analysis applies to the case $B > B^*$.

First, assume $B \le B^*$. Let $x^{(A)}$ be the solution in S with the maximum Hamming weight, where $|x^{(A)}|_1 = A = \max_{x\in S}|x|_1 < B^*$, and let λ_A be the corresponding value of λ with respect to $x^{(A)}$. For any solution in S with Hamming weight less than A, its corresponding value of λ is not greater than λ_A according to the setting way of λ. Thus if an iteration of the while-loop chooses a solution in S with Hamming weight less than A, then the number of the fitness evaluations wasted in the iteration (specifically, in the crossover phase of the iteration) is not greater than λ_A. An iteration of the while-loop chooses the solution $x^{(A)}$ with probability $\Omega(1/(D+1))$ since the size of S can be bounded by $D+1$. By Lemma 13, the expected number of fitness evaluations that the MO (μ+(λ, λ)) GA takes to get a solution with Hamming weight $A+1$, can be bounded by $\mathrm{O}(D\lambda_A) = \mathrm{O}(D\sqrt{n/(n-A)})$, which obviously can be accepted by the algorithm.

To get the expected runtime that the MO (μ+(λ, λ)) GA takes to reach a population in which one of the solutions has Hamming weight B^*, starting with the initial population $\{x_{\text{orig}}\}$, it is necessary to consider all possible values of A ($B \le A < B^*$) and sum over all the waiting times from A to $A+1$. Therefore, the expected reoptimization time of the MO (μ+(λ, λ)) GA can be bounded by $\mathrm{O}(D\cdot\sum_{A=B}^{B^*-1}\sqrt{n/(n-A)}) = \mathrm{O}(\sqrt{nD^3})$, because

$$\sum_{i=B}^{B^*-1}\sqrt{\frac{1}{n-i}} \le \int_B^{B^*}\sqrt{\frac{1}{n-i}}\,\mathrm{d}i = 2\sqrt{n-B} - 2\sqrt{n-B^*} \le 2\sqrt{D}.$$

Observe that the upper bound for the size of the population S in above analysis is not tight, which actually can be bounded by $A-B+1 \le D+1$. Thus another analysis about the expected reoptimization time for $B \le B^*$ is given as follows. The above analysis gives that the MO (μ+(λ, λ)) GA takes expected runtime $\mathrm{O}(|S|\lambda_A) = \mathrm{O}((A-B+1)\sqrt{n/(n-A)})$ to sample a solution with Hamming weight $A+1$, and expected runtime

$$\mathrm{O}\left(\sum_{A=B}^{B^*-1}\left((A-B+1)\sqrt{n/(n-A)}\right)\right)$$

to find a solution with Hamming weight B^* starting with the initial population $\{x_{\text{orig}}\}$. Because the values of function $f(x)$ and its second derivative $f''(x)$,

$$f(x) = (x-B+1)\sqrt{\frac{n}{n-x}}, \quad f''(x) = \sqrt{\frac{n}{(n-x)^3}} + \frac{3}{4}(x-B+1)\sqrt{\frac{n}{(n-x)^5}},$$

are always positive when $B \le x \le B^*$, so

$$\begin{aligned}\sum_{A=B}^{B^*-1}\left((A-B+1)\sqrt{\frac{n}{n-A}}\right) &\le \frac{f(B)+f(B^*)}{2}\cdot(B^*-B)\\ &= \frac{D}{2}\cdot\left(\sqrt{\frac{n}{n-B}} + (D+1)\cdot\sqrt{\frac{n}{n-B^*}}\right)\\ &\le \frac{D(D+2)}{2}\cdot\sqrt{\frac{n}{n-B^*}}.\end{aligned}$$

Therefore, the claimed bound of $\mathrm{O}(\min\{\sqrt{n D^3}, D^2\sqrt{n/(n-B^*)}\})$ for $B \le B^*$ is proved, if the parameters of the MO $(\mu+(\lambda,\lambda))$ GA are set dependently with the chosen solution x in each iteration of the while-loop, as $p=\lambda/n, c=1/\lambda$, and $\lambda = \lceil \sqrt{n/(n-|x|_1)}\, \rceil$ (runtime $\mathrm{O}(D^2\sqrt{n/(n-B^*)})$ is to be read as $\mathrm{O}(\sqrt{n}D^2)$ if $n=B^*$).

