Reoptimization Times of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints

Feng Shi
Central South University
School of Inf. Sc. and Engineering
Changsha, China

Martin Schirneck
Hasso Plattner Institute
Potsdam, Germany

Tobias Friedrich
Hasso Plattner Institute
Potsdam, Germany

Timo Kötzing
Hasso Plattner Institute
Potsdam, Germany

Frank Neumann
The University of Adelaide
School of Computer Science
Adelaide, Australia

ABSTRACT

The investigations of linear pseudo-Boolean functions play a central role in the area of runtime analysis of evolutionary computing techniques. Having an additional linear constraint on a linear function is equivalent to the NP-hard knapsack problem and special problem classes thereof have been investigated in recent works. In this paper, we extend these studies to problems with dynamic constraints and investigate the runtime of different evolutionary algorithms to recompute an optimal solution when the constraint bound changes by a certain amount. We study the classical (1+1) EA and population-based algorithms and show that they recompute an optimal solution very efficiently. Furthermore, we show that a variant of the $(1+(\lambda, \lambda))$ GA can recompute the optimal solution more efficiently in some cases.

CCS CONCEPTS

•Mathematics of computing → Evolutionary algorithms; **•Theory of computation →** *Random search heuristics;* **•General and reference →** General conference proceedings;

KEYWORDS

runtime analysis, reoptimization time, dynamic constraint, uniform constraint, evolutionary algorithm

1 INTRODUCTION

Rigorous runtime analysis has contributed significantly to the theoretical understanding of evolutionary computing techniques over the last 20 years [? ? ?]. In this area of research the class of linear pseudo-Boolean functions plays a crucial role. The simplest linear function OneMax has been the subject of the first runtime analysis of the classical (1+1) EA [?]. Since then various proof techniques have been developed for the analysis of the (1+1) EA on the class of all linear functions. Besides inferring the asymptotic runtime we also gained an understanding of the internal mechanisms leading to the famous $\Theta(n \log n)$ bound [?]. More detailed studies pushed the insights into the optimization process even further and revealed the leading constants [? ? ?].

We investigate the optimization of linear functions for which the collection of feasible solutions is subject to additional constraints. The most general setting of linear functions under linear constraints is known to be equivalent to the classical NP-hard knapsack problem [?]. The (1+1) EA takes exponential time to solve the knapsack problem, even on instances that are optimizable using simple greedy heuristics [?]. Therefore, we consider a subclass of uniform constraints restricting the Hamming weight of a feasible solution. Recent investigations have considered uniform constraints in a static setting, where a fixed bound B on the number of 1-bits is given [?]. We extend these studies to a dynamic change of constraints from B to some new value B^*. Our goal is to analyze the number of generations needed by an evolutionary algorithm to reoptimize an optimal solution of weight B to an optimal solution observing the new bound B^*. We give bounds on the reoptimization times depending on the problem size n, the order of the constraints B and B^* as well as the extent of the change $D = |B^* - B|$.

In this work we compare the runtime of several nature-inspired algorithms on both OneMax and general linear profit functions. Our findings are summarized in Table ??. We start with the classical (1+1) EA. It can reoptimize OneMax quickly. However, the elitist selection and single-objective fitness makes it impossible to find a direct improvement on a general linear objective once the cardinality bound B^* has been reached. This requires the algorithm to "swap" certain bits and causes a super-quadratic runtime.

Various constraint handling methods for evolutionary computing techniques have been discussed in the literature (see the survey [?]). The runtime analysis of evolutionary algorithms for problems with constraints such as minimum spanning trees or minimum vertex covers shows that significant better upper bounds can be achieved by considering the constraints as an additional objective [? ? ?]. We investigate population-based EAs which store individuals with respect to different values/violations of the constraint function. Our multi-objective evolutionary algorithm (MOEA) maintains a population of candidate solutions, one for each Hamming weight between B and B^*. The MOEA avoids the

GECCO '17, Berlin, Germany
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. 978-1-4503-4920-8/17/07...$15.00
DOI: http://dx.doi.org/10.1145/3071178.3071270

necessity of a swap but, in turn, the size of the population slows down the optimization process. An improved variant of the MOEA, the Multi-Objective Evolutionary Algorithm with single bit flip (MOEA-S), tackles the problem of a large population size. It is able to emulate the swapping move with only two individuals.

Finally, we examine the performance of an adaption of the $(1+(\lambda,\lambda))$ GA [?]. As opposed to many other GAs, it does not use crossover to recombine good parts of candidate solutions but instead to repair malicious mutations. It has been shown, using an adaptive parameter setting, that the $(1+(\lambda,\lambda))$ GA can optimize OneMax in linear time [?]. We use a variant of this algorithm, called the Multi-Objective Genetic Algorithm (MOGA), to prove that for OneMax, a reoptimization time with sub-linear dependence on n is possible when the change $D = |B^* - B|$ is a constant.

