HAL
open science

Fast re-optimization via structural diversity

Benjamin Doerr, Carola Doerr, Frank Neumann

To cite this version:

Benjamin Doerr, Carola Doerr, Frank Neumann. Fast re-optimization via structural diversity. The Genetic and Evolutionary Computation Conference, Jul 2019, Prague, Czech Republic. pp.233-241, 10.1145/3321707.3321731 . hal-02175763

HAL Id: hal-02175763

https://hal.sorbonne-universite.fr/hal-02175763v1

Submitted on 14 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fast Re-Optimization via Structural Diversity

Benjamin Doerr
École Polytechnique, CNRS, LIX
Palaiseau, France

Carola Doerr
Sorbonne Université, CNRS, LIP6
Paris, France

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

ABSTRACT

When a problem instance is perturbed by a small modification, one would hope to find a good solution for the new instance by building on a known good solution for the previous one. Via a rigorous mathematical analysis, we show that evolutionary algorithms, despite usually being robust problem solvers, can have unexpected difficulties to solve such re-optimization problems. When started with a random Hamming neighbor of the optimum, the (1+1) evolutionary algorithm takes $\Omega(n^2)$ time to optimize the LeadingOnes benchmark function, which is the same asymptotic optimization time when started in a randomly chosen solution. There is hence no significant advantage from re-optimizing a structurally good solution.

We then propose a way to overcome such difficulties. As our mathematical analysis reveals, the reason for this undesired behavior is that during the optimization structurally good solutions can easily be replaced by structurally worse solutions of equal or better fitness. We propose a simple diversity mechanism that prevents this behavior, thereby reducing the re-optimization time for LeadingOnes to $O(\gamma\delta n)$, where γ is the population size used by the diversity mechanism and $\delta \leq \gamma$ the Hamming distance of the new optimum from the previous solution. We show similarly fast re-optimization times for the optimization of linear functions with changing constraints and for the minimum spanning tree problem.

CCS CONCEPTS

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

1 INTRODUCTION

Evolutionary algorithms have been applied to many real-world problems in important areas such engineering [5] and supply chain management [3]. The underlying optimization problems arising in these real-world applications are usually not static, but have dynamic and stochastic components. Due to the ability to adapt to changing environments, evolutionary algorithms have been applied to various stochastic and dynamic problems [19, 22]. Furthermore, many optimization problems faced in practice occur repeatedly, with slight variations in the precise instance data. Instead of solving these instances from scratch, it is common practice to start the optimization in a solution that showed good quality for previously solved problems [29, 35].

Theoretical investigations regarding the behavior of evolutionary algorithms and other bio-inspired algorithms have been carried out for different types of dynamic problems (see [27] for an overview). This includes the MAZE problem for which difference in terms of performs for simple evolutionary algorithms and ant colony optimization approaches have been pointed out. In the context of dynamic constraints, linear functions with dynamically changing linear constraints have been investigated [31]. These investigations have been extended experimentally to the knapsack problem with a dynamic constraint bound. In addition, a general study of a simple evolutionary multi-objective approach for general costs functions with dynamic constraints has been provided in [26], which analyses the approximation behaviour of the algorithm in terms of the submodularity ratio of the problem. Other important studies included investigations on dynamic makespan scheduling [18], dynamic shortest paths [14] and variants of the dynamic vertex cover problem [20, 30]. A general method to analyze the runtime of evolutionary algorithms in dynamic contexts has been given in [4].

With this paper, we contribute to the theoretical understanding of evolutionary algorithms when dealing with re-optimization problems. As dynamic problems change over time, a previously high quality solution x^{old} may become unsuitable after a dynamic change has happened. We assume that a user of the algorithm is aware of the fact that a change has occurred. This is in contrast to classical dynamic problems where often the algorithm has to deal with changes automatically during the run and has to adapt to the changed problems. However, it should be noted that evolutionary algorithms for dynamic problems often incorporate a change detection mechanism [24]. After a change, the solution x^{old} might still be structurally quite close to a solution that is of high quality after the dynamic change has occurred. This is especially the case if only a few components have changed. Previously examined approaches have indirectly build on this by using a multi-objective formulation of the given problem where the constraint is treated as an additional objective [25, 26, 31].

We explore the use of a previously good solution in a more direct way by proposing a population-based approach that directly searches for improvements close to the previously best solution x^{old}. In our studies, we consider problems where the dynamic change is quantified by a parameter δ. It is often desirable not to deviate from a previously chosen solution that much in terms of design parameters as such changes might be difficult to implement. Therefore, we search for solutions after a given change has occurred that are close to the solution in the decision space prior to the change. We

present a simple evolutionary algorithm called $(\gamma+1)$ REA. It works with a diverse set of solutions at Hamming distance at most γ from a previously good solution x^{old}, where γ is a parameter of the algorithm. In order to have global search capabilities, it also keeps the best solution found for the considered problem at a time. The population of $(\gamma+1)$ REA contains for each i, $0 \le i \le \gamma$, the best-so-far solution at Hamming distance i to x^{old}. With this diversity mechanism, we aim at putting a stronger emphasis on exploring the neighborhood of the previous best solution.

We show the effectiveness of our approach on a wide range of optimization problems by rigorous runtime analyses [1, 12, 17]. Our analyses use common rigorous techniques from this area of research to show the working principles of our proposed method.

We start by investigating the classical LeadingOnes problem and consider the scenario that the problem is perturbed by flipping δ bits of the target bit string. We show that a solution of fitness at least as high as the best possible solution within Hamming distance $i \le \gamma+1$ to x^{old} is computed in expected time less than or equal to $2e(\gamma+1)in$. For Hamming distances $i > \gamma+1$, we bound the expected time to find such a solution from above by $2en^2$. Furthermore, we show a lower bound of $\Omega(n^2)$ for computing an optimal solution at Hamming distance $\delta \in [\gamma+2, n]$, that is, when the optimal solution is (just a little) further away from the starting solution than Hamming distance γ. This lower bound also holds when re-optimizing with the classic $(1+1)$ EA. These lower bounds show that it is indeed the proposed diversity mechanism that makes the difference between an easy re-optimization and a re-optimization that is not faster than optimizing from a random solution.

We then investigate the effectiveness of our approach on a constraint optimization problem where the constraint bound changes. Investigating our algorithm on the class of linear functions with a uniform constraint, we show that it re-computes an optimal solution in expected time $O(\gamma\delta n)$, where δ is the amount by which the constraint bound changes.

Finally, we investigate the minimum spanning tree problem. This classical combinatorial optimization problem has been subject to a wide range of theoretical investigations in the area of runtime analysis of bio-inspired computing [15, 16, 21, 23, 34]. We consider a dynamic version of the problem where either δ edges are added or removed from the current graph. Our results show that $(\gamma+1)$ REA is able to recover an optimal solution in time $O(\gamma\delta n)$ in both situations.

The paper is structured as follows. We introduce the algorithm and setting for dynamic changes in Section 2. In Section 3, we present our results for re-optimizing the LeadingOnes problem. We analyze linear functions with a dynamic uniform constraint in Section 4. We present the results for re-optimizing the minimum spanning tree problem in Section 5 and finish with some concluding remarks.

2 THE $(\gamma+1)$ RE-OPTIMIZATION EA

Our algorithm, the $(\gamma+1)$ Re-Optimization EA (REA), has as input a user-specified solution x^{old}. We typically assume that x^{old} was a solution of high quality for the function f^{old}.

We are concerned in this work with the situation in which the function f^{old} is perturbed by some change, resulting in a new objective function f. We study the time needed to recover a solution of quality at least $f^{\text{old}}(x^{\text{old}})$. That is, in the context of maximization problems, we study the number of function evaluations that are needed by $(\gamma+1)$ REA to generate a solution y with $f(y) \ge f^{\text{old}}(x^{\text{old}})$, and in the context of minimization problems we require a solution y with $f(y) \le f^{\text{old}}(x^{\text{old}})$.