For $B > B^*$, let $x^{(A)}$ be the solution in S with the minimum Hamming weight, where $|x^{(A)}|_1 = A = \min_{x\in S} |x|_1 > B^*$, and let λ_A be the corresponding value of λ with respect to $x^{(A)}$. Similar to the analysis for $B \le B^*$, if an iteration of the while-loop chooses a solution in S with Hamming weight greater than A, then the number of the fitness evaluations wasted in the iteration is not greater than λ_A. Combining Lemma 14 and the upper bound $D+1$ for the population size, the MO $(\mu+(\lambda,\lambda))$ GA takes expected runtime $\mathrm{O}(D\sqrt{n/A})$ to sample a solution with Hamming weight $A-1$, and expected runtime $\mathrm{O}(D\sum_{A=B^*+1}^{B}\sqrt{n/A}) = \mathrm{O}(\sqrt{nD^3})$ to find a solution with Hamming weight B^* starting with the initial population $\{x_{\mathrm{orig}}\}$, because

$$\sum_{i=B^*+1}^{B}\sqrt{\frac{1}{i}} \le \int_{B^*+1}^{B+1}\sqrt{\frac{1}{i}}\,\mathrm{d}i = 2\sqrt{B+1}-2\sqrt{B^*+1} \le 2\sqrt{D}.$$

For $B > B^*$, similar to the analysis for $B \le B^*$, the upper bound $D+1$ for the population size $|S|$ used in above analysis can be replaced by $B-A+1 \le D+1$. Thus the MO $(\mu+(\lambda,\lambda))$ GA takes expected runtime $\mathrm{O}(|S|\lambda_A) = \mathrm{O}((B-A+1)\sqrt{n/A})$ to sample a solution with Hamming weight $A-1$, and takes expected runtime $\mathrm{O}(\sum_{A=B^*+1}^{B}((B-A+1)\sqrt{n/A}))$ to find a solution with Hamming weight B^*. For the upper bound of $\sum_{A=B^*+1}^{B}((B-A+1)\sqrt{n/A})$, because the values of function $f(x)$ and its second derivative $f''(x)$,

$$f(x) = (B-x+1)\sqrt{\frac{n}{x}}, \quad f''(x) = \sqrt{\frac{n}{x^3}} + \frac{3(B-x-1)}{4}\sqrt{\frac{n}{x^5}},$$

are always positive when $B^* \le x \le B$, so

$$\begin{aligned}\sum_{A=B^*+1}^{B}\left((B-A+1)\sqrt{\frac{n}{A}}\right) &\le \frac{f(B)+f(B^*)}{2}\cdot(B-B^*)\\ &= \frac{D}{2}\cdot\left(\sqrt{\frac{n}{B}}+(D+1)\sqrt{\frac{n}{B^*}}\right) \le \frac{D(D+2)}{2}\cdot\sqrt{\frac{n}{B^*}}.\end{aligned}$$

Therefore, the claimed bound of $\mathrm{O}(\min\{\sqrt{nD^3}, D^2\sqrt{n/B^*}\})$ for $B > B^*$ is proved, if the parameters of the MO $(\mu+(\lambda,\lambda))$ GA are set dependently with the chosen solution x in each iteration of the while-loop, as $p=\lambda/n, c=1/\lambda$, and $\lambda = \lceil\sqrt{n/|x|_1}\,\rceil$. $\square$

5.2 Linear Function with Dynamic Uniform Constraint

We now turn to linear functions under dynamic uniform constraints and start by lower bounding the probability of an improvement if the parameters are set in the right way.

