Revised Analysis of the (1+1) EA for the Minimum Spanning Tree Problem

Carsten Witt
DTU Compute
Technical University of Denmark
2800 Kgs. Lyngby
Denmark

ABSTRACT

We revisit the classical analysis of the (1+1) EA for the minimum spanning tree problem in the case that nothing is known about the weights of the underlying graph. Here the original upper bound on the expected running time by Neumann and Wegener [Theor. Comput. Sci. 378(1), 32–40, 2007], which depends on the largest weight of the graph, is of no use. The best upper bound available before in this case is due to Reichel and Skutella [FOGA 2009, 21–28] and is of order $O(m^3 \log n)$, where m is the number of edges and n the number of vertices. Using an adaptive drift analysis, we show the improved bound $O(m^2(\sqrt{c(G)} + \log n))$, where $c(G)$ is the circumference (length of the longest cycle) of the graph. This is only by an asymptotic factor of at most $\sqrt{n}/\log n$ away from the classical lower bound. Furthermore, an alternative fitness function leading to the bound $O(m^2 \log n)$ is proposed, and limitations of the adaptive drift analysis are pointed out.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms

Keywords

Evolutionary Algorithms, Minimum Spanning Tree Problem, Runtime Analysis

1. INTRODUCTION

The running time analysis of randomized search heuristics, in particular evolutionary algorithms (EAs), on problems on combinatorial optimization has advanced considerably in the last decade [11]. This line of research gives theoretically founded insight into what choice of algorithms and parameters allows for efficient solution of a problem and what choice is inefficient. One conclusion is that even very simple EAs and local search algorithms find optimal or approximate solutions to several combinatorial optimization in expected polynomial time. This applies to both "easy" problems from the class P of polynomial-time solvable problems as well as (in terms of approximate solutions) to several NP-hard problems.

Recently developed analytical tools such as multiplicative and adaptive drift analysis [4] enable us to prove very exact running time statements for EAs on pseudo-boolean example functions such as OneMax and LeadingOnes [3] and even the whole class of linear functions [15]. Here even the exact constant in the leading term of the polynomial describing the running time is known. In contrast, results in combinatorial optimization are often of asymptotic nature, i. e., use O-notation, and are not necessarily tight, i. e., only upper and lower bounds on the running time are known, which may differ by large factors. In particular, there is a collection of analyses (e. g., [2, 10, 13]) leading to polynomial lower bounds, whereas the upper bounds are pseudo-polynomial in that they depend on a certain problem parameter such as the largest weight of a graph. While it is believed that the dependence on the largest weight is only an artefact of the analyses and there has been some progress on selected problems (outlined below), it is an outstanding problem to prove tight bounds. With the recently developed analytical toolbox, it is worth revisiting the combinatorial problems and studying whether we can obtain improved bounds.

In this paper, we focus on how a simple (1+1) EA and randomized local search (RLS) solve the well-known minimum spanning tree (MST) problem. This is one of the earliest (originally published at GECCO 2004) and very influential running time results of EAs in combinatorial optimization. Given an undirected, weighted and connected graph $G = (V, E, w)$, Neumann and Wegener [10] study a canonical fitness function returning the total edge weight of a search point, provided it encodes a tree. Otherwise the fitness functions contains penalty terms leading to trees. With respect to the (1+1) EA, they obtain an upper bound on the expected running time of $O(m^2(\log n + \log w_{\max}))$, where $n := |V|$, $m = |E|$, and $w_{\max}$ is the largest edge weight of the graph, and a lower bound of $\Omega(m^2 \log n)$. Hence, if nothing is known about the weights, upper and lower bounds can be arbitrarily far apart. This is in sharp contrast to the analysis of RLS, where Reichel and Skutella [13] could show the tight result $\Theta(m^2 \log n)$. Briefly speaking, a major diffi-

GECCO'14, July 12–16, Vancouver, BC, Canada.

ACM 978-1-4503-2662-9/14/07…$15.00.
http://dx.doi.org/10.1145/2576768.2598237.

culty is introduced to the analysis due to the ability of the (1+1) EA to exchange many edges in a step.

There is a single prior work by Reichel and Skutella [12] which aims at removing the dependence on $w_{\max}$ from the running time bound for the (1+1) EA on the MST problem. The idea is to replace the fitness function by another one having smaller weights but leading to the same acceptance behavior of the (1+1) EA. In this way, it is seen that $w_{\max}$ need not exceed $m^{m/2}$, hence $\log(w_{\max}) \leq m \ln m$, which implies the general upper bound $O(m^3 \log n)$ on the expected running time. Note that there still is a gap of order m between upper and lower bound.

Our contribution narrows this gap to at most $\sqrt{n}/\log n$. More precisely, we show the upper bound $O(m^2(\log n + \sqrt{c(G)}))$, where $c(G)$ is the circumference (length of the longest cycle) of the underlying graph. Hence, if the graph has circumference up to $O(\log^2 n)$, the bound is asymptotically tight. The result is obtained by adaptive drift analysis along with a carefully chosen potential function. Moreover, the analysis reveals structural insights into the behavior of (1+1) EA on the problem, which allow us to define an alternative fitness function whose set of global optima still coincides with the MSTs of the graph. Somewhat surprisingly, the expected running time can be bounded by $O(m^2 \log n)$ in this case, which corresponds to the bound for RLS and the lower bound w. r. t. the classical fitness function.

This paper is structured as follows. In Section 2, we give formal definitions of the algorithms and problems studied. In Section 3, we describe the structural insights that lead to the alternative fitness function and that are fundamental for the forthcoming drift analysis. This analysis implies the new result w. r. t. the original fitness function and is presented in Section 4. Limitations of the technique, in particular why the factor $\sqrt{c(G)}$ could not be avoided, are discussed in Section 5. We finish with some conclusions.

2. PRELIMINARIES

We consider two classical randomized search heuristics called RLS and (1+1) EA, see Algorithms 1 and 2, which are intensively studied in the theory of randomized search heuristics [1, 8]. The search space is $\{0,1\}^m$ since subsets of edges will be encoded by their characteristic vector. The (1+1) EA is a globally searching hillclimber, whereas RLS samples from a neighborhood of size 2. Steps changing two bits are crucial for the MST problem since all MSTs have the same number of edges. The running time (synonymously, optimization time) of the algorithms is defined as the random number of iterations until an optimal search point has been sampled.