The rest of the paper is structured as follows. Section 2 introduces related definitions and the four algorithms. A detailed analysis of the $(1+1)$ EA, MOEA (and its variant MOEA-S), and MOGA are given in Sections 3 through 5, respectively. We use Section 6 to conclude this work.

2 PRELIMINARIES

2.1 General Setting

We analyze the behavior of several evolutionary computation techniques on the search space $\{0,1\}^n$ of all bit strings with fixed length n. The objective is given as a linear function. That is, for a sequence $(w_i)_{1\leq i\leq n}$ of positive real weights, let the *profit* of a search point $x = x_1x_2\ldots x_n$ be defined as

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

W.l.o.g. the weights are not smaller than 1. We use $w_{\max} = \max_i w_i$ to denote the maximum weight. The simplest profit function is $\text{OneMax}(x) = \sum_{i=1}^{n} x_i$. We would like to point out that we distinguish between the profit and the fitness of an individual. The latter is an implementation detail of the algorithm in question while the general aim is to maximize the former (cf. Section ??).

The constrained aspect of the optimization is modeled by declaring a subset of the search space as the *feasible region* and disregard infeasible search points as solutions to the optimization problem, even if their profit would be higher than that of any feasible point. An *optimal solution* is then a feasible bit string of maximum profit. We content ourselves with restricting the number of 1-bits in a feasible solution. It has been shown that the more general class of "linear constraints" leads to exponential runtimes for many EAs even on simple problems [? ?]. Let $|x|_1$ denote the *Hamming weight* of x, that is, the number of 1-bits in x, and $|x|_0 = n - |x|_1$ the number of 0-bits in x. Further, let the *cardinality bound* $0 \leq B \leq n$ be a non-negative integer. In summary, we consider the following general optimization problem,

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

We are not primarily interested in the time a given EA needs to solve this problem. Instead, we fix a second integer $0 \leq B^* \leq n$ and investigate the number of generations the algorithm needs to sample an optimal solution to the problem with the new cardinality bound B^* for the first time, starting from an optimal solution x_{org} to the original problem. We refer to this setting as profit function P being under *dynamic uniform constraint*. The number of generations needed is called the *reoptimization time* and symbol T is used to denote this random variable. Its expectation $E[T]$ (over the random decisions of the algorithm) is the *expected reoptimization time*.

In our analysis we distinguish four cases depending on the type of profit P and the relation of the two bounds B and B^*. The profit function can either be equal to OneMax or a general linear function. Independently, B^* can either be larger or smaller than B; the case $B = B^*$ is trivial. Not surprisingly, the reoptimization time depends on the absolute distance of B and B^*. To ease notation we let $D = |B^* - B|$ denote this distance. Observe that D is then a positive integer not larger than n.

2.2 Algorithms

We consider four nature-inspired algorithms, namely, the $(1+1)$ Evolutionary Algorithm, the Multi-Objective Evolutionary Algorithm, the MOEA with single bit flip and the Multi-Objective Genetic Algorithm. All these methods employ their own *fitness function* for the optimization, reflecting different ways to handle the constraint.

Algorithm 1: (1+1) EA

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

For the $(1+1)$ EA (Algorithm ??) we use the notion of fitness that has been suggested by Friedrich et al. in [?]. The single-objective fitness function $f_{(1+1)}$, on bit strings $x \in \{0,1\}^n$, 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. The large penalty term scales with the extent of the constraint violation and thus guides the search towards the feasible region (given by B^*). Also, once the algorithm samples the first feasible solution, its elitist selection bars it from adopting an infeasible search point ever again.

The two variants of the Multi-Objective Evolutionary Algorithm use a vector-valued fitness function by combining the Hamming weight and the profit of a solution, $f_{\text{MOEA}}(x) = (|x|_1, P(x))$. We say that a solution y *dominates* another solution z w.r.t. f_{MOEA}, written $y \succcurlyeq_{\text{MOEA}} z$, if $|y|_1 = |z|_1$ while $P(y) \geq P(z)$. This declares a partial ordering on $\{0,1\}^n$ in which two strings are comparable if and only if they have the same number of 1-bits. We let $y \succ_{\text{MOEA}} z$ stand for $y \succcurlyeq_{\text{MOEA}} z \wedge P(y) \neq P(z)$.