Algorithm 1: The $(\gamma+1)$ REA for the re-optimization (here: maximization) of a function $f : \{0,1\}^n \to \mathbb{R}$, which emerged from the function f^{old} by a dynamic change.

1 **Input:** Solution x^{old};
2 **Initialization:**
3 $\quad x^0, x^* \leftarrow x^{\text{old}}$;
4 $\quad$ **for** $i = 1, 2, \ldots, \gamma+1$ **do** $x^i \leftarrow$ undefined, $f^i \leftarrow -\infty$;
5 **Optimization: for** $t = 1, 2, 3, \ldots$ **do**
6 $\quad$ Select parent x by choosing x^* with probability $1/2$ and uniformly at random from $\{x^i \mid i \in [0..\gamma+1]\} \setminus \{x^*\}$ otherwise;
7 $\quad$ Create y from x by flipping in each bit independently with probability $1/n$; // standard bit mutation
8 $\quad$ **if** $f(y) \ge f(x^*)$ **then** $x^* \leftarrow y$;
9 $\quad i \leftarrow \min\{H(y, x^{\text{old}}), \gamma+1\}$;
10 $\quad$ **if** $f(y) \ge f^i$ **then** $x^i \leftarrow y$, $f^i \leftarrow f(y)$;

Note that in this work we study both maximization and minimization problems. Algorithm 1 summarizes the $(\gamma+1)$ REA for maximization problems; we will describe it below. For minimization problems, only three changes are necessary: the f^i are initialized by ∞ in line 3, and the $\ge$-signs in lines 8 and 10 need to be exchanged for a $\le$-sign.

We quantify the difference between the old function f^{old} and the new function f by a parameter δ, which denotes the smallest distance at which a solution of quality at least $f^{\text{old}}(x^{\text{old}})$ exists. That is, there exists a solution y at Hamming distance $H(y, x^{\text{old}}) = \delta$ for which $f(y) \ge f^{\text{old}}(x^{\text{old}})$ and for all solutions y' with $H(y', x^{\text{old}}) < \delta$ it holds that $f(y') < f^{\text{old}}(x^{\text{old}})$. In our applications we assume that an upper bound $\gamma \ge \delta$ of this perturbation is known to the user (and set $\gamma = n$ otherwise).

Our algorithm stores for each i, $i \in [\gamma] := \{1, 2, \ldots, \gamma\}$, one best-so-far solution x^i of Hamming distance i to x^{old}. For notational convenience we define $x^0 := x^{\text{old}}$. In order to advance the search beyond the radius of γ (e.g., if we risk that the upper bound γ is too small), the algorithm also stores an additional search point $x^{\gamma+1}$ which is the best-so-far solution of Hamming distance greater than γ to x^{old}. The points x^i, $i \in [\gamma+1]$ are initialized as undefined, the best function value at distance i, f^i, as $-\infty$.

In every iteration the $(\gamma+1)$ REA first selects a parent individual x from which one offspring y will be generated. The parent is chosen through a biased random selection. With probability $1/2$ we select as x the search point x^* with the best-so-far objective value. We choose x uniformly at random from $\{x^i \mid i \in [0..\gamma+1] := \{0\} \cup [\gamma]\} \setminus \{x^*\}$ otherwise. That is, each $x^i \ne x^*$ is selected with probability $1/(2(\gamma+1))$. A new solution candidate y is created from the selected parent x by standard bit mutation with mutation rate

$p = 1/n$. If the Hamming distance $i = H(y, x^{\text{old}})$ of y to x^{old} is at most γ this offspring y replaces the previous best individual x^i at distance i if it is at least as good, i.e., if and only if $f(y) \geq f(x^i)$. For offspring y with $H(y, x^{\text{old}}) > \gamma$, the selection is made between y and $x^{\gamma+1}$, by applying the same selection rules as in the case $i \leq \gamma$.

Note that, despite the name, the $(\gamma + 1)$ REA maintains a population size of size $\gamma + 2$. We use $(\gamma + 1)$ REA for notational convenience.

The biased parent selection of the $(\gamma + 1)$ REA is meant to avoid too severe slow-downs when the upper bound γ of the perturbation is large. With uniform parent selection the slowdown caused by sub-optimal search points can be as large as proportional to the population size $\gamma + 2$. With the biased selection, in contrast, each step simulates, with probability 1/2, a regular $(1 + 1)$ EA.

The advantages of storing the points x^i, $i \in [0..\gamma + 1]$ will be motivated in the next section, using the example of re-optimizing the LeadingOnes problem as illustration.

3 RE-OPTIMIZING LEADINGONES

As a first example to demonstrate the working principles of the $(\gamma + 1)$ REA we regard the LeadingOnes problem, one of the most classical benchmark problems in the theory of evolutionary computation. It has the characteristic property that the decision variables can only be optimized sequentially, that is, only when the first i variables are set to the optimal value the $(i+1)$-st variable has an influence on the fitness. Such behaviors are common in non-artificial problems, see, e.g., the examples in [9, Section 4] or [10, Section 3.2] for two different shortest path problems.

For a "target string" $z \in \{0, 1\}^n$ and a permutation σ of the index set $[n]$, the LeadingOnes function $f_{z,\sigma}$ is defined via

$$f_{z,\sigma}(x) := \max\{j \in [0..n] \mid \forall k \in [j] : x_{\sigma(k)} = z_{\sigma(k)}\}$$

for all $x \in \{0, 1\}^n$. Our aim is maximizing the LeadingOnes functions. We note that z is the unique global maximum of $f_{z,\sigma}$, regardless of σ. This problem is referred to as LeadingOnes because traditionally only the non-permuted instance $f_{(1,\ldots,1),\text{id}}$ with target string $(1, \ldots, 1)$ was studied. This function simply returns the number of initial (*leading*) ones of each solution candidate. Many EAs, and including our $(\gamma + 1)$ REA, show exactly the same performance on any of the instances $f_{z,\sigma}$ and it thus suffices to study this particular instance with $z = (1, \ldots, 1)$ and σ being the identity function id.

When perturbing the LeadingOnes function, small changes can result in large changes in fitness: if we assume that x^{old} is an optimal solution for $f_{z,\sigma}$, i.e., $x^{\text{old}} = z$, then changing z to z' by flipping the i-th bit of z gives a new fitness value $f_{z',\sigma}(x^{\text{old}}) = i - 1$. We also note that the new optimal solution, which is z', is at Hamming distance one of x^{old}. However, all the solutions which differ from x^{old} only in positions of index greater than i have the same fitness value $i - 1$. In consequence, the $(1 + 1)$ EA performs a random walk on this plateau until it eventually flips the i-th bit. When i is small, it is likely that at this point the $(1 + 1)$ EA has lost track of the previously good entries in the positions $j > i$, so that it then has to recover significant parts of the tail of z. This unfavorable behavior of the $(1 + 1)$ EA motivates our decision to store for each Hamming distance $i \in [0..\gamma]$ a best-so-far solution x^i, and to assign positive probability of selecting x^i as parent individual even if $f(x^i)$ is strictly smaller than the current-best fitness $f(x^*)$. In the situation described above, in which only the i-th bit has been flipped, the $(\gamma + 1)$ REA always has a chance of at least $1/(2(\gamma + 1))$ of selecting x^{old} as parent individual. Conditioning on x^{old} being the selected parent, the probability to sample as offspring the new optimal solution z' is at least $1/(en)$, since exactly the i-th bit needs to be flipped. The expected optimization time of the $(\gamma + 1)$ REA is hence at most $2e(\gamma + 1)n$, whereas the expected re-optimization time of algorithms not using the diversity mechanism can be considerably larger, cf. Lemma 3.4.