Algorithm 1 (1+1) EA

$t := 0$.
Choose uniformly at random $x_0 \in \{0,1\}^m$.
repeat
  Create x' by flipping each bit in x_t independently with probability $1/m$.
  $x_{t+1} := x'$ if $f(x') \leq f(x_t)$, and $x_{t+1} := x_t$ otherwise.
  $t := t + 1$.
until some stopping criterion is fulfilled.

Recall that we, throughout the paper, assume an arbitrary but fixed connected, undirected and weighted graph

Algorithm 2 Randomized local search (RLS)

$t := 0$.
Choose uniformly at random $x_0 \in \{0,1\}^m$.
repeat
  Choose $b \in \{1,2\}$ uniformly. Flip b bits in x_t chosen uniformly at random.
  $x_{t+1} := x'$ if $f(x') \leq f(x_t)$, and $x_{t+1} := x_t$ otherwise..
  $t := t + 1$.
until some stopping criterion is fulfilled.

$G = (V, E, w)$, where $w\colon E \to \mathbb{N}$, for which an MST has to be found. We denote the edges by $e_1, \ldots, e_m$ and their associated weights by $w_1, \ldots, w_m$. Neumann and Wegener [10] propose the following fitness function. Let $x \in \{0,1\}^m$ be a search point describing a selection of edges, i. e., a subgraph of G, by identifying one-bits with selected edges. We define $c(x)$ to be the number of connected components (CCs) in this subgraph. If the current search point does not encode a spanning tree, the first aim of the function is to reduce the number of CCs to 1 and the second one is to arrive at a spanning subtree, i. e., a subgraph choosing exactly $n-1$ edges. These two aims are encoded in the fitness function by appropriate penalty terms. Afterwards, the most difficult phase of optimization begins. From the set of trees, one of smallest total weight has to be found by mutations swapping edges.

Let $M \geq mw_{\max}$, where $w_{\max} := \max\{w_1, \ldots, w_m\}$, be a number that is chosen so large that unconnected graphs always have worse fitness than connected ones and cyclic subgraphs worse fitness than trees. The fitness function is defined by

$$f(x) = M^2(c(x)-1) + M\left(\sum_{i=1}^{m} x_i - (n-1)\right) + \left(\sum_{i=1}^{m} w_i x_i\right).$$

Hence, if x encodes a tree, its weight is returned. Once a search point of the (1+1) EA or RLS describes a tree, the algorithm will never accept non-trees. It is not too difficult to show (similar to the OneMax problem) that the algorithms quickly arrive at trees.

Theorem 1 ([10]). *The expected time until RLS or (1+1) EA working on the fitness function f have constructed a (not necessarily minimal) spanning tree is $O(m \log n)$.*

Hence, in light of the overall bound $O(m^2(\log n + \sqrt{c(G)}))$ we want to prove, the time until a tree has been found is asymptotically negligible. In the remainder of the paper, we assume that (1+1) EA/RLS has already constructed a tree and analyze the progress towards MSTs. Assuming x to encode a tree, we denote by $w(x) := \sum_{i=1}^{m} w_i x_i$ its weight and define w_{opt} as the weight of an MST.

Finally, we list one important drift theorem used to capture this progress. It is taken from [4] except for the trivial adaptation to arbitrary positive $s_{\min}$-values, which can be found in [5].

Theorem 2 (Multiplicative Drift). *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum $s_{\min} > 0$. Let $\{X^{(t)}\}_{t \geq 0}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random first point in time $t \geq 0$ for which $X^{(t)} = 0$.*

Suppose that there exists a $\delta > 0$ such that

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

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

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

Moreover, it holds that $\Pr(T > (\ln(s_0/s_{\min}) + r)/\delta)) \leq e^{-r}$ for any $r > 0$.

3. LOCAL SEARCH, STRUCTURAL PROPERTIES OF SPANNING TREES, AND ALTERNATIVE FITNESS FUNCTIONS

Once the (1+1) EA has created a search point describing a spanning tree, it is necessary for an improvement to flip some one-bits, corresponding to the removal of edges, and the same number of zero-bits, corresponding to the inclusion of edges; otherwise the resulting graph would not contain $n-1$ edges. RLS is dependent on steps flipping exactly two bits, and these yield a relative progress towards the optimum, as the following lemma describes.

LEMMA 3 ([10]). *Let x be a search point describing a non-minimum spanning tree. Then there exist some $k \in \{1, \dots, n-1\}$ and k accepted 2-bit flips such that the average weight decrease of these flips is at least $(w(x) - w_{\text{opt}})/k$.*

The multiplicative drift theorem (Theorem 2) can now be used to prove that the final phase of optimization, i. e., until RLS and (1+1) EA have converted an arbitrary tree to an MST, takes time $O(m^2(\log n + \log w_{\max}))$. The key arguments are that a two-bit flip has probability $\Omega(1/m^2)$, which accounts for the factor $O(m^2)$, and that $w(x) - w_{\text{opt}} \leq m w_{\max}$. Taking the logarithm, the last bound accounts for the factor $\log n + \log w_{\max}$. (Note that $\log m = O(\log n)$ since $m \leq n^2$.)

The proof of Lemma 3 goes back to the following structural result.

LEMMA 4 ([9]). *Let T be a minimum spanning tree and S be an arbitrary (possibly also minimal) spanning tree. Then there exists a bijection $\Phi\colon T \setminus S \to S \setminus T$ such that for every edge $e \in T \setminus S$, $\Phi(e)$ closes a cycle in S and $w(e) \leq w(\Phi(e))$.*

The previous two lemmas were also used in the prior work by Reichel and Skutella [12] that aims at removing the dependence on $w_{\max}$ from the running time bound for the (1+1) EA. As already mentioned, they replace the objective function by another one with bounded weights but leading to the same acceptance behavior of the algorithm. Since the acceptance behavior of RLS and (1+1) EA only depends on the ordering of fitness values of the search points considered for selection (they are so-called ranking-based algorithms), the following definition, inspired by [12], comes in handy. It also takes into account the maximum Hamming distance $H(x, x')$ of two search points compared in a selection step.