The idea of the MOEA (Algorithm ??) is to maintain a set S of incomparable individuals, one for each Hamming weight between B and B^*. The population is initialized with an optimal solution

Profit Function	(1+1) EA	MOEA	MOEA-S	MOGA	
OneMax	$O\left(n \log\left(\frac{n-B}{n-B^*}\right)\right)$	$O\left(nD \log\left(\frac{n-B}{n-B^*}\right)\right)$	$O\left(n \log\left(\frac{n-B}{n-B^*}\right)\right)$	$O\left(\sqrt{n} D^{\frac{3}{2}}\right)$	if $B < B^*$
	$O\left(n \log\left(\frac{B}{B^*}\right)\right)$	$O\left(nD \log\left(\frac{B}{B^*}\right)\right)$	$O\left(n \log\left(\frac{B}{B^*}\right)\right)$	$O\left(\sqrt{n} D^{\frac{3}{2}}\right)$	if $B > B^*$
linear function	$O(n^2 \log(B^* w_{\max}))$	$O(nD^2)$	$O(n \log D)$	$O(nD^2)$	

Table 1: Overview of Results. 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 Genetic Algorithm (MOGA) 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 $O(n \log(B/B^*))$ are to be read as $O(n \log B)$, 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 [?].

Algorithm 2: MOEA; Assuming $B \leq B^*$.

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

x_{org} having exactly B 1-bits. If standard bit mutation applied to a random member of S results in an admissible offspring y, the algorithm checks whether it is already dominated by another individual in S. If not, y is included and all solutions that are dominated by the new string (excluding y itself, of course) are discarded. Note that set S can grow up to size $|B^* - B| + 1 = D + 1$.

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

```
1 $S \leftarrow \{x_{\text{org}}\}$;
2 while stopping criterion not met do
3     Choose $x \in S$ uniformly at random;
4     $y \leftarrow$ flip bit $x_i$ with $i \in \{0, \ldots, n\}$ chosen u.a.r.;
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 MOEA and its MOEA-S variant (Algorithm ??) have two major differences. The latter uses the single-bit flip operator usually employed in Random Local Search (RLS). The other distinction is the notion of dominance, written $\succcurlyeq_{\text{MOEA-S}}$, although both variants use the same fitness function. If $B \leq B^*$, bit strings $y, z \in \{0,1\}^n$, for which *at most one* value $|y|_1, |z|_1$ equals B^* or $B^* - 1$, are ordered lexicographically,

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

If both $|y|_1, |z|_1 \in \{B^*, B^* - 1\}$, we put

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

As a result, two strings are incomparable, written $y \parallel_{\text{MOEA-S}} z$, iff $|y|_1 = B^*$ and $|z|_1 = B^* - 1$ or vice versa. If $B > B^*$, we only switch the dependency on the number of 1-bits in (??) to $|y|_1 \leq |z|_1$. Again, the population S of the MOEA-S collects incomparable solutions during the optimization, but now can have at most 2 elements.

The MOGA (Algorithm ??) is a multi-objective adaption of the $(1+(\lambda,\lambda))$ GA [?]. Every iteration of the MOGA has three phases. During the mutation phase the algorithm first draws a search point $x \in S$ u.a.r. and a number ℓ according to the binomial distribution $\text{Bin}(n,p)$ with parameters n and the *mutation probability* p. Then, λ offsprings are generated by the mutation operator mutate_ℓ which flips exactly ℓ bits chosen u.a.r. This means, the offspring of x are λ ℓ-Hamming neighbors chosen uniformly. We call an offspring of x *valid* if $|x|_1 \leq B^*$ and at least one 0-bit was flipped in the creation, or $|x|_1 \geq B^*$ and at least one 1-bit flipped. At the end of the mutation phase the algorithm chooses a valid offspring x' for further processing, if there is any; otherwise, $x' = x$. The crossover phase recombines parent string x with its offspring x'. For some fixed *crossover probability* c, the $\text{cross}_c(x, x')$ operator creates a bit string by choosing, in every position $1 \leq i \leq n$, x'_i with probability c and x_i otherwise. This crossover is tried λ times. To rank the generated bit strings, we use the same notion of dominance as the MOEA, i.e., $y \succcurlyeq_{\text{MOEA}} z \Leftrightarrow |y|_1 = |z|_1 \wedge P(y) \geq P(z)$. (Note that this is different from the one used in the MOEA-S.) Hence, there is at most one $\succcurlyeq_{\text{MOEA}}$-maximal recombined individual that has Hamming weight exactly one larger than x. The result of the crossover phase is this string, denoted y', if it exists; otherwise, $y' = x$. In the selection phase the algorithm checks whether y' meets the cardinality constraint and is not dominated by a solution previously in S. If this test is successful, the population is updated in the usual way.

Algorithm 4: MOGA; Assuming $B \leq B^*$. Concept from [?].