As for OneMax, every bit in the considered bit string has the same weight (i.e., every bit string with Hamming weight $A+1$ is an optimal solution with Hamming weight $A+1$), Lemmata 13 and 14 thus study the probability of an iteration of the while-loop in the MO $(\mu+(\lambda,\lambda))$ GA to find an arbitrary solution with Hamming weight $A+1$ if $B \le B^*$ ($A-1$ if $B > B^*$), starting with a solution with Hamming weight A. Similarly, for the linear profit function, we can study the probability of an iteration of the while-loop to find an optimal solution with Hamming weight $A+1$ if $B \le B^*$ ($A-1$ if $B > B^*$), starting with an optimal solution with Hamming weight A. However, as mentioned in the proof of Theorem 9, it is unnecessary to find the optimal solution for each Hamming weight between B and B^*. Thus, using the notion "candidate" given in the proof of Theorem 9, we analyze the probability of an iteration of the while-loop to find a candidate with Hamming weight $A+1$ if $B \le B^*$ ($A-1$ if $B > B^*$), starting with a candidate with Hamming weight A. Recall the definition of "candidate". For the case $B \le B^*$, a feasible solution x' is a *candidate* if there exists an optimal solution x^* such that $x^*_i = 1$ whenever $x_i = 1$.

Lemma 16 *Starting with a candidate x of a linear profit function under dynamic uniform constraint with $|x|_1 = A$ ($B \le A < B^*$), the probability of an iteration of the while-loop in the MO $(\mu+(\lambda,\lambda))$ GA to get a candidate y^* with $|y^*|_1 = A+1$ is greater than C/λ, where $C > 0$ is a constant, if $p = \lambda/n, c = 1/\lambda$, and $\lambda = \lceil \sqrt{n/(B^* - |x|_1)}\, \rceil$.*

Proof The reasoning runs in a similar way to that of Lemma 13. Let I be the set containing the positions of all 0-bits in x such that a candidate with Hamming weight $|x|_1 + 1$ can be found by flipping any one of these 0-bits. Note that I has at least $B^* - |x|_1$ elements. As the analysis given in Lemma 13, to get a candidate y^* with Hamming weight $|x|_1 + 1$, it is a necessary condition that the solution x' obtained by the mutation phase has a 1-bit x'_j with $j \in I$.

For any mutant $x^{(i)} = \mathtt{mutate}_\ell(x)$ that is obtained by the operator $\mathtt{mutate}_\ell(x)$ on x, the probability that $x^{(i)}_j = 0$ for all $j \in I$ is not greater than

$$\prod_{t=0}^{\ell-1} \frac{n - (B^* - |x|_1) - t}{n}.$$

Thus the probability that $x^{(i)}_j = 0$ for all $1 \le i \le \lambda$ and all $j \in I$ is

$$\left(\prod_{t=0}^{\ell-1} \frac{n - (B^* - |x|_1) - t}{n}\right)^{\lambda} \le \left(\frac{n - (B^* - |x|_1)}{n}\right)^{\ell\lambda},$$

and the probability that there exists a mutant among $\{x^{(1)}, \dots, x^{(\lambda)}\}$ such that it has a 1-bit with position in I is not less than

$$1 - \left(\frac{n - (B^* - |x|_1)}{n}\right)^{\ell\lambda}.$$

Assume that an offspring whose j-th bit is 1 has been obtained by the operator $\texttt{mutate}_\ell(x)$ during the mutation phase, where $j \in I$. Note that the solution is a valid offspring of x, but it may not be the unique valid offspring of x in $\{x^{(1)}, \dots, x^{(\lambda)}\}$. Consequently, the solution is chosen as x' with probability $\Omega(1/\lambda)$. And at the end of the mutation phase, the event $x'_j = 1$ happens with probability

$$\frac{1}{\lambda}\left(1 - \left(\frac{n - (B^* - |x|_1)}{n}\right)^{\ell\lambda}\right).$$