DEFINITION 5. *Let $d \in \{1, \dots, n\}$. Two pseudo-boolean functions $f, g \in \{0,1\}^n \to \mathbb{R}$ are called* neighborhood-d rank-equivalent*, for short* neighborhood-d equivalent*, if for all x, x' such that $H(x, x') \leq d$, it holds* $\operatorname{sgn}(f(x) - f(x')) = \operatorname{sgn}(g(x) - g(x'))$.

The formulation that for all x, x' we have $\operatorname{sgn}(f(x) - f(x')) = \operatorname{sgn}(g(x) - g(x'))$ is equivalent to that for all x, x' we have $f(x) \leq f(x')$ if and only if $g(x) \leq g(x')$. We prefer the first formulation since it immediately points out that equal function values are also preserved.

The observation made in the following lemma implies for RLS that $w_{\max}$ may be replaced by m on any MST problem. This result is proved in [13]. Roughly speaking, it is sufficient to replace every weight w_i of the linear weight function $w(x)$ by the rank of the i-th weight in the increasing sequence (with equals weights getting equal ranks). For the sake of completeness, we supply the simple arguments here.

LEMMA 6. *Let $f(x) = f_m x_m + \cdots + f_1 x_1$ be a linear function and assume w. l. o. g. that $f_m \geq \cdots \geq f_1 > 0$. Then f and the function $r(x) = r_m x_m + \cdots + r_1 x_1$, where $r_i := |\{f_1, \dots, f_i\}|$, are neighborhood-2 equivalent.*

PROOF. Let $x \in \{0,1\}^n$ be arbitrary but fixed. Let x' be another search point satisfying $H(x, x') \leq 1$. Consider the function r from this lemma. If $H(x, x') = 1$, then $f(x') > f(x)$ if and only if $r(x') > r(x)$ since all weights are positive in both functions. Analogously the case $f(x') < f(x)$ and $r(x') < r(x)$ is treated. The case $f(x') = f(x)$ is not possible.

Suppose now $H(x, x') = 2$ and let $i > j$ be the two indices where the search points differ. If i and j are both 0-bits or 1-bits in x, we argue as before. Otherwise, $f(x') > f(x)$ if i is a 0-bit, j a 1-bit in x and $f_i > f_j$. By construction of the r_i, $r(x') > r(x)$ follows. Analogously, $f(x') < f(x)$ implies $r(x') < r(x)$. Finally, the case $f(x') = f(x)$ is equivalent to $f_i = f_j$. Then also $r_i = r_j$ and $r(x') = r(x)$ follows. □

The linear function $r(x)$ from Lemma 6 can be understood as a "surrogate function" on which RLS can be run instead of $w(x)$ when searching for MSTs, without changing its stochastic behavior. As a result, the $O(\log w_{\max} + \log n)$ factor from the running time bound is replaced by the logarithm of the largest r-value, i. e., $\log(r_1 + \cdots + r_m) \leq \log((m^2 + m)/2)$, which is $O(\log m) = O(\log n)$. This completes the argument from [13] and proves that the running time of RLS, which only considers neighborhoods of size at most 2, is $O(m^2 \log n)$. Note that this matches the lower bound from [10].

The structure of the function $r(x)$ from Lemma 6 allows us further insights on the optimization process during the search for MSTs, which in the end will lead to the alternative fitness function for the (1+1) EA. We start with another definition.

DEFINITION 7. *Two linear functions $f(x) = f_m x_m + \cdots + f_1 x_1$ and $g(x) = g_m x_m + \cdots + g_1 x_1$ are called* weight-rank equivalent *if for all $i, j \in \{1, \dots, n\}$ it holds* $\operatorname{sgn}(f_i - f_j) = \operatorname{sgn}(g_i - g_j)$.

The function $r(x)$ from Lemma 6 is the unique weight-rank equivalent function to $w(x)$ which minimizes the maximal weight in natural numbers. Interestingly, the function $r(x)$ makes sense as a "surrogate" function also for the (1+1) EA on the MST problem even though it is not necessarily neighborhood-d equivalent to the actual fitness function for $d > 2$. In fact, every function that is weight-rank equivalent to the classical fitness function for the MST (assuming trees as search points) preserves the location of global optima, as the following lemma shows.

LEMMA 8. *Suppose $g(x) = g_m x_m + \cdots + g_1 x_1$ is weight-rank equivalent to the function $w(x) = w_m x_m + \cdots + w_1 x_1$ obtained from an MST instance. Let x_{opt} describe a minimum spanning tree with respect to the weight function w. Then for every $x \in \{0,1\}^m$ describing a spanning tree it holds $g(x) \geq g(x_{\text{opt}}) =: g_{\text{opt}}$. Moreover, $g(x) = g_{\text{opt}}$ if and only if x describes an MST w. r. t. w.*

If x does not describe an MST, there exists some $k \in \{1, \ldots, n-1\}$ and k accepted 2-bit flips such that the average weight decrease of these flips is at least $(g(x) - g_{\text{opt}})/k$.

PROOF. The analysis from Lemma 6 shows that two functions being weight rank-equivalent are also neighborhood-2 rank-equivalent. Hence, improving 2-bit flips with respect to w are improving 2-bit flips with respect to g and vice versa. Lemma 4 is in force with respect to the g-weights and thereby also Lemma 3. In particular, non-optimal trees (w. r. t. w) must have a larger g-value than minimum spanning trees (w. r. t. w).

All minimum spanning trees w. r. t. w have the same w-value and can be converted into each other by two-bit flips due to Lemma 4. Hence, they must all have the same g-value, too. □

Hence, if the search for minimum spanning trees takes place on the graph with weight function g instead of f, and additionally $g(x) \geq g(x_{\text{opt}}) + 1$ is guaranteed for non-MSTs x (to maintain $s_{\min} = 1$ in Theorem 2), we can redo the analysis by Neumann and Wegener [10] to obtain the following result.