Summary of our results for LeadingOnes. In the remainder of this section we formalize the observations made above. In Section 3.1 we prove an upper bound for the expected re-optimization time of the $(\gamma + 1)$ REA on LeadingOnes, which in particular shows that the $(\gamma + 1)$ REA finds the best solution that is in distance $i \leq \gamma + 1$ from x^{old} in time $O(\gamma i n)$ only. We complement this results by a lower bound for the performance of the $(1 + 1)$ EA (Lemma 3.4 in Section 3.2), which shows that for this algorithm the re-optimization times are $\Omega(n^2)$ when the fitness of x^{old} is at most $n/2$, even when the Hamming distance of x^{old} and x^{opt} is small. In particular, when started with a random Hamming neighbor of the optimum, the $(1 + 1)$ EA still needs $\Omega(n^2)$ iterations to find the optimum.

These bounds show that the $(\gamma + 1)$ REA is significantly faster in solving re-optimization problems of the LeadingOnes type. We also provide a lower bound for the $(\gamma + 1)$ REA (Theorem 3.3) which shows that $H(x^{\text{old}}, x^{\text{opt}}) \leq \gamma + 1$ is a necessary condition for a fast re-optimization: Already from $H(x^{\text{old}}, x^{\text{opt}}) = \gamma + 2$ on the $(\gamma + 1)$ REA can have an at least quadratic expected re-optimization time. Upper and lower bounds thus illustrate the trade-off between choosing a too large γ, which results in a slow-down that is linear in γ, and a too small γ, which results in an at least quadratic re-optimization time. On the other hand, our results also prove that the $(\gamma + 1)$ REA with a too small γ still has an expected optimization time of at most $2en^2$, which is not much worse than the known $\frac{1}{2}en^2$ upper bound for the $(1 + 1)$ EA.

3.1 Upper Bound for LeadingOnes

Theorem 3.1 provides an upper bound for the re-optimization time of the $(\gamma + 1)$ REA on the LeadingOnes problem in which the target string has been modified from x^{old} to x^{opt}. Both situations of an accurate and a too small upper bound γ on the perturbation $\delta = H(x^{\text{old}}, x^{\text{opt}})$ are covered by this bound. More precisely, the theorem shows that regardless of γ and δ the $(\gamma + 1)$ REA has an expected re-optimization time that is at most quadratic. When $\gamma \geq \delta - 1$ the expected re-optimization time is $O(\gamma n)$. Since for this problem it provides no additional difficulties, we not only compute the expected runtimes, but we follow the approach suggested in [6] and first show a domination statement and then derive from that the expected runtime and a tail bound.

Theorem 3.1. *Let f be a generalized LeadingOnes function with unique optimum x^{opt}. Let $x^{\text{old}} \in \{0, 1\}^n$. For all $i \in [0..H(x^{\text{old}}, x^{\text{opt}})]$, let T_i be the the number of function evaluations that the $(\gamma + 1)$ REA needs to find a solution y with $f(y) \geq \max\{f(y') \mid y' \in \{0, 1\}^n, H(y', x^{\text{old}}) \leq i\}$.*

(1) *If $i \le \gamma+1$, then T_i is dominated by a sum of i independent geometric distributions with success rate $\frac{1}{2e(\gamma+1)n}$. Consequently,*

$$E[T_i] \le 2e(\gamma+1)in =: \mu^+,$$

$$\Pr[T_i \ge (1+\varepsilon)\mu^+] \le \exp\left(-\frac{\varepsilon^2 i}{2(1+\varepsilon)}\right) \text{ for all } \varepsilon \ge 0.$$

(2) *Regardless of i, the time T_i is dominated by a sum of n independent geometric random variables with success rate $\frac{1}{2en}$. Consequently,*

$$E[T_i] \le 2en^2 =: \mu^+,$$

$$\Pr[T_i \ge (1+\varepsilon)\mu^+] \le \exp\left(-\frac{\varepsilon^2 n}{2(1+\varepsilon)}\right) \text{ for all } \varepsilon \ge 0.$$

When $\gamma \ge \delta - 1$, the expected re-optimization time of the $(\gamma+1)$ REA on the modified LeadingOnes function is thus at most $\min\{2e(\gamma+1)\delta n, 2en^2\}$, provided that x^{old} was an optimal solution for f^{old}.

Proof. By the symmetry of all operators used in the $(\gamma+1)$ REA, we can assume without loss of generality that the optimum of f is $x^{\text{opt}} = (1,\dots,1)$. Let $0 \le i \le H(x^{\text{old}}, x^{\text{opt}})$. We first consider the case that $i \le \gamma+1$. Due to the nature of the LeadingOnes function, there is a unique search point $x^{i,*}$ in $\{y \in \{0,1\}^n \mid H(y, x^{\text{old}}) = i\}$ with maximal fitness. This search point is equal to x^{old} in all bit positions except the first i positions in which x^{old} is zero. Hence $H(x^{i,*}, x^{\text{old}}) = i$. If $i \le \gamma$, then let T_i^* be the iteration in which the program variable x^i takes the value $x^{i,*}$. For $i = \gamma+1$, let $T_i^* = T_i$. Note that T_i^* stochastically dominates T_i for all $i \le \gamma+1$, so it suffices to show our claim for T_i^* instead of T_i.

We use a fitness level argument to estimate T_i^*. If $i \le \gamma$, then for all $j \le i$, we say that the algorithm is in state j if $x^j = x^{j,*}$ and, if $j < i$, also $x^{j+1} \ne x^{j+1,*}$ holds. For $i = \gamma+1$, we say that the algorithm is in state

- $j < \gamma$ if $x^j = x^{j,*}$ and $x^{j+1} \ne x^{j+1,*}$,
- $j = \gamma$, when $x^j = x^{j,*}$ and $f(x^*) < \max\{f(y') \mid y' \in \{0,1\}^n, H(y', x^{\text{old}}) \le \gamma+1\}$,
- $j = \gamma+1$ if $f(x^*) = \max\{f(y') \mid y' \in \{0,1\}^n, H(y', x^{\text{old}}) \le \gamma+1\}$.

In both cases $i \le \gamma$ and $i = \gamma+1$, we see that when the algorithm is in state $j < i$, then with probability at least $\frac{1}{2(\gamma+1)}$ it chooses x^j as parent and (in this case) with probability $\frac{1}{n}(1-\frac{1}{n})^{n-1} \ge \frac{1}{en}$, flips exactly the unique bit that $x^{j,*}$ and $x^{j+1,*}$ differ in. Hence each iteration in state j has a probability of at least $\frac{1}{2(\gamma+1)en}$ of ending in a higher state. By the classic fitness level theorem [33] we obtain the bound $E[T_i^*] \le 2e(\gamma+1)in$. By the domination version of the fitness level theorem [6, Theorem 2] we also obtain the domination result and the tail bound, for the latter using a Chernoff bound for sums of independent geometric random variables [6, Theorem 3(i)].

When i is arbitrary, and not necessarily at most $\gamma+1$, we can still apply the classic fitness level method to the fitness of the best solution x^*. Note that when x^* has some fitness j, then with probability at least $\frac{1}{2}$ this x^* is chosen as parent and with probability $\frac{1}{n}(1-\frac{1}{n})^{n-1} \ge \frac{1}{en}$ exactly the $(j+1)$-st bit is flipped. The resulting offspring y has a fitness greater than x^* and thus replaces x^* (and possibly some x^k). Hence in each iteration, with probability at least $\frac{1}{2en}$ the fitness of x^* increases. Again, the fitness level theorems give the claimed bounds for the time T to find the optimum. Since T dominates all T_i, the claims for T_i are proven. □

3.2 Lower Bound for LeadingOnes

We now show that the (1 + 1) EA without diversity mechanism can have a quadratic runtime to optimize LeadingOnes even when started with a Hamming neighbor of the optimum. In fact, many Hamming neighbors lead to this runtime, so this result also holds when starting with a random Hamming neighbor. Using similar arguments, we also show that the requirement $i \le \gamma+1$ in the first part of Theorem 3.1 cannot be relaxed. Already for a Hamming distance of $\gamma+2$, we have an expected quadratic optimization time.