```
1  $S \leftarrow \{x_{\text{org}}\}$;
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          $x^{(i)} \leftarrow \text{mutate}_\ell(x)$;
7      $V = \{x^{(i)} \mid x^{(i)}$ 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         $y^{(i)} \leftarrow \text{cross}_c(x, x')$;
13     $M = \{y^{(i)} \mid y^{(i)}$ is $\succcurlyeq_{\text{MOEA}}$-maximal $\wedge |y^{(i)}|_1 = |x|_1 + 1\}$;
14     if $M = \{y\}$ then
15         $y' \leftarrow y$;
16     else $y' \leftarrow x$;
       /* Selection phase. */
17     if $(B^* \geq |y'|_1 \geq B) \wedge (\nexists w \in S\colon w \succcurlyeq_{\text{MOEA}} y')$ then
18         $S \leftarrow (S \cup \{y'\}) \setminus \{z \in S \mid y' \succ_{\text{MOEA}} z\}$;
```

3 ANALYSIS OF THE (1+1) EA

Theorem 3.1. *The expected reoptimization time of the* (1+1) EA *on* OneMax *under dynamic uniform constraint is*

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

Theorem 3.2. *The expected reoptimization time of the* (1+1) EA *on a linear profit function under dynamic uniform constraint is* $O(n^2 \log(B^* w_{\max}))$.

Due to limited space we omit the proofs of the theorems above and only give the high-level ideas here. The behavior of the (1+1) EA on linear functions has been examined in literature, see e.g. [? ? ?]. A detailed discussion focussing on constrained problems can be found in [?] and [?].

First, consider OneMax under dynamic uniform constraint. It is convenient for the analysis to use different potential functions depending on whether the current bit string maintained by the (1+1) EA is feasible or not. In the former case we use $|x|_1$, and $|x|_0$ in the latter. It is straightforward to show that in the infeasible region the expected drift is $\Omega(|x|_1/n)$, and $\Omega(|x|_0/n)$ when feasible. If $B \leq B^*$, the initial solution x_{org} is feasible and the Multiplicative Drift Theorem [?] gives an expected number of $O(n \log(|x_{\text{org}}|_0/|x^*|_0))$ generations to reach any optimal solution x^* to the new problem with bound B^*. If $B > B^*$, the reasoning is similar but with the optimization starting in the infeasible region. The first feasible solution sampled by the (1+1) EA might itself not be optimal; however, the number of additional iterations until optimality is immaterial.

In the general case the main difficulties arise once the (1+1) EA reaches a non-optimal solution x with $|x|_1 = B^*$. Now flipping a single 0-bit cannot improve on the fitness of x as the resulting offspring would be infeasible. Hence, the algorithm has to "swap" a 1 for a 0 of higher weight in a single mutation, accounting for the super-quadratic runtime.

4 ANALYSIS OF THE MOEA VARIANTS

In this section we analyze the Multi-Objective Evolutionary Algorithm and its single bit flip variant. The MOEA aims at overcoming the limitation of the (1+1) EA by maintaining a pool of candidate solutions that softens the influence of the cardinality bound B^*. On the other hand, the size of the population S can slow down the optimization process.

Theorem 4.1. *The expected reoptimization time of the* MOEA *on* OneMax *under dynamic uniform constraint is*

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

Proof. First, assume $B < B^*$. In order to employ drift analysis, we define the potential of the MOEA as $M = \min_{x \in S} |x|_0$. This value decreases as the maximum profit of solutions in S is maximized. Recall that the cardinality of populations S can reach $D + 1$.

Let M' denote the potential after one iteration, starting from M. By choosing the member of S with the minimum number of 0-bits, M, flipping one of them and nothing else, we get an expected drift of at least

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

The Multiplicative Drift Theorem [?] now implies an expected runtime of $O(nD \log((n - B)/(n - B^*)))$ iterations until the initial potential of $|x_{\text{org}}|_0 = n - B$ is reduced down to $n - B^*$.

For $B > B^*$ the reasoning is the same but with potential $M = \max_{x \in S} |x|_1$ and the roles of 1-bits and 0-bits inverted. □

While the reoptimization time on OneMax under dynamic uniform constraint suffers from the potentially large population of the MOEA, the next theorem shows that the algorithm can speed up the reoptimization on general linear profit functions significantly when D is small.

Theorem 4.2. *The expected reoptimization time of the MOEA on a linear profit function under dynamic uniform constraint is* $O(nD^2)$.

Proof. We only prove the case $B < B^*$, the other one is symmetric. For every integer $B \leq u < B^*$, let $x^{(u)} = \arg\max_{|x|_1 = u} P(x)$ be a solution of maximum profit among all solutions with Hamming weight u. Suppose $x^{(u)} \in S$, then choosing $x^{(u)}$ for mutation and flipping exactly one 0-bit of maximum weight creates solution $x^{(u+1)}$, optimal among all solutions with $u + 1$ 1-bits, within an expected number of $en(D + 1)$ generations. Observe that neither $x^{(u)}$ nor $x^{(u+1)}$ ever get deleted from S as they are maximal w.r.t. $\succcurlyeq_{\text{MOEA}}$. Summing over the waiting times for all $x^{(u+1)}$, starting from the initial solution $x_{\text{org}} = x^{(B)} \in S$, gives the claimed bound. □