THEOREM 9. *Given an undirected weighted connected graph $G = (V, E, w)$, let the fitness function $g\colon \{0,1\}^m \to \mathbb{R}$ be defined by*

$$g(x) = M^2(c(x) - 1) + M\left(\sum_{i=1}^{m} x_i - (n-1)\right) + \sum_{i=1}^{m} x_i \cdot \mathit{rank}(w_i),$$

where $M := m^2$ and $\mathit{rank}(w_i)$ denotes the rank of weight w_i in the increasingly sorted sequence. The expected time until the (1+1) EA finds an optimum for $g(x)$ is bounded by $O(m^2 \log n)$, and every such optimum describes an MST in G.

The "surrogate" function g from the preceding theorem can be used instead of the classical function f and guarantees polynomial running time. However, it is based on non-trivial problem knowledge about the MST problem. In fact, the required structural results from Lemma 4 are shown using the ideas underlying the correctness proof of Kruskal's algorithm, which also considers the ranks of the edge weights only. See also the characterization of the set of MSTs by weight ranks in Section 6 of [14]. So with this level of problem knowledge, no evolutionary algorithm would have to be applied.

Otherwise, without knowing Kruskal's algorithm or related structural insights, one would probably have come up with the original fitness function f as defined in [10]. Therefore, it is still interesting to analyze whether the time to find an optimum for f (in case of large weights) is less than the $O(m^3 \log n)$ bound that was obtained by Reichel and Skutella [12]. This is dealt with in the next section.

4. IMPROVED UPPER BOUND FOR THE (1+1) EA

In this section, we show our improved upper bound on the running time of the (1+1) EA for the MST problem, using the classical fitness function f. In the following theorem, the circumference (length of the longest cycle) of an undirected graph is denoted by $c(G)$. If G is acyclic (in which case the MST problem is trivial anyway), we define $c(G) := 0$.

THEOREM 10. *Given an arbitrary MST instance, the running time of the (1+1) EA on the corresponding fitness function f is upper bounded by $O(m^2(\log n + \sqrt{c(G)}))$ both in expectation and with probability $1 - O(n^{-c})$ for any constant $c > 0$.*

To prove the theorem, we conduct an adaptive drift analysis, where the underlying potential function $h(x)$, to be minimized, depends on both the weight function w and the circumference $c(G)$. The exact definition of the potential function is inspired by the techniques developed in [15] and further applied in [6] and [7]. More precisely, we apply drift analysis using a potential function that will be defined below in accordance with the definition of weight-rank equivalence. However, this is not a straightforward application of the above-mentioned literature but includes non-trivial modifications in the potential function and the drift analysis due to the characteristics of the underlying MST problem.

Once having defined the potential function, the idea is to analyze the potential $X^{(t)} := h(x^{(t)})$ of the random search point $x^{(t)}$ maintained by the (1+1) EA on f at time t. We first bound its expected one-step change $E(X^{(t)} - X^{(t+1)} \mid X^{(t)})$, i. e., the expected decrease of distance from time t to time $t+1$. The following lemma states this bound as well as a bound on the maximum value of the potential function, which is required in the multiplicative drift theorem (Theorem 2).

LEMMA 11.

1. $E(X^{(t)} - X^{(t+1)} \mid X^{(t)}) \geq \frac{X^{(t)}}{2e^2 m^2}$.
2. $\ln(X^{(t)}/s_{\min}) \leq O(\log n) + \sqrt{2ec(G)}$.

Deferring the definition of h and the proof of the previous lemma, we obtain our theorem.

PROOF OF THEOREM 10. We will argue later that $h(x) = 0$ if x encodes an MST and $h(x) \geq s_{\min}$ for some $s_{\min} > 0$ otherwise. The expected running time now follows immediately by plugging the two statements of Lemma 11 into Theorem 2. The tail bound is obtained by setting $r = c \ln n$. □

In the following, we unroll the proofs of the drift statements. The proof of Lemma 11 relies on the analysis of the one-step drift of the potential function $h\colon \{0,1\}^m \to \mathbb{R}$. We now introduce the setup required to define $h(x)$. Without loss of generality, we assume $w_m \geq \cdots \geq w_1 > 0$ and write search points as $x_m \ldots x_1$, calling x_m the leftmost and x_1 the rightmost bit. To reduce the number of cases needed in the forthcoming drift analysis, we assume that for all $i, j \in [m] := \{1, \ldots, m\}$

$$i > j \quad \Rightarrow \quad w_i = w_j \vee w_i - w_j \geq W^*$$

for some W^* defined below. This holds without loss of generality since we can simply multiply all (integral) weights

by W^* without changing the acceptance behavior of the (1+1) EA.

DEFINITION 12. *Let an undirected weighted graph G be given and let x^* denote an arbitrary but fixed MST. For $i \in [m]$, let*

$$\gamma_i := (1+\Phi)^{i-1},$$

where

$$\Phi := \frac{\sqrt{2ec(G)}}{m}$$

Based on this, let

$$g_i := \min\{\gamma_i, g_{i-1} + w_i - w_{i-1}\}$$

for all $1 < i \leq m$, $g_1 := \gamma_1 := 1$ and $g(x) := \sum_{i=1}^{m} g_i x_i$. Finally, let the potential function $h\colon \{0,1\}^m \to \mathbb{R}$ be defined by

$$h(x) := g(x) - g(x^*).$$

For any $i \in [m]$, we also define:

- $\kappa(i) := \max\{j \leq i \mid g_j = \gamma_j\}$, *the most significant index right of i (possibly i itself) capping according to the sequence γ_i,*
- $\eta(i) := \min\{j \geq i \mid g_j = \gamma_j\}$, *the least significant index left of i (possibly i itself) capping according to the sequence γ_i,*
- $L(i) := \{m, \ldots, \kappa(i)\}$, *the indices left of $\kappa(i)$,*
- $R(i) := \{\kappa(i) - 1, \ldots, 1\}$, *the indices right of $\kappa(i)$.*

The idea of the potential function, reflected by the minimum operator in the definition of g_i, is to cap the original weights at γ_i in case they are "too steeply increasing" and to rebuild the slope of original weights otherwise by letting $g_i - g_{i-1} = w_i - w_{i-1}$. The variable $\kappa(i)$ is intuitively a "breaking point" and denotes the most significant index right of i where g_j equals γ_j. The intuition is that the potential function will underestimate the progress made at indices being at least as significant as i. In all less significant indices (those in $R(i)$), we will pessimistically assume that they contribute a loss, and the choice of $\kappa(i)$ guarantees that this loss is overestimated.