The reason for these high runtimes is as follows. For the (1+1) EA, the initial structurally good solution is easily replaced by other solutions of same fitness, which are structurally further away from the optimum. By this the advantage of starting with a good solution is lost. When the optimum has Hamming distance at least $\gamma+2$ from x^{old}, then (i) the $(\gamma+1)$ REA finds it hard to generate the optimum from one of the x^k, $k \in [0..\gamma]$, as these have a Hamming distance at least 2 from the optimum, and (ii) the $(\gamma+1)$ REA also finds it hard to find the optimum via optimizing x^* as this search point again quickly becomes structurally distant from the optimum.

The rough reason for structurally good solutions moving away from the optimum is that bits with higher index than the current fitness plus one are neutral, that is, they are subject to mutation, but have no influence on the fitness and can therefore not bias the selection. For such bits, independent of their initial values in x^{old}, the probabilities of being zero or one converge to 1/2. This was first shown in [8, proof of Theorem 10] and later exploited in several analyses how evolutionary algorithms optimize the LeadingOnes function [2, 11, 13, 28, 32].

For the sake of completeness, we quickly repeat the statement in [8] and its proof (where we note that in [8] apparently the binomial coefficients were forgotten in the proof). We note that an essentially identical result (their assumption $t \ge n\ln(n)$ can be freely omitted) was independently proven in [13] with identical arguments).

Lemma 3.2. *Let $X_0, X_1, \dots$ be a sequence of binary random variables such that $\Pr[X_t = X_{t-1}] = 1 - \frac{1}{n}$ and $\Pr[X_t = 1 - X_{t-1}] = \frac{1}{n}$ independently for all $t \ge 1$. Then*

$$\Pr[X_t = X_0] = \tfrac{1}{2} + \tfrac{1}{2}(1-\tfrac{2}{n})^t,$$
$$\Pr[X_t \ne X_0] = \tfrac{1}{2} - \tfrac{1}{2}(1-\tfrac{2}{n})^t.$$

Proof. We have

$$\begin{aligned}
&\Pr[X_t = X_0] - \Pr[X_t \ne X_0] \\
&\quad= \sum_{\substack{i=0\\2\mid i}}^{t} \binom{t}{i}\left(\frac{1}{n}\right)^i\left(1-\frac{1}{n}\right)^{t-i} - \sum_{\substack{i=0\\2\nmid i}}^{t} \binom{t}{i}\left(\frac{1}{n}\right)^i\left(1-\frac{1}{n}\right)^{t-i} \\
&\quad= \sum_{i=0}^{t} \binom{t}{i}\left(-\frac{1}{n}\right)^i\left(1-\frac{1}{n}\right)^{t-i} = \left(-\frac{1}{n} + \left(1-\frac{1}{n}\right)\right)^t \\
&\quad= \left(1-\frac{2}{n}\right)^t.
\end{aligned}$$

The claims follow from $\Pr[X_t = X_0] + \Pr[X_t \neq X_0] = 1$ and elementary transformations. □

We are now ready to state and prove our lower bound result. For the ease of presentation, we only cover the case that $\gamma \leq \frac{n}{4} - 2$. Since constant factors play no important role in the proof, it is immediately clear from the proof that this condition could be relaxed to $\gamma \leq (1-\varepsilon)n$ for any constant $\varepsilon > 0$. Further, we are optimistic that the result holds for all values of γ. However, we feel that the case of values of γ that are linear in n is not interesting enough to justify the extra effort. Note that for $\gamma = \Omega(n)$ and $\delta \geq \gamma + 2$ our starting solution has a linear Hamming distance from the optimum. This can hardly be seen as re-optimization from a solution close to the optimum.

Theorem 3.3. *Let $\gamma \leq \frac{1}{4}n - 2$. Let f be the LeadingOnes function with unique optimum $x^{\text{opt}} = (1, \ldots, 1)$. For all $\delta \in [\gamma + 2..n]$ there is an $x^{\text{old}} \in \{0,1\}^n$ with $H(x^{\text{old}}, x^{\text{opt}}) = \delta$ such that the expected time the $(\gamma + 1)$ REA started with x^{old} takes to find the optimum of f is $\Omega(n^2)$.*

Proof. Let x^{old} be the search point defined by $x_1^{\text{old}} = \cdots = x_\delta^{\text{old}} = 0$ and $x_{\delta+1}^{\text{old}} = \cdots = x_n^{\text{old}} = 1$. Note that $H(x^{\text{old}}, x^{\text{opt}}) = \delta$.

Consider the first iteration t_0 in which the search point stored in x^* reaches a fitness of at least $\gamma + 2$. Let x be the parent chosen (which by assumption has a fitness of at most $\gamma + 1$) and let y be the offspring generated in this iteration (which will end up in x^*). Since the algorithm as mutation operation flips bits independently with probability $\frac{1}{n}$ and since we know $f(y) \geq \gamma + 2$, we have $y_1 = \cdots = y_{\gamma+2} = 1$ and all further bits are obtained from the corresponding bit of x by flipping it with probability $\frac{1}{n}$. Let $I_0 := \{i \in [\gamma + 3..\lceil \frac{1}{2}n \rceil] \mid x_i = 0\}$ and $I_1 := \{i \in [\gamma + 3..\lceil \frac{1}{2}n \rceil] \mid x_i = 1\}$. We compute

$$\begin{aligned}
\Pr[f(y) \geq \tfrac{1}{2}n] &= \prod_{i=\gamma+3}^{\lceil \frac{1}{2}n \rceil} \Pr[y_i = 1] \\
&= \prod_{i \in I_0} \Pr[y_i = 1] \prod_{i \in I_1} \Pr[y_i = 1] \\
&= (\tfrac{1}{n})^{|I_0|} (1 - \tfrac{1}{n})^{|I_1|} \leq (1 - \tfrac{1}{n})^{n/2 - \gamma - 2} \\
&\leq (1 - \tfrac{1}{n})^{n/4} \leq e^{-1/4}.
\end{aligned}$$

Let us condition on $f(y) \leq \frac{1}{2}n$ in the following (and recall that we have this event with probability at least $1 - e^{-1/4} \geq 0.2$). We first argue that we can assume that whenever in the following $0.1n^2$ iterations we choose an x^k, $k \in [0..\gamma]$, as parent, then the offspring does not replace the current value of x^*. Since the search point stored by the algorithm in x^k, $k \in [0..\gamma]$, at all times has a Hamming distance of at least $\gamma + 2 - k \geq 2$ from any search point with fitness $\gamma + 2$ or larger, the probability that an x^k is mutated to a search point with fitness at least the one of x^*, is at most n^{-2}. By a simple union bound over $0.1n^2$ iterations, we obtain that with probability at least 0.9, in no iteration of the time interval $I = [t_0 + 1..t_0 + 0.1n^2]$ an offspring of an x^k makes it into x^*.

Taking also this assumption, we can ignore all iterations in the time interval I that use a parent different from x^* as they cannot generate the optimum and cannot interfere with the process on x^*. In the remaining iterations in I, the $(\gamma + 1)$ REA simulates a $(1 + 1)$ EA using x^* as population. By Lemma 3.4 below, the first $\frac{1}{16}n^2$ of these iterations (or fewer, if there are fewer such iterations in I) with constant probability do not create the optimum. Taking this and all assumptions taken on the way together, we see that with constant probability, the $(\gamma + 1)$ REA within $\frac{1}{16}n^2$ iterations does not find the optimum. Consequently, the expected optimization time is $\Omega(n^2)$.

□

We finish the proof of the main result by providing the missing ingredient that the $(1+1)$ EA with constant probability needs a quadratic number of function evaluations to optimize LeadingOnes even when initialized with an arbitrary search point of fitness at most $n/2$, that is, even when the initial search point is a Hamming neighbor of the optimum. This result might be of independent interest. Again, we did not try to optimize the constants, in particular, a quadratic runtime could also be shown when the initial fitness is as large as $(1 - \varepsilon)n$ for an arbitrarily small positive constant ε.