Now we consider the event that the solution y' obtained by the crossover phase is a candidate with Hamming weight $|x|_1 + 1$, based on x and x', where bit x'_j is assumed to be 1 ($j \in I$). Note that the mutation operator $\texttt{mutate}_\ell(x)$ flips exactly ℓ bits in x, thus only the ℓ positions where x and x' are different need to be considered. For $y^{(i)} = \texttt{cross}_c(x, x')$, the probability that $y^{(i)}$ chooses x'_j as its j-th bit and the other $\ell - 1$ bits in x as its bits in the corresponding positions is at least $c(1-c)^{\ell-1}$. Therefore, the crossover phase gets a candidate y^* having Hamming weight $|x|_1 + 1$ with probability not less than $1 - (1 - c(1-c)^{\ell-1})^\lambda$.

The probability to get a candidate y^* having Hamming weight $|x|_1 + 1$ within an iteration of the while-loop, is at least

$$\frac{1}{\lambda}\left(1 - \left(\frac{n - (B^* - |x|_1)}{n}\right)^{\ell\lambda}\right)\left(1 - \left(1 - c\,(1-c)^{\ell-1}\right)^{\lambda}\right).$$

We combine the above conclusion with Lemma 13 to get a bound of C/λ, C a constant, on the probability to sample a candidate y^* with $|y^*| = |x|_1 + 1$. □

Now we consider the other case $B > B^*$. Recall that an unfeasible solution x' of a linear profit function with Hamming weight between B and B^*, is a *candidate* if there is an optimum x^* such that $x_i = 0$ implies $x^*_i = 0$. Using similar ideas as in the previous proof, we have the lemma below.

Lemma 17 *Starting with a candidate x of a linear profit function under dynamic uniform constraint with $|x|_1 = A$ ($B \ge A > B^*$), the probability of an iteration of the while-loop in the MO $(\mu+(\lambda,\lambda))$ GA to get a candidate y^* with $|y^*|_1 = A - 1$ is greater than C/λ, where $C > 0$ is a constant, if $p = \lambda/n$, $c = 1/\lambda$, and $\lambda = \lceil \sqrt{n/(|x|_1 - B^*)}\, \rceil$.*

Finally, we show the upper bound of the expected reoptimization time for the MO $(\mu+(\lambda,\lambda))$ GA on linear functions with dynamic uniform constraints.

Theorem 18 *The expected reoptimization time of the MO* (μ+(λ, λ)) *GA on a linear profit function under dynamic uniform constraint is of order* $\mathrm{O}(nD)$*. Hereby, the parameters adapt to the solution* x *chosen for mutation:*

$$\lambda = \begin{cases} \lceil \sqrt{n/\max\{B^* - |x|_1, 1\}}\,\rceil, & \textit{if } B \leq B^*; \\ \lceil \sqrt{n/\max\{|x|_1 - B^*, 1\}}\,\rceil, & \textit{if } B > B^*, \end{cases}$$

the mutation probability is $p = \lambda/n$ *and the crossover probability is* $c = 1/\lambda$.

Proof The setting of parameter λ given above slightly differs from that given in Lemmata 16 and 17, because the two lemmata do not consider the case that an iteration of the while-loop chooses a solution with Hamming weight B^*, for which the denominator $B^* - |x|_1$ of the fraction in λ equals 0 (the solution may not be an optimal solution with Hamming weight B^* for the linear function, and the algorithm cannot stop). However, the claims of Lemmata 16 and 17 also hold under the parameter setting given above. Observe that if an iteration of the while-loop chooses a solution with Hamming weight B^*, then the offspring obtained by the crossover phase has Hamming weight $B^* + 1$ if $B \leq B^*$ (or $B^* - 1$ if $B > B^*$), which would be rejected by the algorithm.