Clearly, $w(x)$ and $g(x)$ are weight-rank equivalent. Hence, by Lemma 8, the set of search points x where $h(x) = 0$ equals the set of MSTs with respect to w. We now list another useful property related to the $\kappa(i)$, which means that the g_i-weights are constant between two different κ-indices if W^* is chosen large enough (more details below). This will simplify the case analysis in the main proof.

LEMMA 13. *If $W^* \geq \gamma_m$ then for any $i \in [m]\colon g_i = \gamma_{\kappa(i)}$*

PROOF. Recall that for any $i \geq 2$ there only are the two cases $w_i - w_{i-1} = 0$ and $w_i - w_{i-1} \geq W^* \geq \gamma_m$. If $w_i - w_{i-1} \geq \gamma_m$ then $g_{i-1} + w_i - w_{i-1} \geq 1 + \gamma_m$, hence the minimum in the definition of g_i is obtained at γ_i, implying $i = \kappa(i)$. In the other case, since $\gamma_i > \gamma_{i-1} \geq g_{i-1}$, the minimum is taken at g_{i-1} and the claim follows (inductively decreasing i until it equals $\kappa(i)$). □

From now on, assume W^* to be sufficiently large for the previous lemma to hold. We obtain the following observation.

OBSERVATION 14. *Sort the set of all different κ-values decreasingly as $\kappa_\ell > \cdots > \kappa_1$. Then for $j \in \{2, \ldots, \ell\}$ we have $g_{\kappa_j} > g_{\kappa_j - 1} = g_{\kappa_j - 2} = \cdots = g_{\kappa_{j-1}}$, i. e., there is a plateau of constant g-weights between two subsequent κ-indices and an increase in g-weight at the next κ-index.*

The following lemma allows us to set $s_{\min} = \Phi$ in the multiplicative drift theorem.

LEMMA 15. *For any $x \in \{0,1\}^m$, $h(x) = 0$ or $h(x) \geq \Phi$.*

PROOF. Since $g(x)$ and $w(x)$ are weight-rank equivalent, Lemma 8 is in force. The non-negativity now follows from the lemmma since $h(x) = g(x) - g(x^*)$ for an arbitrary MST x^* w. r. t. w. Moreover, if x is not optimal, $g(x)$ and thus $h(x)$ can be written as the sum of k positive differences of edges weights (resulting from the edge swaps). Each such difference is at least $\gamma_2 - \gamma_1 = \Phi$. □

As mentioned above, we will analyze the stochastic process $X^{(t)}_{t \geq 0}$ where $X^{(t)} = h(x^{(t)})$ for all t, and define $\Delta_t := X^{(t)} - X^{(t+1)}$. Recall that we are interested in the first point in time t where $X^{(t)} = 0$ holds. The one-step drift $E(\Delta_t \mid X^{(t)})$ of the potential function will be worked out conditioned on certain events depending on two flipping bits. The following notions prepare the definition of these events.

DEFINITION 16. *Given $x^{(t)} \in \{0,1\}^n$, denote by x' the random search point created by mutation of $x^{(t)}$ (before selection). We define*

- $I := \{i \in [m] \mid x^{(t)}_i = 1\}$ *the one-bits in $x^{(t)}$,*
- $I^* := \{i \in I \mid x'_i = 0\}$ *the one-bits flipping to 0,*
- $Z := \{i \in [m] \mid x^{(t)}_i = 0\}$ *the zero-bits in $x^{(t)}$,*
- $Z^* := \{i \in Z \mid x'_i = 1\}$ *the zero-bits flipping to 1.*

Note that the random sets I^* and Z^* are disjoint and that the remaining bits in $[m]$ contribute nothing to the Δ_t-value.

Obviously, for $\Delta_t \neq 0$ it is necessary that $x^{(t+1)} \neq x^{(t)}$. We fix an arbitrary search point $x^{(t)}$ and let A be the event that $x^{(t+1)} \neq x^{(t)}$. The event A requires that

$$I^* \neq \emptyset \text{ and } \sum_{j \in I^*} w_j - \sum_{j \in Z^*} w_j \geq 0.$$

Hence, for A to occur it is necessary that for at least one $i \in I^*$

$$\sum_{j \in I^*} w_j - \sum_{j \in Z^* \cap L(i)} w_j \geq 0,$$

since we only ignore the loss due to the bits in $R(i)$. We decompose the event A according to two indices i and j, where i denotes the leftmost flipping one-bit and and j the leftmost flipping zero-bit right of $\eta(i)$.

DEFINITION 17. *The event $A_{i,j}$, where $i, j \in [m]$, occurs iff the following five conditions hold simultaneously.*

1. *$I^* \neq \emptyset$ and $Z^* \neq \emptyset$.*
2. *$i = \max I^*$.*
3. *$j = \max(Z^* \cap \{1, \ldots, \eta(i) - 1\})$.*
4. *$\sum_{k \in I^*} w_k - \sum_{k \in Z^* \cap L(i)} w_k \geq 0$.*
5. *A spanning tree is obtained by flipping the bits from $I^* \cup Z^*$ in $x^{(t)}$.*

Obviously, all events $A_{i,j}$ are disjoint. Note that each accepted mutation must flip as many zero-bits as one-bits to maintain spanning trees. In particular, each accepted mutation flips at least one pair of bits (i,j), where i is a one-bit, j a zero-bit and j weighs at most as much as i. If bit j weights strictly less, then j is right of i. Otherwise, j must be of same weight of i and hence in $\eta(i)-1,\dots,\kappa(i)$. Hence, the disjoint union of the events $A_{i,j}$ is a superset of A (in other words, is necessary for A). The key inequality used to bound the one-step drift is stated in the following lemma.

LEMMA 18. *$E(\Delta_t \mid A_{i,j}) \geq \frac{g_i - g_j}{2e}$ for all $i,j \in [m]$ such that $\Pr(A_{i,j}) > 0$ and all $t \geq 0$.*

Before we prove Lemma 18, let us show how it can be used to prove Lemma 11.