Lemma 3.4. *Consider a run of the $(1 + 1)$ EA on the LeadingOnes function f, initialized with an arbitrary search point x^0 such $f(x^0) \leq n/2$. Let T be the first iteration in which an optimal solution is generated. Then $\Pr[T \leq n^2/16] \leq \frac{1}{e} + \exp(-\Omega(n))$.*

Proof. Let t_0 be any number such that among the first t_0 iterations of the $(1 + 1)$ EA, in exactly $n - 1$ iterations, called *relevant iterations* in the following, an offspring y is created that agrees with the parent x in the first $f(x)$ bits. Note that the remaining $t_0 - (n-1)$ iterations create offspring worse than the parent, so that they have no influence on the optimization process except wasting time. In each relevant iteration, with probability exactly $1/n$ an offspring strictly better than the parent is generated (namely when the first missing bit is flipped). Hence with probability $(1 - \frac{1}{n})^{n-1} \geq 1/e$, none of the $n - 1$ relevant iterations creates a strict improvement. Let us condition on this event in the following.

We now regard in detail the search point x resulting from the first t_0 iterations. Put differently, we analyze the parent individual of iteration $t_0 + 1$. By our assumption, we have $f(x) = f(x^0)$. As discussed before, we can ignore the non-relevant iterations as they do not change the current individual. In each relevant iteration, the parent is replaced by the offspring, which was generated by flipping each bit greater than $f(x) + 1$ independently with probability $1/n$. Consequently, for $i \in [f(x) + 2..n]$, the value x_i is obtained from x_i^0 by $n - 1$ times independently flipping the bit value with probability $1/n$ (independently from the other bits).

This observation remains true (in an analogous fashion) for future generations $t \geq t_0$: Let x be the search point at the end of some iteration $t \geq t_0$ and assume by induction that it is such that for all $i \geq f(x) + 2$, the bit value x_i of the i-th bit is obtained from x_i^0 by flipping it some number $n_t \geq n - 1$ times independently with probability $1/n$. Let y be the offspring generated (from x) in iteration $t + 1$ and let x' be the outcome of the selection between x and y. Clearly, for all $i \geq f(x) + 2$, the bit value y_i of the i-th bit is obtained from x_i^0 by flipping it $n_t + 1$ times independently with probability $\frac{1}{n}$. If $f(y) < f(x)$, then y is discarded and our claim holds for x' since it holds for x. If $f(y) \geq f(x)$, then the first $f(y) + 2$ bits of y are determined by the fitness, but the remaining

bits have no influence on the decision to continue with y. Hence for all $i \geq f(x) + 2$, the value of y_i is obtained from x_i^0 by flipping it $n_t + 1$ times independently with probability $1/n$. Obviously, the same statement holds for x' in this case.

In summary, we see that under the assumption taken initially (which holds with probability at least $1/e$), in all iterations following iteration t_0, the parent individual x is such that all bits $i \in [f(x) + 2..n]$ independently have the distribution of taking the initial bit value x_i^0 and flipping it some number $n_t \geq n - 1$ of times independently with probability $\frac{1}{n}$. By Lemma 3.2, independent of the initialization of the bit value and independently for all $i \geq f(x) + 2$, we have $\Pr[x_i = 1] \leq \frac{1}{2} + \frac{1}{2}(1 - \frac{2}{n})^{n_t} \leq \frac{1}{2} + \frac{1}{2}(1 - \frac{2}{n})^{(n-1)} \leq \frac{1}{2} + \frac{1}{2}(1 - \frac{2}{n})^{n/2} \leq \frac{1}{2} + \frac{1}{2e} =: p \leq 0.7$.

We use this statement to describe the fitness gain in one iteration $t > t_0$. To have a positive fitness gain from a parent x, it is necessary that the first $f(x)$ bits do not flip and that bit $f(x) + 1$ does flip (we call this event a success). This happens with probability $(1 - \frac{1}{n})^{f(x)} \frac{1}{n} \leq \frac{1}{n}$. Since in this case, each further bit of the offspring is one with probability at most p, regardless of the outcomes of the other bits, the fitness gain in case of a success is dominated by a geometric distribution with parameter $1 - p$.

We finish the proof by showing that the total fitness gain X in the time interval $[t_0 + 1..t_0 + \frac{1}{6}n^2]$ with high probability is less than $\frac{1}{2}n$ and thus not sufficient to reach the optimum (recall that the fitness after iteration t_0 still was the initial fitness $f(x^0) \leq \frac{1}{2}n$). Since the probability for a success is at most $\frac{1}{n}$ regardless of what happened in the previous iterations, by Lemma 3 of [6] the number of successes in these $\frac{1}{16}n^2$ iterations is dominated by a sum of $\frac{1}{16}n^2$ independent binary random variables with success probability $\frac{1}{n}$. Applying a common Chernoff bound, e.g., the simple multiplicative bound in Theorem 10.1 of [7], we see that with probability $1 - \exp(n/48)$, the number of successes is at most $\frac{1}{8}n$. In this case, the fitness gain X is dominated by a sum of $\frac{1}{8}n$ independent geometric distributions with success probability $1 - p$. Hence $E[X] = \frac{1}{8(1-p)}n$ and, using a Chernoff bound for geometric random variables like Theorem 3 (i) of [6],

$$\Pr[X \geq \tfrac{1}{2}n] \leq \Pr[X \geq 1.2E[X]] \leq \exp\left(-\frac{(0.2n)^2}{2 \cdot \frac{1}{8}n(1 + 0.2n/n)}\right) = \exp(-\Omega(n)).$$

Hence, under our initial assumption of having no fitness gain in the first $n - 1$ iterations, with probability $1 - \exp(-\Omega(n))$ the expected runtime of the EA is more than $n^2/16$. Since the initial assumption was satisfied with probability $1/e$, the claim is proven. □

4 RE-OPTIMIZING LINEAR FUNCTIONS WITH MODIFIED UNIFORM CONSTRAINTS

The next example for which we analyze the performance of the $(\gamma + 1)$ REA is a constrained optimization problem. Specifically, we study the maximization of a linear *profit* function $p : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^n w_i x_i$ subject to the *uniform constraint* $\sum_{i=1}^n x_i \leq B$. The perturbation concerns the size of the uniform constraint: in the perturbed problem, the size bound B is replaced by $B - \delta$ or $B + \delta$.

As mentioned in the introduction, this problem has been previously studied in [31], and constitutes one of the few constrained optimization problems for which the running time of EAs has been formally analyzed. Shi et al. analyzed the *expected reoptimization time* of the $(1+1)$ EA and of three multi-objective EAs. In our terminology, they thus assume that x^{old} was an optimal solution for the problem (before the size bound B had been changed), and bound the expected time needed by the EAs to identify an optimal solution x^* for the perturbed problem. A main conclusion of the work by Shi et al. is that it can be beneficial to regard the constrained problem as a two-objective problem with the size (i.e., the number of ones) of the solution as one objective, and the profit values $p(x)$ as second objective.

For the $(1 + 1)$ EA, Shi et al. transform the constrained problem

$$\max p(x) = \sum_{i=1}^n w_i x_i \quad \text{s.t.} \sum_{i=1}^n x_i \leq B \tag{1}$$

into a pseudo-Boolean objective function

$$f : \{0,1\}^n \to \mathbb{R}, x \mapsto p(x) - C \max\Big\{\sum_{i=1}^n x_i - B, 0\Big\}, \tag{2}$$

where $C := n|w_{\max}| + 1$ for $w_{\max} := \max\{|w_i| \mid 1 \leq i \leq n\}$. With this choice, the penalty term guides the search towards the feasible region, which, once hit by the (1+1) EA, is not left by this algorithm, thanks to its elitist selection. It is proven in [31] that the $(1 + 1)$ EA has an $O(n^2 \log(Bw_{\max}))$ expected reoptimization time.

We analyze the expected optimization time of the $(\gamma + 1)$ REA on this problem formulation. The following theorem shows that it is $O(n\gamma\delta)$, provided that the perturbation estimate γ satisfies $\gamma \geq \delta - 1$.