Denote by λ_j be the corresponding value of λ with respect to a solution with Hamming weight j. The following proof for the expected reoptimization time runs in a similar way to that of Theorem 15. For $B \leq B^*$, let $x^{(A)}$ be the candidate in S with the maximum Hamming weight, where $A = |x^{(A)}|_1 < B^*$. Note that since $x^{(A)}$ may not be the solution in S with the maximum Hamming weight, the size of the population S cannot be bounded by $A - B + 1$, as that given in Theorem 15. By the same argument, the number N_{fe} of fitness evaluations taken in an iteration of the while-loop cannot be upper bounded by λ_A as Theorem 15. Fortunately, we have the observation that the maximum Hamming weight that the solutions in S have is K ($B \leq K \leq B^*$) if and only if there is a solution in S for each value between B and K (since the crossover phase only considers the solutions with Hamming weight exactly one larger than that of the solution chosen by the iteration of the while-loop), which implies that $|S| = K - B + 1$. Moreover, the term $\lceil \sqrt{n/\max\{B^* - |x|_1, 1\}}\,\rceil$ is monotonically non-decreasing in $|x|_1$, $B \leq |x|_1 \leq B^*$. Using these properties, the expected value $E(N_{fe})$ of N_{fe} can be upper bounded by

$$\begin{aligned} E(N_{fe}) &= \frac{1}{|S|} \cdot \sum_{j=B}^{K} \lambda_j = \frac{1}{K - B + 1} \cdot \sum_{j=B}^{K} \lambda_j \leq \frac{1}{B^* - B + 1} \cdot \sum_{j=B}^{B^*} \lambda_j \\ &\leq \frac{1}{D+1} \cdot \left(\sqrt{n} + \sum_{|x|_1 = B}^{B^*-1} \sqrt{\frac{n}{B^* - |x|_1}} \right) \\ &\leq \frac{\sqrt{n}}{D+1} \cdot \left(1 + \sum_{i=1}^{D} \sqrt{\frac{1}{i}} \right) \\ &\leq \frac{\sqrt{n}}{D+1} \cdot \left(1 + \int_0^D i^{-\frac{1}{2}}\, \mathrm{d}i \right) = \frac{\sqrt{n}(2\sqrt{D} + 1)}{D+1}. \end{aligned}$$

By Lemma 16, the MO $(\mu+(\lambda, \lambda))$ GA takes an expected number of $\mathrm{O}(D\lambda_A)$ iterations of the while-loop to find a candidate with Hamming weight $A+1$. Combining the expected number $E(N_{fe})$ of fitness evaluations taken in an iteration of the while-loop, the MO $(\mu+(\lambda, \lambda))$ GA takes expected runtime $\mathrm{O}(D\lambda_A \cdot E(N_{fe})) = \mathrm{O}(\sqrt{nD}\lambda_A)$ to sample a candidate with Hamming weight $A + 1$, which can be accepted by the algorithm.

By considering all possible values of A ($B \leq A \leq B^* - 1$) and summing over the waiting times from A to $A + 1$, the MO $(\mu+(\lambda, \lambda))$ GA takes expected runtime

$$\begin{aligned}\mathrm{O}\left(\sqrt{nD} \cdot \sum_{A=B}^{B^*-1} \lambda_A\right) &= \mathrm{O}\left(\sqrt{nD} \cdot \sum_{A=B}^{B^*-1} \sqrt{\frac{n}{B^* - A}}\right) \\ &= \mathrm{O}\left(n\sqrt{D} \cdot \sum_{i=1}^{D} \sqrt{\frac{1}{i}}\right) \\ &= \mathrm{O}\left(n\sqrt{D} \cdot \left(\int_0^D i^{-\frac{1}{2}} \,\mathrm{d}i\right)\right) = \mathrm{O}\left(nD\right)\end{aligned}$$

to find a candidate with Hamming weight B^* starting with the initial population $\{x_{\text{orig}}\}$, which is also an optimal solution with Hamming weight B^*.

For the case $B > B^*$, using the similar reasoning to that given above and Lemma 17 yields the same expected reoptimization runtime $\mathrm{O}(nD)$. □

6 Conclusion

In this article, we presented a dynamic model for optimizing linear functions under a uniform constraint. Our results show that different types of evolutionary algorithms are capable of efficiently recomputing an optimum after a constraint change, starting from an optimal solution to the old setting. Evolutionary computing techniques seem to be particularly well-equipped for this task. Our bounds on the reoptimization time improve significantly upon the known optimization times from scratch in many cases.