The population size of the MOEA-S variant is bounded by a constant in order to avoid long waiting times until a certain candidate solution is chosen for mutation. In the following lemma we investigate the structure of the solutions in S.

LEMMA 4.3. *While no solution of Hamming weight $B^* - 1$ (B^*, if $B > B^*$) has been sampled by the* MOEA-S*, the population size is* 1. *If the population 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_{org}. While the Hamming *weight* of this solution is between B and $B^* - 1$ (B^*, if $B > B^*$), a mutation is accepted if it flips a 0-bit (1-bit). By the definition of the dominance relation $\succcurlyeq_{\text{MOEA-S}}$, the offspring replaces its parent. The condition in line ?? of Algorithm ?? is not met prior to S consisting of a single solution of weight B^*-1 and another 0 flipping (B^* and a 1-bit flip).

For the second part, consider the iteration in which said condition is fulfilled. Due to the RLS operator the initial two incomparable solutions have Hamming *distance* 1. May the difference be at position $1 \le i \le n$. 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^* (lower bound of $B^* - 1$) in line ?? of the algorithm or it creates an offspring that is already dominated. A flip at position i transforms the two solutions into each other. Therefore, the only possible way to update z is to flip a 0-bit in y at index $j \neq i$. Note that the resulting offspring y' now has Hamming weight one larger than y and is supposed to replace z (if it yields at least the same profit). String y' looks almost the same as z, the only difference is that $z_i = 1$ and $z_j = 0$ are now swapped. Similarly, y can only be updated by flipping a 1-bit at position j in z and thus switching the $y_i = 0$ and $y_j = 1$. Both updates preserve the distance. □

THEOREM 4.4. *The expected reoptimization time of the* MOEA-S *on* ONEMAX *under dynamic uniform constraint is*

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

PROOF. The proof is immediate from the observation that the MOEA-S behaves like RLS on ONEMAX, see Lemma ??. The runtimes differ from those in Theorem ?? only by a constant factor. □

THEOREM 4.5. *The expected reoptimization time of the* MOEA-S *on a linear profit function under dynamic uniform constraint is* $O(n \log D)$.

PROOF. W.l.o.g. the weights of the profit function P are ordered monotonically non-increasing, i.e., $w_{\max} = w_1 \ge w_2 \ge \cdots \ge w_n$. Thus, the initial solution is $x_{\text{org}} = 1^B 0^{n-B}$. We pessimistically assume that the target optimum is also unique and equals $1^{B^*}0^{n-B^*}$, i.e., $w_{B^*} > w_{B^*+1}$. For $x \in \{0,1\}^n$ and indices $1 \le k \le l \le n$, let $x_{[k,l]} = x_k x_{k+1} \dots x_l$ stand for the substring of x from position k to l, including. We call $x_{[1,B^*]}$ the *first block* and $x_{[B^*+1,n]}$ the *second block* of x.

We start the analysis at the first point in time at which set S contains two search points. We claim that Algorithm ?? then needs $O(n \log D)$ rounds in expectation to optimize the solution with Hamming weight B^*. This is indeed enough to establish the result since Theorem ?? implies that the starting point $|S| = 2$ can be reached in an initial phase of $O(n \log D)$ iterations.

It remains to prove the claim. First, suppose $B < B^*$. 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

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

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 easy to see that the potential is non-negative and that $M = 0$ 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 prove this, recall that z can only be updated by flipping a 0-bit in y, afterwards the two solutions differ exactly in the position i of the previous flip (Lemma ??). 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, $i \le B^*$ and $B^* - 1 = |y_{[1,B^*]}|_1 < |z_{[1,B^*]}|_1 = B^*$, as desired.

We now examine the update behavior of the solutions in S w.r.t. potential M. To this end, we only need to consider the cases in which the position i of the defect and the flipping position j are in different blocks. Suppose $i \le B^* < j$, thus $w_i > w_j$. Flipping a 0 in z would be discarded; however, flipping $z_j = 1$ is the same as trying to swap a 1 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 needs to be flipped. But this means swapping a 0 into the first block of z and would be discarded for reducing the profit. The case $j \le B^* < i$ is symmetric, using $w_j > w_i$.