Theorem 4.1. *Let $f^{\text{old}} : \{0,1\}^n \to \mathbb{R}, x \mapsto p(x) - C \max\{\sum_{i=1}^n x_i - B_{\text{old}}, 0\}$ be a function as in (2), with linear profit function p. Let x^{old} be an optimal solution for f^{old} satisfying $\sum_{i=1}^n x_i^{\text{old}} \leq B_{\text{old}}$ (i.e., x^{old} is feasible solution for the corresponding constrained problem (1)). Let δ be a positive integer satisfying $\delta \leq \min\{B_{\text{old}}, n - B_{\text{old}}\}$, let $B \in \{B_{\text{old}} - \delta, B_{\text{old}} + \delta\}$, and let $f : \{0,1\}^n \to \mathbb{R}, x \mapsto p(x) - C \max\{\sum_{i=1}^n x_i - B, 0\}$ be the perturbed fitness function that we obtain from f^{old} by replacing the penalty term $C \max\{\sum_{i=1}^n x_i - B_{\text{old}}, 0\}$ by $C \max\{\sum_{i=1}^n x_i - B, 0\}$.*

For all $i \in [\gamma + 1]$ the expected number of fitness evaluations needed by the $(\gamma + 1)$ REA to identify a solution of function value at least $\max\{f(y) \mid H(y, x^{\text{old}}) \leq i\}$ is $O(n\gamma i)$.

Proof. Let $i \leq \gamma + 1$. Let T^i denote the random variable that counts the number of function evaluations needed by the $(\gamma+1)$ REA to identify an optimal solution $x^{i,*}$ at Hamming distance i from x^{old}; i.e., a solution $x^{i,*} \in \arg\max\{f(y) \mid H(y, x^{\text{old}}) = i\}$.

For an inductive proof, we first bound $\mathrm{E}[T^1]$. Note that at any point in time the size of the population is at most $\gamma + 2$. The solution x^{old} (which is never removed from the population) has thus a probability of at least $1/(2(\gamma + 1))$ of being selected as parent individual. Conditioning on x^{old} being selected as parent, the probability of flipping a 0-bit of maximal weight ($B > B_{\text{old}}$) or a 1-bit of minimal weight ($B < B_{\text{old}}$), respectively, and no other bit is at

least $(1/n)(1-\frac{1}{n})^{n-1} \geq 1/(en)$. The expected waiting time for creating a point $x^1 \in \arg\max\{f(y) \mid H(y, x^{\text{old}}) = 1\}$ is thus at most $2en(\gamma+1) = O(n\gamma)$.

For fixed $j \in [i-1]$ assume that x^j has already been updated to a point of maximal possible fitness, i.e, $f(x^j) = \max\{f(y) \mid y \in \{0,1\}^n \text{ with } H(y, x^{\text{old}}) = j\}$. By the same reasoning as above, the probability to select x^j as parent is at least $1/(2(\gamma+1))$, and the probability to flip a 0-bit (1-bit) of maximal (minimal) weight is at least $1/(en)$, showing that $\mathrm{E}[T^{j+1}] \leq \mathrm{E}[T^j] + 2en(\gamma+1) = O(n\gamma(j+1))$ by the induction hypothesis. □

We note, without going into great detail, that the expected reoptimization time can strongly depend on the structure of the weights of the linear profit function. For an illustrative example, let us assume that the profit function is the BinaryValue function, i.e., the linear function with $w_i = 2^{n-i}$. Assume that $B_{\text{old}} = cn$ for some $c < 1$. Assume further that x^{old} is an optimal solution for f^{old}, i.e., x^{old} is the string with entry 1 in positions $i \leq B_{\text{old}}$ and entry 0 in positions $i > B_{\text{old}}$. Finally, assume that the new size bound is $B = B_{\text{old}} + 1$, i.e., we have $\delta = 1$. Then, regardless of γ, the expected reoptimization time is at least linear in n, since the B-th bit needs to be flipped in order to obtain the unique optimal solution for the perturbed function, which is the string having the first B entries equal to one, and all others equal to zero. If, on the other hand, the linear profit function is the OneMax function (i.e., the linear functions with $w_1 = w_2 = \ldots = w_n = 1$), the expected reoptimization time for the same perturbation of the size bound is constant. More precisely, it suffices to select x^{old} as parent and to flip in it exactly one of the $(1-c)n$ zero-bits. Unless a new optimal solution has already been found, the probability that a solution with fitness f^{old} is selected as parent equals $1/2$, regardless of γ. The probability to create an optimal solution for the new problem instance is thus at least $(1/2)(1-c)n/(en) = \Theta(1)$. This example can easily be extended to many other situations. A more detailed discussion of these effects can be found in [31].

5 MINIMUM SPANNING TREES

The classical minimum spanning tree (MST) problem can be formulated as follows. Given an edge-weighted undirected graph $G = (V, E, w)$, with $n = |V|$ nodes and $m = |E|$ edges, the goal is to find a subset $E' \subseteq E$ of minimal cost such that the graph $G(V, E')$ is connected. We denote by w_i the weight of edge e_i, $1 \leq i \leq m$, and assume that weights are strictly positive. We consider the search space $\{0,1\}^m$ where $x \in \{0,1\}^m$ gives a selection of edges, i.e., edge e_i is selected if and only if $x_i = 1$.

We consider the fitness function $f(x) = (c(x), w(x))$ where $c(x)$ denotes the number of connected components of the graph given by x and $w(x) = \sum_{i=1}^m w_i x_i$ is the weight of the chosen edges. The fitness function should be minimized with respect to lexicographic order which is equivalent to assigning to each additional components a large penaly. This standard formulation of the MST problem has already been investigated in [16] and a multi-objective formulation trading off $c(x)$ and $w(x)$ against each other has been considered in [15].

5.1 Additional Edges

We first consider the case where δ edges $e \notin E$ are added to the graph $G = (V, E)$. Let $x^* = (x_1^*, \ldots, x_m^*)$ be a solution representing a minimum spanning tree before a change has occurred.

We assume that the number (but not necessarily the endpoints, nor the weights) of the additional edges is known. The size of the search space is thus increased by δ. The new edges are labeled $e_{m+1}, \ldots e_{m+\delta}$. The $(\gamma+1)$ REA uses as input for the reoptimization the search point $x^{\text{old}} = (x_1^*, \ldots x_m^*, x_{m+1}, \ldots x_{m+\delta})$ with $x_{m+1} = \ldots = x_{m+\delta} = 1$.

Since the number of edges in an MST of a graph with n vertices is $n-1$, we know in this setting—unlike the cases considered in the previous sections—that the actual perturbation is δ. We nevertheless assume that a general bound $\gamma \geq \delta$ is used in the $(\gamma+1)$ REA, since x^{old} may have been communicated without the size bound δ.

We start our performance analysis of $(\gamma+1)$ REA with some structural observations about the minimum spanning tree in the extended graph.

Lemma 5.1. *Let T be a minimum spanning tree for a given graph $G = (V, E, w)$ and let G_δ be obtained from G by adding a set E_δ consisting of δ edges that satisfy $E \cap E_\delta = \emptyset$. Then there exists a minimum spanning tree T^* of G_δ that can be obtained by removing δ edges from $T \cup E_\delta$.*

Proof. Let T^* be a minimum spanning tree of $T \cup E_\delta$. Obviously T^* can be obtained from $T \cup E_\delta$ by deleting δ edges. We show that T^* is also a minimum spanning tree of G_δ. As T is a minimum spanning tree of G, for any cut that is obtained by removing an edge e of T, there is no edge in $E \setminus T$ that has a smaller weight than e and connects the two components (as T is a minimum spanning tree of G). Hence, only edges of E_δ can result in a spanning tree of smaller weight than T and a minimum spanning tree of $T \cup E_\delta$ is also a minimum spanning tree of G_δ. □

According to Lemma 5.1, we can obtain a minimum spanning tree of G_δ by deleting δ edges of $T \cup E_\delta$ in decreasing order of their weights that do not disconnect the graph.