However, the constrained nature of the problem gives rise to several obstacles. Most notably, the usual selection mechanisms bar the algorithms from taking possible short-cuts via infeasible solutions. This, in turn, can lead to artificially high waiting times for further improvements once the optimization is close to the boundary of the feasible region. We showed how a multi-objective approach as well as the use of a population of individuals can abate this negative influence. Hereby, one must carefully balance the speedup of multiple individuals with the slowdown of large populations. The additional objective stemming from the constraint handling naturally lends itself as a criteria when to include new solutions. Finally, our multi-objective adaption of the $(1+(\lambda, \lambda))$ GA, achieves an even better performance employing an adaptive parameter scheme.

This work is meant as a first step towards the theoretical understanding of nature-inspired optimization under dynamic constraints. We examined several promising constraint handling techniques on the most basic objective functions. We will leave it for future research to extend these insights to more classes of algorithms and possibly some real-world optimization problems.

Acknowledgements The work leading up to this article has received funding from the German Research Foundation under Grant agreement FR2988 (TOSU) and the Australian Research Council under Grant agreements DP140103400 and DP160102401. We would like to thank the anonymous referees, whose comments and suggestions have greatly improved this paper.

References

1. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics: Foundations and Recent Developments, vol. 1. World Scientific, Singapore (2011)
2. Bienaymé, I.J.: Considérations à l'appui de la découverte de Laplace sur la loi de probabilité dans la méthode des moindres carrés. Imprimerie de Mallet-Bachelier (1853)
3. Doerr, B., Doerr, C.: Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings. In: Proceedings of the 2015 Genetic and Evolutionary Computation Conference (GECCO), pp. 1335–1342 (2015)
4. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theor. Comput. Sci. **567**, 87–104 (2015)
5. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)
6. Droste, S., Jansen, T., Wegener, I.: On the analysis of the $(1+1)$ evolutionary algorithm. Theor. Comput. Sci. **276**, 51–81 (2002)
7. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. Evol. Comput. **18**, 617–633 (2010)
8. Friedrich, T., Kötzing, T., Lagodzinski, J.A.G., Neumann, F., Schirneck, M.: Analysis of the $(1+1)$ EA on subclasses of linear functions under uniform and linear constraints. In: Proceedings of the 14th Workshop on the Foundations of Genetic Algorithms (FOGA), pp. 45–54 (2017)
9. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Nat. Comput. **3**, 21–35 (2004)
10. Jansen, T.: Analyzing Evolutionary Algorithms. The Computer Science Perspective. Natural Computing Series. Springer, Berlin (2013)
11. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)
12. Kötzing, T., Lissovoi, A., Witt, C.: $(1 + 1)$ EA on generalized dynamic OneMax. In: Proceedings of the 13th Workshop on the Foundations of Genetic Algorithms (FOGA), pp. 40–51 (2015)
13. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica **65**, 754–771 (2013)
14. Mezura-Montes, E., Coello, C.A.C.: Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm Evol. Comput. **1**, 173–194 (2011)
15. Mühlenbein, H.: How genetic algorithms really work: Mutation and hillclimbing. In: Proceedings of the 2nd Conference on Parallel Problem Solving from Nature (PPSN), vol. 92, pp. 15–25 (1992)
16. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Nat. Comput. **5**, 305–319 (2006)
17. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity. Springer, Berlin (2010)
18. Shi, F., Schirneck, M., Friedrich, T., Kötzing, T., Neumann, F.: Reoptimization times of evolutionary algorithms on linear functions under dynamic uniform constraints. In: Proceedings of the 2017 Genetic and Evolutionary Computation Conference (GECCO), pp. 1407–1414 (2017)
19. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**, 294–318 (2013)
20. Zhou, Y., He, J.: A runtime analysis of evolutionary algorithms for constrained optimization problems. IEEE Trans. Evol. Comput. **11**, 608–619 (2007)