We get from this analysis that the potential M cannot increase during the optimization. Also, in the worst case all the 1-bits that were added while S still was a singleton, fell in the second block. Hence, the number of 1s in the first block did not change from the initial point x_{org} and we get $M \le 2B^* - 1 - 2B = 2D - 1$. We now compute the expected drift of M towards 0. Let p denote the probability that in the current round the position in which y and z differ is in the first block, this corresponds to $i \le B^*$. 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 $i > B^*$, the potential is decreased once one of the $B^* - |y_{[1,B^*]}|_1$ 0-bits in the first block of y flips. Putting it all together yields

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

An application of the Multiplicative Drift Theorem [?] proves the claimed bound of $O(n \log D)$ for $B < B^*$.

For $B > B^*$, the reasoning is almost the same. Only that now the number of 1-bits does 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^*, D successful flips are necessary; for solution y with $|y| = B^* - 1$, $D + 1$ are necessary. Thus, the initial potential is $M \le 2B^* - 1 - (B^* - D - 1) - (B^* -$

$D) = 2D$. Once we have reached the two solutions $S = \{y, z\}$, the behavior is identical. □

5 ANALYSIS OF THE MOGA

The MOGA runs in a similar way to the MOEA, by constructing the solution of maximum profit among all solutions with Hamming weight $B+i+1$ based on the solution of maximum profit among all solutions with Hamming weight $B+i$ for $0 \le i \le D-1$ successively (if $B < B^*$), to get the optimal solution with Hamming weight B^*. The major difference between the two algorithms is the operation to construct the solution with Hamming weight $B + i + 1$ based on the solution with Hamming weight $B + i$. The MOEA uses the standard mutation operator flipping each bit with probability $1/n$. The MOGA incorporates the idea of the $(1+(\lambda, \lambda))$ GA, using the operations $\text{mutate}_\ell()$ and $\text{cross}_c(,)$ [?], to speed up the process to get the solution of maximum profit among all solutions with Hamming weight $B + i + 1$.

From the MOGA (given in Algorithm ??) and the definition of the dominance $\succcurlyeq_{\text{MOEA}}$, we can get that the size of the population S is bounded by $D + 1$. We will use this upper bound on the population size for our analysis.

5.1 OneMax with Dynamic Uniform Constraint

We start by analyzing MOGA on OneMax with a dynamic uniform constraint and lower bound the probability of achieving an improvement by a constant if the parameters are set in the right way.

Lemma 5.1. *Starting with a solution x of* OneMax *under dynamic uniform constraint with Hamming weight $A < B^*$, the probability of an iteration of the while-loop in the* MOGA *to get a solution y^* with Hamming weight $A + 1$ is greater than a constant $C > 0$, if $p = \frac{\lambda}{n}$, $c = \frac{1}{\lambda}$, and $\lambda = \sqrt{n/(n - |x|_1)}$.*

Proof. In the following discussion, we first analyze the probability without considering the parameter setting.

To get a solution y^* with Hamming weight $A+1$, it is a necessary condition that the solution x^* obtained by the mutation phase is a valid offspring of x, that is, there is an index j $(1 \le j \le n)$ such that $x^*_j = 1$ and $x_j = 0$. The probability that the offspring $x' = \text{mutate}_\ell(x)$ of x is not a valid offspring of x, is

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

Thus the probability that none of the λ offspring is a valid offspring of x is

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

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

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

Assume that the solution x^* obtained by the mutation phase is a valid solution of x. Thus for $y^{(i)} = \text{cross}_c(x, x^*)$, the probability that $|y^{(i)}|_1 = |x|_1 + 1$ is at least $c(1-c)^{\ell-1}$, indicating that the probability $|y^*|_1 = |x|_1 + 1$ is at least

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

By the analysis above, we have that the probability of an iteration of the while-loop to get a solution y^* with Hamming weight $A + 1$ 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 parameter setting into the analysis of the probability. Denote by L the random variable sampled by $\text{Bin}(n, p)$, and denote by K the success to get a solution y^* with Hamming weight $A + 1$. Then we have

$$Pr[K] \ge \sum_{\ell=\lceil\lambda/2\rceil}^{\lfloor 3\lambda/2 \rfloor} Pr[K|L = \ell] \cdot Pr[L = \ell],$$

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

Since $\lambda \ge 2$, $c = \frac{1}{\lambda}$, and that we only consider the values $\ell \in [\lambda/2, 3\lambda/2]$, we can get

$$\begin{aligned}\left(1 - c\,(1-c)^{\ell-1}\right)^{\lambda} &\le \left(1 - \frac{1}{\lambda}\left(1 - \frac{1}{\lambda}\right)^{\frac{3\lambda}{2}}\right)^{\lambda} \\ &\le \left(1 - \frac{1}{8\lambda}\right)^{\lambda} \le e^{-\frac{1}{8}}.\end{aligned}$$

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

$$1 - \left(\frac{|x|_1}{n}\right)^{\ell\lambda} \ge 1 - \left(\frac{|x|_1}{n}\right)^{\frac{\lambda^2}{2}} \ge 1 - e^{-\frac{1}{2}}.$$