Lemma 5.2. *Let T be a minimum spanning tree for a given graph $G = (V, E, w)$ and let G_δ be obtained from G by adding a set E_δ consisting of δ edges not previously present in E. An optimal solution x^i, $1 \leq i \leq \delta$, representing a connected graph of Hamming distance exactly i to x^{old} is obtained from $T \cup E_\delta$ by sequentially removing exactly i edges of the largest weight such that the graph does not get disconnected.*

Furthermore, an optimal solution x^{i+1} at Hamming distance $i+1$ from x^{old} can be obtained from x^i by removing the largest edge whose removal does not make the graph disconnected.

Proof. Consider the graph G_δ. According to Lemma 5.1 we can obtain a minimum spanning tree T^* for G_δ by removing those δ edges from $T \cup E_\delta$ that are of largest weight and whose removal does not make the graph disconnected. It suffices to remove these edges sequentially, in decreasing order of weight.

Let $E^* = \{e_1, \ldots, e_\delta\} \subseteq T \cup E_\delta$, with $w(e_1) \geq \ldots \geq w(e_\delta)$ be such a set of δ edges whose removal yields a minimum spanning tree. For $1 \leq i \leq \delta$ set $E_i := \{e_1, \ldots, e_i\}$; i.e., E_i is a subset of E^* consisting of a set of edges having the i largest weights (ties

are broken arbitrarily). Let x^i be the string obtained from x^{old} by flipping those bits that correspond to the edges E_i. The Hamming distance $H(x^{\text{old}}, x^i)$ of x^i to x^{old} is i, and the weight of x^i equals $w(x^i) = w(x^{\text{old}}) - w(E_i)$.

We show that x^i is an optimal solution at Hamming distance i to x^{old}, in the sense that $c(x^i) = 1$ and $w(x^i) = \min\{w(y) \mid H(x^{\text{old}}, y) = i, c(y) = 1\}$. Let y^i be a solution representing a connected graph at Hamming distance i to x^{old} having the smallest weight among all such solutions. Since $c(x^i) = 1 = c(y^i)$, we only need to show that $w(x^i) = w(y^i)$. If y^i is obtained by removing exactly i edges from x^{old} then we have $w(x^i) = w(y^i)$ due to the construction of x^i. If y^i is not obtained by removing exactly i edges, then an additional edge $e \notin E_\delta$ has to be inserted. In this case, by the restriction that $H(y^i, x^{\text{old}}) = i$, there can be at most $i - 1$ edges that have been removed from x^{old}, and these edges have to be such that the resulting graph is not disconnected. This implies $w(y^i) \geq w(x^{\text{old}}) - w(E_{i-1}) > w(x^i)$, contradicting the choice of y^i. We therefore obtain that y^i is obtained from x^{old} by removing i edges and therefore we have $w(x^i) = w(y^i)$.

Hence, the optimal solution x^i is a solution obtained from x^{old} by removing a set E_i of largest weight that does not make the graph disconnected. Having reached an optimal solution x^i which is a subset of the edges chosen by x^{old}, an optimal solution x^{i+1} is obtained by flipping the 1-bit corresponding to the largest edge whose removal does not make the graph disconnected. □

Theorem 5.3. *Let $\gamma \geq \delta$. Then the expected time until $(\gamma + 1)$ REA has computed a minimum spanning tree after the addition of δ edges to the graph $G = (V, E, w)$ when starting with x^{old} is $O(\gamma\delta n)$.*

Proof. Let G_δ be the graph obtained form G by the addition of the δ edges.

As in the previous two sections we perform an inductive proof and show that for each $1 \leq i \leq \delta$ the expected number of iterations needed by the $(\gamma + 1)$ REA to obtain an optimal solution x^i at Hamming distance i from x^{old} is $O(\gamma in)$. Let $x^0 = x^{\text{old}}$ and let $i \in [0..\delta - 1]$ be such that the $(\gamma + 1)$ REA has found an optimal solution at Hamming distance i from x^{old}. By Lemma 5.2 an optimal solution x^{i+1} at Hamming distance $i + 1$ from x^{old} is obtained from x^i by flipping in it exactly one of those 1-bits that correspond to an edge of largest weight and which is such that its removal does not make the graph disconnected, and flipping no other bit in x^i. The solution x^i is selected with probability at least $1/(2(\gamma + 1))$ and the corresponding mutation happens with probability at least $1/(en)$, so that the expected time needed to create from x^i an optimal point x^{i+1} at Hamming distance $i + 1$ is $O(\gamma n)$. Since the value of i has to be increased at most δ times, a minimum spanning tree T^* of G_δ is obtained from x^{old} in expected time $O(\gamma\delta n)$. □

5.2 Removal of Edges

We now consider the case where a set of δ edges $E_\delta \subset E$ is removed from the graph $G = (V, E, w)$ such that a still connected graph $G_{\text{new}} = (V, E \setminus E_\delta, w_{\text{new}})$ is obtained (where w_{new} denotes the restriction of w to the edges in $E \setminus E_\delta$). Let $x^* = (x_1^*, \ldots, x_m^*)$ be a solution representing a minimum spanning tree of G. We remove the bits corresponding to the removed edges in order to obtain the solution x^{old} that we are using for the initialization of the reoptimization process. Without loss of generality and to ease the presentation, we assume that the last δ bits are removed, which implies $x^{\text{old}} = (x_1^*, \ldots, x_{m-\delta}^*)$.

Theorem 5.4. *Let $\gamma \geq \delta$. Then the expected time until the $(\gamma + 1)$ REA has computed a minimum spanning tree after the removal of δ edges from $G = (V, E, w)$ when starting with x^{old} is $O(\gamma\delta n)$.*

Proof. Let $\delta' \leq \delta$ be the number of edges that have been removed from the minimum spanning tree represented by x^* for $G = (V, E, w)$. The solution x^{old} is a minimum spanning forest of $G_{\text{new}} = (V, E \setminus E_\delta, w_{\text{new}})$ consisting of $\delta' + 1$ connected components. For our analysis, we always pick the solution x^i such that for x^i and all x^j, $0 \leq j \leq i < \delta'$, a minimum spanning forest with $\delta' - j + 1$ connected components for G_{new} has already been obtained. Solution x^i is chosen as a parent for mutation with probability at least $1/(2(\gamma + 1))$. It is well known (and used, for example, in Prim's algorithm) that flipping the bit corresponding to an edge of minimal weight that does not create a cycle produces a solution x^{i+1} that is a minimum spanning forest with $\delta' - i$ connected components at Hamming distance $i + 1$ to x^{old}. There are at most δ' steps in which the value of i has to be increased such that a minimum spanning tree for G_δ which has Hamming distance $\delta' \leq \delta$ to x^{old} is obtained. This implies that a minimum spanning tree for G_δ is obtained after an expected number of $O(\gamma\delta' n) = O(\gamma\delta n)$ steps. □

6 CONCLUSIONS

The task of re-optimizing a previously encountered problem plays a crucial role in real-world applications. We contribute to the theoretical understanding and design of evolutionary algorithms for such dynamically changing problems and introduced a diversity-based approach which searches for good solutions around a good solution prior to the perturbation. This allows the algorithm to remember good components of the given problem. Our theoretical results show that this leads to highly effective evolutionary algorithms as it prevents recomputation of previously obtained knowledge about the given problem.

ACKNOWLEDGMENTS

This work has been supported by the Australian Research Council through grants DP160102401 and DP190103894, by COST action CA15140 ('ImAppNIO'), and by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences. Parts of this research has been conducted during a research visit of Frank Neumann as invited professor at Sorbonne University, with financial support from the LIP6 laboratory.

REFERENCES

[1] Anne Auger and Benjamin Doerr. 2011. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. Vol. 1. World Scientific.

[2] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. 2010. Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem. In *Proc. of Parallel Problem Solving from Nature (PPSN'10) (Lecture Notes in Computer Science)*, Vol. 6238. Springer, 1–10.