PROOF OF LEMMA 11. We still fix an arbitrary search point $x^{(t)}$, denote by $X^{(t)} = h(x^{(t)})$ its potential and investigate the following step. As observed above, in the step the potential remains either unchanged or a certain event $A_{i,j}$ occurs. The total drift can then be expressed as

$$\begin{aligned} &E(X^{(t)} - X^{(t+1)} \mid X^{(t)}) \\ &= \sum_{i\in I, j\in Z\cap\{1,\dots,\eta(i)-1\}, \Pr(A_{i,j}>0)} E(\Delta_t \mid A_{i,j}) \cdot \Pr(A_{i,j}). \end{aligned}$$

Using Lemma 18, the last expression is at least

$$\sum_{i\in I, j\in Z\cap\{1,\dots,\eta(i)-1\}, \Pr(A_{i,j})>0} \frac{g_i - g_j}{2e} \Pr(A_{i,j}).$$

Note that $\Pr(A_{i,j}) \geq (1/m^2)(1-1/m)^{m-2} \geq 1/(em^2)$ if $A_{i,j}$ is possible since it is sufficient to flip bits i and j and not to flip the rest to accept the mutation. Note also that $g_i - g_j \geq 0$ since $j \leq \eta(i) - 1$, so that we may omit terms from the sum to obtain a lower bound.

For every accepted two-bit flip there is an event $A_{i,j}$ corresponding to the two flipping bits. By Lemma 8, there are k accepted two-bit flips $(i_1,j_i),\dots,(i_k,j_k)$, where $j_r \leq \eta(i_r) - 1$ for $r \in [k]$. Consequently,

$$\begin{aligned} E(X^{(t)} - X^{(t+1)} \mid X^{(t)}) &\geq \sum_{r=1}^{k} (g_{i_r} - g_{j_r}) \Pr(A_{i_r,j_r}) \\ &\geq \sum_{r=1}^{k} \frac{(g_{i_r} - g_{j_r})}{2e} \frac{1}{em^2} \\ &= \frac{g(X^{(t)}) - g(x^*)}{2e^2m^2}, \end{aligned}$$

where the equality holds since the i_r denote the one-bits in $X^{(t)} \setminus x^*$ and the j_r the one-bits in $x^* \setminus X^{(t)}$. Hence, we get

$$E(X^{(t)} - X^{(t+1)} \mid X^{(t)}) \geq \frac{h(X^{(t)})}{2e^2m^2},$$

which proves the first statement of Lemma 11.

For the second statement of Lemma 11, we use that for every t it holds $X^{(t)} \leq \sum_{i=1}^{m} \gamma_i$. Using the geometric series and the inequality $1 + x \leq e^x$, the latter is at most

$$\frac{\left(1 + \frac{\sqrt{2ec(G)}}{m}\right)^m}{\frac{\sqrt{2ec(G)}}{m}} \leq \frac{me^{m\sqrt{2ec(G)}/m}}{\sqrt{2ec(G)}} \leq me^{\sqrt{2ec(G)}}.$$

From this we obtain

$$\ln(X^{(0)}) \leq \sqrt{2ec(G)} + \ln m.$$

Finally, $\ln(s_{\min}) = \ln(\Phi) \geq \ln(1/m)$, which altogether proves $\ln(X^{(0)}/\ln(s_{\min})) \leq \sqrt{2ec(G)} + O(\log n)$. □

The still outstanding proof of Lemma 18 requires a careful analysis of the one-step drift, taking into account the specific structure of the drift function.

PROOF OF LEMMA 18. Recall that we want to condition on the event $A_{i,j}$ (Definition 17), where i is the leftmost flipping one-bit and j a flipping zero-bit right of $\eta(i)$. Moreover, recall the notions introduced in Definitions 12 and 16. Let

$$\begin{aligned} \Delta_L(i) &:= \left(\sum_{k\in I^*} g_k - \sum_{\ell\in Z^*\cap L(i)} g_\ell\right) \cdot \mathbb{1}_A, \\ \Delta_R(i) &:= \left(\sum_{k\in Z^*\cap R(i)} g_k\right) \cdot \mathbb{1}_A, \end{aligned}$$

where $\mathbb{1}_A$ denotes the indicator random variable of event A. Recall that $\Delta_t = 0$ if A does not occur. Otherwise, $\Delta_t = \sum_{k\in I^*} g_k - \sum_{\ell\in Z^*} g_\ell$. Hence, we have $\Delta_t = (\Delta_L(i) - \Delta_R(i))$ regardless of i. By linearity of expectation, we obtain

$$E(\Delta_t \mid A_{i,j}) = E(\Delta_L(i) \mid A_{i,j}) - E(\Delta_R(i) \mid A_{i,j}). \quad (1)$$

We first show that $(\Delta_L(i) \mid A_{i,j})$ is a nonnegative random variable, i. e., the probability of any negative outcome is 0. To prove this, assume that $A_{i,j}$ holds, which implies that no bit left of i flips to 0. We inspect the two sums from the definition of $\Delta_L(i)$. Since only spanning trees are accepted, any accepted mutation changes as many zero- as one-bits. Hence, it must hold that $|I^*| \geq |Z^* \cap L(i)|$. Pessimistically (with the aim of proving a lower bound on $\Delta_L(i)$), we assume that equality holds for the size of the two sets. Then there are r pairs $(k_1,\ell_1),\dots,(k_r,\ell_r)$, where $k_s \in I^*$ and $\ell_s \in Z^* \cap L(i)$, such that

$$(\Delta_L(i) \mid A_{i,j}) = \left(\sum_{s=1}^{r} g_{k_s} - g_{\ell_s}\right) \cdot \mathbb{1}_A.$$

Note that for any s we have $\ell_s \geq \kappa(i)$ and $k_s \leq i$ since i is the leftmost flipping one-bit. By definition,

$$g_k - g_\ell \geq w_k - w_\ell \text{ for } k < \ell, \quad (2)$$
$$g_k - g_\ell = w_k - w_\ell = 0 \text{ for } i \geq k > \ell \geq \kappa(i). \quad (3)$$

Hence,

$$\begin{aligned} (\Delta_L(i) \mid A_{i,j}) &= \left(\sum_{s=1}^{r} g_{k_s} - g_{\ell_s}\right) \cdot \mathbb{1}_A \\ &\geq \left(\sum_{s=1}^{r} w_{k_s} - w_{\ell_s}\right) \cdot \mathbb{1}_A \\ &= \left(\sum_{k\in I^*} w_k - \sum_{\ell\in Z^*\cap L(i)} w_\ell\right) \cdot \mathbb{1}_A \geq 0 \end{aligned}$$

where the first inequality uses (2) and (3) and the last inequality holds by definition of $A_{i,j}$.