Thus $Pr[K|L = \ell]$ is greater than a positive constant, indicating that $Pr[K]$ is also bounded away from 0. □

Lemma 5.2. *Starting with a solution x of* OneMax *under dynamic uniform constraint with Hamming weight $A > B^*$, the probability of an iteration of the while-loop in the* MOGA *to get a solution y^* with Hamming weight $A - 1$ is greater than a constant $C > 0$, if $p = \frac{\lambda}{n}$, $c = \frac{1}{\lambda}$, and $\lambda = \sqrt{n/|x|_1}$.*

Proof. The proof runs in a way similar to that of Lemma ??. First, we analyze the probability without considering the parameter setting.

To get a solution y^* with Hamming weight $A-1$, it is a necessary condition that the solution x^* obtained by the mutation phase is a valid offspring of x, i.e., there is an index j $(1 \le j \le n)$ such that $x^*_j = 0$ and $x_j = 1$ (since $|x|_1 > B^*$). The probability that x^* is a valid offspring of x is at least

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

Assume that the solution x^* obtained by the mutation phase is a valid offspring of x. For the crossover phase, the probability that $|y^*|_1 = |x|_1 - 1$ is at least

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

Therefore the probability of an iteration of the while-loop to get a solution y^* with Hamming weight $A-1$ is at least

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

The analysis of the probability incorporating the parameter setting is almost the same as that given in Lemma ??. It is not hard to get that the probability of an iteration of the while-loop in the MOGA to get a solution y^* with Hamming weight $A-1$ is bounded away from 0. □

Theorem 5.3. *The expected reoptimization time of the* MOGA *on* OneMax *under dynamic uniform constraint is* $O(\sqrt{n}D^{\frac{3}{2}})$ *if* $p=\frac{\lambda}{n}$, $c=\frac{1}{\lambda}$, *and* $\lambda=\sqrt{n/(n-|x|_1)}$ *when* $B^*>B$, *or* $p=\frac{\lambda}{n}$, $c=\frac{1}{\lambda}$, *and* $\lambda=\sqrt{n/|x|_1}$ *when* $B^*<B$.

Proof. We start with the case $B<B^*$. Since the size of the population is bounded by $D+1$, by Lemma ??, the MOGA takes expected time $O(D\lambda)=O(D\sqrt{n/(n-B)})$ to get a solution with Hamming weight $B+1$ if starting with x_{org} (including the fitness evaluations in Crossover phase).

By iteratively applying the analysis above, the MOGA takes expected time $O(D\sum_{i=B}^{B^*-1}\sqrt{\frac{n}{n-i}})=O(\sqrt{n}D^{\frac{3}{2}})$ to find a solution with Hamming weight B^*, because

$$\begin{aligned}\sum_{i=B}^{B^*-1}\sqrt{\frac{1}{n-i}} &\le \int_{B}^{B^*}\sqrt{\frac{1}{n-i}}\,\mathrm{d}i\\ &= 2\sqrt{n-B}-2\sqrt{n-B^*}\le 2\sqrt{D}.\end{aligned}$$

For the case that $B>B^*$, using the analysis similar to that given above and Lemma ??, we can get that the MOGA takes expected time $O(D\sqrt{n}\cdot\sum_{i=B^*+1}^{B}\sqrt{\frac{1}{i}})=O(\sqrt{n}D^{\frac{3}{2}})$ to find a solution with Hamming weight B^*, because

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

□

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.

Lemma 5.4. *Starting with a solution x of a linear profit function under dynamic uniform constraint that has the maximum profit among all solutions with Hamming weight $A<B^*$, the probability of an iteration of the while-loop in the* MOGA *to get a solution y^* that has the maximum profit among all solutions with Hamming weight $A+1$ is $\Omega(n^{-1/2})$, if $p=\frac{\lambda}{n}$, $c=\frac{1}{\lambda}$, and $\lambda=\sqrt{n}$.*

Proof. The proof runs in a way similar to that of Lemma ??. We first analyze the probability without considering the parameter setting.

Let j be the index of the 0-bit in x such that w_j is greater than any element in $\{w_i|x_i=0, 1\le i\le n\}$. To get the solution y^* that has the maximum profit among all solutions with Hamming weight $A+1$, it is a necessary condition that $x_j^*=1$, where x^* is the solution obtained by the mutation phase. For any offspring $x'=\text{mutate}_\ell(x)$ of x, the probability that $x'_j=0$ is $\prod_{t=0}^{\ell-1}\frac{n-1-t}{n}$. Thus the probability for $x_j^{(i)}=0$ for any $x^{(i)}\in\{x^{(1)},\dots,x^{(\lambda)}\}$ is

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

Assume that we have gotten an offspring whose j-th bit is 1 during the mutation phase. Since it may not be the unique valid offspring of x in $\{x^{(1)},\dots,x^{(\lambda)}\}$, the probability that the solution is chosen as x^* is at least $1/\lambda$. Therefore, the event $x_j^*=1$ occurs with probability $\frac{1}{\lambda}(1-(\frac{n-1}{n})^{\ell\lambda})$.

Assume that the j-th bit of the solution x^* obtained by the mutation phase is 1. Again we consider the event to sample the maximum-profit solution y^* (among all strings with Hamming weight $A+1$), based on x^*. For $y^{(i)}=cross_c(x,x^*)$, the probability that $y^{(i)}=y^*$ is at least $c(1-c)^{\ell-1}$. Therefore, the probability to get y^* for the crossover phase is at least $1-(1-c(1-c)^{\ell-1})^{\lambda}$.