[3] Raymond Chiong, Thomas Weise, and Zbigniew Michalewicz (Eds.). 2012. *Variants of Evolutionary Algorithms for Real-World Applications*. Springer. https://doi.org/10.1007/978-3-642-23424-8

[4] Raphaël Dang-Nhu, Thibault Dardinier, Benjamin Doerr, Gautier Izacard, and Dorian Nogneng. 2018. A New Analysis Method for Evolutionary Optimization of Dynamic and Noisy Objective Functions. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM, 1467–1474.

[5] Kalyanmoy Deb. 2012. *Optimization for Engineering Design - Algorithms and Examples, Second Edition*. PHI Learning Private Limited. http://phindia.com/bookdetails/optimization_for_engineering_design_-algorithms_and_examples_by-deb_kalyanmoy_-isbn-978-81-203-4678-9

[6] Benjamin Doerr. 2018. Better Runtime Guarantees Via Stochastic Domination. In *Proc. of Evolutionary Computation in Combinatorial Optimization (EvoCOP'18)*. Springer, 1–17. Full version available at https://arxiv.org/abs/1801.04487.

[7] Benjamin Doerr. 2018. Probabilistic Tools for the Analysis of Randomized Optimization Heuristics. *CoRR* abs/1801.06733 (2018). arXiv:1801.06733 http://arxiv.org/abs/1801.06733

[8] Benjamin Doerr, Michael Gnewuch, Nils Hebbinghaus, and Frank Neumann. 2007. A Rigorous View on Neutrality. In *Proc. of IEEE Congress on Evolutionary Computation (CEC'07)*. IEEE, 2591–2597.

[9] Benjamin Doerr, Edda Happ, and Christian Klein. 2011. Tight Analysis of the (1+1)-EA for the Single Source Shortest Path Problem. *Evolutionary Computation* 19 (2011), 673–691.

[10] Benjamin Doerr, Edda Happ, and Christian Klein. 2012. Crossover Can Provably be Useful in Evolutionary Computation. *Theoretical Computer Science* 425 (2012), 17–33.

[11] Benjamin Doerr, Dirk Sudholt, and Carsten Witt. 2013. When Do Evolutionary Algorithms Optimize Separable Functions in Parallel?. In *Proc. of Foundations of Genetic Algorithms (FOGA'13)*. ACM, 48–59.

[12] Thomas Jansen. 2013. *Analyzing Evolutionary Algorithms - The Computer Science Perspective*. Springer.

[13] Jörg Lässig and Dirk Sudholt. 2013. Design and Analysis of Migration in Parallel Evolutionary Algorithms. *Soft Computing* 17 (2013), 1121–1144.

[14] Andrei Lissovoi and Carsten Witt. 2015. Runtime Analysis of Ant Colony Optimization on Dynamic Shortest Path Problems. *Theoretical Computer Science* 561 (2015), 73–85.

[15] Frank Neumann and Ingo Wegener. 2006. Minimum Spanning Trees Made Easier via Multi-Objective Optimization. *Natural Computing* 5 (2006), 305–319.

[16] Frank Neumann and Ingo Wegener. 2007. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science* 378 (2007), 32–40. https://doi.org/10.1016/j.tcs.2006.11.002

[17] Frank Neumann and Carsten Witt. 2010. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity* (1st ed.). Springer.

[18] Frank Neumann and Carsten Witt. 2015. On the Runtime of Randomized Local Search and Simple Evolutionary Algorithms for Dynamic Makespan Scheduling. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 3742–3748.

[19] T.T. Nguyen and X. Yao. 2012. Continuous Dynamic Constrained Optimization: The Challenges. *IEEE Transactions on Evolutionary Computation* 16, 6 (2012), 769–786. https://doi.org/10.1109/TEVC.2011.2180533

[20] Mojgan Pourhassan, Wanru Gao, and Frank Neumann. 2015. Maintaining 2-Approximations for the Dynamic Vertex Cover Problem Using Evolutionary Algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*. ACM, 903–910. https://doi.org/10.1145/2739480.2754700

[21] Günther R. Raidl, Gabriele Koller, and Bryant A. Julstrom. 2006. Biased Mutation Operators for Subgraph-Selection Problems. *IEEE Trans. Evolutionary Computation* 10 (2006), 145–156. https://doi.org/10.1109/TEVC.2006.871251

[22] Pratyusha Rakshit, Amit Konar, and Swagatam Das. 2017. Noisy evolutionary optimization algorithms - A comprehensive survey. *Swarm and Evolutionary Computation* 33 (2017), 18–45. https://doi.org/10.1016/j.swevo.2016.09.002

[23] Joachim Reichel and Martin Skutella. 2009. On the size of weights in randomized search heuristics. In *Proc. of Foundations of Genetic Algorithms (FOGA'09)*. ACM, 21–28. https://doi.org/10.1145/1527125.1527130

[24] Hendrik Richter and Shengxiang Yang. 2013. Dynamic Optimization Using Analytic and Evolutionary Approaches: A Comparative Review. In *Handbook of Optimization - From Classical to Modern Approach*, Ivan Zelinka, Václav Snásel, and Ajith Abraham (Eds.). Intelligent Systems Reference Library, Vol. 38. Springer, 1–28. https://doi.org/10.1007/978-3-642-30504-7_1

[25] Vahid Roostapour, Aneta Neumann, and Frank Neumann. 2018. On the Performance of Baseline Evolutionary Algorithms on the Dynamic Knapsack Problem. In *Proc. of Parallel Problem Solving from Nature (PPSN'18) (Lecture Notes in Computer Science)*, Vol. 11101. Springer, 158–169. https://doi.org/10.1007/978-3-319-99253-2_13

[26] Vahid Roostapour, Aneta Neumann, Frank Neumann, and Tobias Friedrich. 2018. Pareto Optimization for Subset Selection with Dynamic Cost Constraints. *CoRR* abs/1811.07806 (2018). arXiv:1811.07806 http://arxiv.org/abs/1811.07806 Conference version appears at AAAI 2019.

[27] Vahid Roostapour, Mojgan Pourhassan, and Frank Neumann. 2018. Analysis of Evolutionary Algorithms in Dynamic and Stochastic Environments. *CoRR* abs/1806.08547 (2018). arXiv:1806.08547 http://arxiv.org/abs/1806.08547

[28] Jonathan E. Rowe and Dirk Sudholt. 2014. The Choice of the Offspring Population Size in the (1, λ) Evolutionary Algorithm. *Theoretical Computer Science* 545 (2014), 20–38.

[29] Baruch Schieber, Hadas Shachnai, Gal Tamir, and Tami Tamir. 2018. A Theory and Algorithms for Combinatorial Reoptimization. *Algorithmica* 80, 2 (2018), 576–607. https://doi.org/10.1007/s00453-017-0274-8

[30] Feng Shi, Frank Neumann, and Jianxin Wang. 2018. Runtime Analysis of Randomized Search Heuristics for the Dynamic Weighted Vertex Cover Problem. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM, 1515–1522. https://doi.org/10.1145/3205455.3205580

[31] Feng Shi, Martin Schirneck, Tobias Friedrich, Timo Kötzing, and Frank Neumann. 2017. Reoptimization Times of Evolutionary Algorithms on Linear Functions Under Dynamic Uniform Constraints. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'17)*. ACM, 1407–1414.

[32] Dirk Sudholt. 2018. On the Robustness of Evolutionary Algorithms to Noise: Refined Results and an Example where Noise Helps. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM, 1523–1530.

[33] Ingo Wegener. 2001. Theoretical Aspects of Evolutionary Algorithms. In *Proc. of Automata, Languages and Programming (ICALP'01) (Lecture Notes in Computer Science)*, Vol. 2076. Springer, 64–78.

[34] Carsten Witt. 2014. Revised analysis of the (1+1) EA for the minimum spanning tree problem. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'14)*. ACM, 509–516. https://doi.org/10.1145/2576768.2598237

[35] Anna Zych-Pawlewicz. 2018. Reoptimization of NP-Hard Problems. In *Adventures Between Lower Bounds and Higher Altitudes - Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday (Lecture Notes in Computer Science)*, Vol. 11011. Springer, 477–494. https://doi.org/10.1007/978-3-319-98355-4_28