Now let $S_{i,j}$ be the event that $|Z^* \cap L(i) \setminus \{j\}| = 0$, that is, that no bit in $L(i)$ (possibly except j) flips to 1. We have

$$E(\Delta_L(i) \mid A_{i,j}) = E(\Delta_L(i) \mid A_{i,j} \cap S_{i,j}) \cdot \Pr(S_{i,j} \mid A_{i,j}) + E(\Delta_L(i) \mid A_{i,j} \cap \bar{S}_i) \cdot \Pr(\bar{S}_i \mid A_{i,j})$$

by the law of total probability. Since the random variable $(\Delta_L(i) \mid A_{i,j})$ cannot have any negative outcomes, all these conditional expectations are non-negative as well. From (1) we thus derive

$$E(\Delta_t \mid A_{i,j}) \geq E(\Delta_L(i) \mid A_{i,j} \cap S_{i,j}) \cdot \Pr(S_{i,j} \mid A_{i,j}) - E(\Delta_R(i) \mid A_{i,j}). \tag{4}$$

We will now bound the terms from (4) from below to obtain our result. For $(S_{i,j} \mid A_{i,j})$ to occur, it is sufficient that all bits in $L(i)$ except i and j do not flip (note that bits i and j flip by assumption). Consequently, $\Pr(S_{i,j} \mid A_{i,j}) \geq (1-1/m)^{m-2} \geq e^{-1}$. According to the definition of $A_{i,j}$, the mutation flipping only i and j is accepted. Hence, we estimate $E(\Delta_L(i) \mid A_{i,j} \cap S_{i,j}) \geq g_i - g_j$. Altogether,

$$E(\Delta_L(i) \mid A_{i,j} \cap S_{i,j}) \cdot \Pr(S_{i,j} \mid A_{i,j}) \geq \frac{g_i - g_j}{e}. \tag{5}$$

Finally, we need a bound on $E(\Delta_R(i))$, which is determined by the bits in $R(i)$ that flip to 1. Here we distinguish two cases with respect to j.

Case 1: $j < \kappa(i)$. By definition of $A_{i,j}$, not all bits in $Z \cap R(i)$ but only those of index at most at j can flip to 1, and bit j is forced to flip. Since only spanning trees are accepted, each flipping zero-bit requires another one-bit to flip to 0. If only bit $k \in Z$ was flipped to 1, the resulting search point would contain a cycle, say edges $e_1, \dots, e_\ell$, one of which is denoted by bit k. For a spanning tree to be obtained, it is necessary that one of the bits associated with edges $e_1, \dots, e_\ell$ flips to 0 simultaneously with k flipping to 1.

Clearly, $\ell \leq c(G)$. Event $A_{i,j}$ already occurs if only bits i and j flip. Moreover, bits are flipped independently. Hence, on $A_{i,j}$, the probability that $k \in Z \cap R(i) \cap \{1, \dots, j-1\}$ flips is bounded from above by $\frac{1}{m} \cdot \frac{c(G)}{m} = \frac{c(G)}{m^2}$. Pessimistically, we assume that A occurs in such a mutation. By linearity of expectation, it follows that

$$E(\Delta_R(i) \mid A_{i,j}) \leq \sum_{k=1}^{j-1} \frac{c(G)}{m^2} g_k.$$

Along with (4) and (5), we get

$$\begin{aligned} E(\Delta_t \mid A_{i,j}) &\geq \frac{g_i - g_j}{e} - \frac{c(G)}{m^2} \sum_{k=1}^{j-1} g_k \\ &\geq \frac{g_i - g_j}{e} - \frac{c(G)}{m^2} \left(\sum_{k=1}^{\kappa(j)-1} \gamma_k + (j - \kappa(j)) \gamma_{\kappa(j)} \right), \end{aligned} \tag{6}$$

where we used Lemma 13 to bound $g_k \leq \gamma_k$ for $k \leq \kappa(j)-1$. Since $j < \kappa(i)$, Observation 14 yields that

$$g_i - g_j = \gamma_{\kappa(i)} - \gamma_{\kappa(j)} > 0,$$

hence

$$\begin{aligned} g_i - g_j &= (1+\Phi)^{\kappa(i)} - (1+\Phi)^{\kappa(j)} \\ &\geq (1+\Phi)^{\kappa(j)} ((1+\Phi)^{\kappa(i)-\kappa(j)} - 1) \\ &\geq \gamma_{\kappa(j)} (\kappa(i) - \kappa(j)) \Phi, \end{aligned} \tag{7}$$

where we used Bernoulli's inequality.

We are left with the sum over k. Plugging in the definition of γ_k in the geometric series, this is estimated by

$$\begin{aligned} \sum_{k=1}^{\kappa(j)-1} \gamma_k &= \sum_{k=1}^{\kappa(j)-1} (1+\Phi)^{k-1} \\ &= \frac{(1+\Phi)^{\kappa(j)-1} - 1}{\Phi} \leq \frac{\gamma_{\kappa(j)}}{\Phi}, \end{aligned}$$

so that the term in parentheses from (6) is bounded according to

$$\begin{aligned} &\sum_{k=1}^{\kappa(j)-1} \gamma_k + (j - \kappa(j)) \gamma_{\kappa(j)} \\ &\quad \leq \frac{j + 1 - \kappa(j)}{\Phi} \gamma_{\kappa(j)} \leq \frac{\kappa(i) - \kappa(j)}{\Phi} \gamma_{\kappa(j)}, \end{aligned}$$

where we used $\Phi < 1$ and $j < \kappa(i)$. Since by definition $c(G)/m^2 = \Phi^2/(2e)$, we bound the whole negative term in (6) as

$$\frac{c(G)}{m^2} \frac{\kappa(i) - \kappa(j)}{\Phi} \gamma_{\kappa(j)} \leq \frac{\Phi \gamma_{\kappa(j)} (\kappa(i) - \kappa(j))}{2e} \leq \frac{g_i - g_j}{2e},$$

using (7).