Summarizing above analysis, we have that the probability of an iteration of the while-loop to get the solution y^* that has the maximum profit among all solutions with Hamming weight $A+1$ is at least

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

Combing above conclusions and the analysis given in Lemma ??, it is not hard to infer the probability of an iteration of the while-loop in the MOGA to sample solution y^* having the maximum profit among all solutions with Hamming weight $A+1$, it is $\Omega(n^{-1/2})$. □

Using similar ideas as in the previous proof, we show the lemma below for the case that the constraint bound decreases.

Lemma 5.5. *Starting with a solution x of a linear profit function under dynamic uniform constraint that has the maximum profit among all solutions with cost $A>B^*$, the probability of an iteration of the while-loop in the* MOGA *to get a solution y^* that has the maximum profit among all solutions with Hamming weight $A-1$ is $\Omega(n^{-1/2})$, if $p=\frac{\lambda}{n}$, $c=\frac{1}{\lambda}$, and $\lambda=\sqrt{n}$.*

Finally, we show the upper bound for MOGA on linear functions with dynamic uniform constraints.

Theorem 5.6. *Setting $p=\frac{\lambda}{n}$, $c=\frac{1}{\lambda}$, and $\lambda=\sqrt{n}$, the expected reoptimization time of the* MOGA *on a linear profit function under dynamic uniform constraint is $O(nD^2)$*

Proof. We first consider the case that $B<B^*$. Since the population size is bounded by $D+1$, by Lemma ??, the MOGA takes expected time $O(D\lambda\sqrt{n})=O(nD)$ to get a solution that has the maximum profit among all solutions with Hamming weight $B+1$, if starting with x_{org}. By iteratively applying the above analysis, the MOGA takes expected time $O(nD^2)$ to get the optimal solution with Hamming weight B^*. The analysis for the case that $B>B^*$ is almost the same as that for the case $B<B^*$. □

6 CONCLUSION

Linear functions play a central role in the area of runtime analysis of evolutionary computing techniques. In this paper, we have investigated linear functions under dynamic uniform constraints. Our results show that various types of evolutionary algorithms are efficient in recomputing the optimal solution if the constraint bound changes. In particular, the algorithms achieve faster recomputation than optimizing from scratch. Our analysis of population-based EAs have shown that using a population including infeasible solutions is helpful as it avoids the necessity of 2-bit flips in the case of linear functions. Furthermore, we have shown that the use of the $(1+(\lambda,\lambda))$ GA helps to achieve better upper bounds for the problems under consideration in some cases.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the German Research Foundation under grant agreement FR2988 (TOSU) and the Australian Research Council under grant agreements DP140103400 and DP160102401.

REFERENCES

[1] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, volume 1. World Scientific, 2011.

[2] B. Doerr and C. Doerr. Optimal Parameter Choices Through Self-Adjustment: Applying the 1/5-th Rule in Discrete Settings. In *Proc. of GECCO'15*, pages 1335–1342, 2015.

[3] B. Doerr, C. Doerr, and F. Ebel. From Black-box Complexity to Designing New Genetic Algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[4] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative Drift Analysis. *Algorithmica*, 64:673–697, 2012.

[5] S. Droste, T. Jansen, and I. Wegener. On the Analysis of the (1+1) Evolutionary Algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[6] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.

[7] T. Friedrich, T. Kötzing, J. A. G. Lagodzinski, F. Neumann, and M. Schirneck. Analysis of the (1+1) EA on Subclasses of Linear Functions under Uniform and Linear Constraints. In *Proc. of FOGA'17*, pages 45–54, 2017.

[8] T. Jansen. *Analyzing Evolutionary Algorithms - The Computer Science Perspective.* Natural Computing Series. Springer, 2013.

[9] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[10] T. Kötzing, A. Lissovoi, and C. Witt. (1+1) EA on Generalized Dynamic OneMax. In *Proc. of FOGA'15*, pages 40–51, 2015.

[11] S. Kratsch and F. Neumann. Fixed-parameter evolutionary algorithms and the vertex cover problem. *Algorithmica*, 65(4):754–771, 2013.

[12] E. Mezura-Montes and C. A. C. Coello. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[13] H. Mühlenbein. How Genetic Algorithms Really Work: Mutation and Hillclimbing. In *Proc. of PPSN'92*, volume 92, pages 15–25, 1992.

[14] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[15] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity.* Springer, 2010.

[16] C. Witt. Tight Bounds on the Optimization Time of a Randomized Search Heuristic on Linear Functions. *Combinatorics, Probability and Computing*, 22:294–318, 2013.

[17] Y. Zhou and J. He. A runtime analysis of evolutionary algorithms for constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 11:608–619, 2007.