Hence, finally,

$$E(\Delta_t \mid A_{i,j}) \geq \frac{g_i - g_j}{e} - \frac{g_i - g_j}{2e} = \frac{g_i - g_j}{2e}.$$

Case 2: $j \geq \kappa(i)$. Then by definition of $A_{i,j}$, we get $g_i - g_j = 0$, so we only need to prove $E(\Delta_t \mid A_{i,j}) \geq 0$. If i and j are the only flipping bits, nothing is to show. Otherwise, we remove i and j from I^* and Z^*, respectively, and recompute i and j according to event $A_{i,j}$. Possibly repeating this process, we either arrive at a pair (i,j) satisfying Case 1, which proves a positive drift, or run out of flipping bits, which means that the drift is 0. □

5. LIMITATIONS OF THE TECHNIQUE

The drift analysis done in the proof of Theorem 10 requires that a result in the style of Lemma 18 can be obtained. Roughly speaking, the lemma says that the multiplicative drift w. r. t. the g-function is in the same order as if only two-bit flips were allowed (more precisely, order $\Omega(1/m^2)$). This leads to a factor $O(\sqrt{c(G)})$ in the running time bound for the following reason. Each flipping zero-bit, say bit k, must be compensated by a flipping one-bit for the spanning tree property to be maintained. Now, if the flipping zero-bit closes a cycle of length ℓ, there are $\ell - 1$ ways of flipping a one-bit, all of which are accepted. Pessimistically, the proof assumes $\ell = \Omega(n)$. In addition, it pessimistically assumes $\Omega(m)$ different choices for the flipping zero-bit k. The math in the analysis of Case 1 in the proof of Lemma 18 only works under these pessimistic assumptions if subsequent g_i-values are by a factor of at least $1 + \Omega(\sqrt{c(G)}/m)$ apart.

The situation pessimistically assumed in the drift analysis cannot be excluded during the optimization. Consider a subgraph on $n'+1$ vertices, where $n' = \Omega(n)$, and assume that a path of length n' through the vertices $v_1, \dots, v_{n'}$, forms a spanning tree for the subgraph. Let $V_L := \{v_1, \dots, v_{n'/4}\}$ and $V_R := \{v_{3n'/4}, \dots, v_{n'}\}$ be the first resp. last $n'/4$ vertices on the path. Each of the $(n')^2/4 = \Omega(m)$ edges between V_L and V_R closes a cycle of length at least $n'/4 = \Omega(n)$,

which is in accordance with the pessimistic assumptions. Hence, the analysis of the one-step drift does not yield immediate room for improvement here.

Of course, looking at the scenario just described, a uniform flipping zero-bit would probably lead to shorter cycles in the next steps. So a drift analysis over several steps, keeping track of the stochastic change of the longest path in the current spanning tree, might help further improve the upper bound on the expected running time.

Finally, it seems not too difficult to obtain from the drift analysis the bound $O(m^2 c(G) \log(n))$ if the mutation probability of the (1+1) EA is reduced from $1/m$ to $1/(m\sqrt{c(G)})$. The lower bound by Neumann and Wegener [10] could also be adapted to this case, altogether resulting in a tight bound $\Theta(m^2 c(G) \log n)$. We consider this not to be particularly interesting since the decreased mutation probability simply would result in many steps not flipping bits at all.

Conclusions

We have revisited the classical analysis of the (1+1) EA for the minimum spanning tree problem and the case of unbounded edge weights. By an adaptive drift analysis, we have obtained the bound $O(m^2(\log n + \sqrt{c(G)}))$, which improves the previous result from [12] by an asymptotic factor of $m/\sqrt{c(G)}$. Furthermore, we have gained structural insights into the problem. These allowed us to define an alternative fitness function on which the running time for the (1+1) EA matches the one of RLS and asymptotically does not depend on the weights of the graph.

We have also pointed out reasons why the gap between upper and lower running time bound could not be closed. Future research on this problem seems necessary. Furthermore, there are other results in combinatorial optimization such as the single-source shortest path problem [2], where the best upper bound on the running time of the (1+1) EA depends on the largest weight in the underlying problem and the analysis from [12] does not apply. Additional insight into the problem structure seems required to come up with any improved bound here.

6. REFERENCES

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

[2] Surender Baswana, Somenath Biswas, Benjamin Doerr, Tobias Friedrich, Piyush P. Kurur, and Frank Neumann. Computing single source shortest paths using single-objective fitness functions. In *Proc. of FOGA '09*, pages 59–66. ACM Press, 2009.

[3] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Proc. of PPSN 2010*, volume 6238 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2010.

[4] Benjamin Doerr and Leslie Ann Goldberg. Adaptive drift analysis. *Algorithmica*, 65(1):224–250, 2013.

[5] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[6] Benjamin Doerr and Sebastian Pohl. Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet. In *Proc. of GECCO '12*, pages 1317–1324. ACM Press, 2012.

[7] Benjamin Doerr, Dirk Sudholt, and Carsten Witt. When do evolutionary algorithms optimize separable functions in parallel? In *Proc. of FOGA '13*, pages 51–64. ACM Press, 2013.

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

[9] Ernst W. Mayr and C. Greg Plaxton. On the spanning trees of weighted graphs. *Combinatorica*, 12(4):433–447, 1992.

[10] Frank Neumann and Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40, 2007.

[11] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Natural Computing Series. Springer, 2010.

[12] Joachim Reichel and Martin Skutella. On the size of weights in randomized search heuristics. In *Proc. of FOGA '09*, pages 21–28. ACM Press, 2009.

[13] Joachim Reichel and Martin Skutella. Evolutionary algorithms and matroid optimization problems. *Algorithmica*, 57(1):187–206, 2010.

[14] Ingo Wegener. Simulated annealing beats metropolis in combinatorial optimization. In *Proc. of ICALP '05*, volume 3580 of *Lecture Notes in Computer Science*, pages 589–601. Springer, 2005.

[15] Carsten Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability and Computing*, 22(2):294–318, 2013.