Runtime Analysis of Evolutionary Algorithms with Biased Mutation for the Multi-Objective Minimum Spanning Tree Problem

Vahid Roostapour
Optimisation and Logistics
The University of Adelaide
Adelaide, Australia

Jakob Bossek
Optimisation and Logistics
The University of Adelaide
Adelaide, Australia

Frank Neumann
Optimisation and Logistics
The University of Adelaide
Adelaide, Australia

ABSTRACT

Evolutionary algorithms (EAs) are general-purpose problem solvers that usually perform an unbiased search. This is reasonable and desirable in a black-box scenario. For combinatorial optimization problems, often more knowledge about the structure of optimal solutions is given, which can be leveraged by means of biased search operators. We consider the Minimum Spanning Tree (MST) problem in a single- and multi-objective version, and introduce a biased mutation, which puts more emphasis on the selection of edges of low rank in terms of low domination number. We present example graphs where the biased mutation can significantly speed up the expected runtime until (Pareto-)optimal solutions are found. On the other hand, we demonstrate that bias can lead to exponential runtime if "heavy" edges are necessarily part of an optimal solution. However, on general graphs in the single-objective setting, we show that a combined mutation operator which decides for unbiased or biased edge selection in each step with equal probability exhibits a polynomial upper bound – as unbiased mutation – in the worst case and benefits from bias if the circumstances are favorable.

CCS CONCEPTS

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

KEYWORDS

Evolutionary algorithms, Minimum spanning tree problem, Runtime analysis, Biased mutation

1 INTRODUCTION

Evolutionary algorithms (EAs) are randomized general-purpose problem solvers that mimic principles from Darwinian evolution theory. These algorithms have proven successful in a wide range of applications, in particular, in tackling (multi-objective) combinatorial $\mathcal{NP}$-hard optimization problems [5, 7]. The theoretical understanding of EAs' working principles has made tremendous progress in the past decades with respect to expected runtime analysis, fixed-budget analysis and general convergence aspects [1, 28].

The problem considered here is a classical combinatorial optimization problem with countless applications in engineering, logistics and many other fields: the *Minimum Spanning Tree* (MST) problem. Given an undirected edge-weighted graph, the goal is to find a spanning sub-graph which is a tree and has minimal total weight among all such trees. When each edge is assigned multiple – usually conflicting – weights, one is interested in a set of multi-objective compromise solutions (moMST). The single-objective MST problem is well-understood and solvable in polynomial time by well-known algorithms, e. g., the algorithm by Kruskal [18]. In contrast, the moMST is proven to be $\mathcal{NP}$-hard [29] and all deterministic approaches may suffer from potential intractability problems. Here, many successful evolutionary multi-objective algorithms have been proposed (see, e. g., [2, 17, 31]).

In the area of runtime analysis of bio-inspired computation, spanning tree problems have obtained significant attention. The classical MST problem has been investigated for simple single-objective approaches of EAs [23] and ant colony optimization [24]. Furthermore, it has been shown that a multi-objective formulation of the problem can lead to significantly faster evolutionary algorithms [22]. For the moMST, it has been shown in [21] that a multi-objective evolutionary algorithm can compute a 2-approximation in pseudo-polynomial time. Furthermore, the results given in [23] have been revisited in the context of multiplicative drift analysis [11] and improved results for special graph classes have been presented in [26, 30].

Usually evolutionary algorithms perform an unbiased search due to their frequent application in settings where knowledge on the fitness function can only be gained by fitness function evaluations. However, if domain knowledge on the composition of (Pareto-)optimal solutions is available one should incorporate this knowledge into mutation operators to speed up the evolution considerably [9, 10, 13–15]. Neumann and Wegener [21] introduced an asymmetric mutation operator on bit strings where the probability for a 1-bit to flip depends on the number of 1-bits in the solution and likewise for 0-bits. Here, on average, the number of 1-bits in a

Vahid Roostapour, Jakob Bossek, and Frank Neumann

solution is not changed which is beneficial for the minimum spanning tree. In fact, the authors were able to obtain runtime speedups adopting this operator for the MST. Jansen and Sudholt [15] further investigated this operator. They give examples where asymmetry is beneficial and where it is not. Doerr et al. [9, 10] tackle the Eulerian cycle problem with asymmetric mutation and prove much slower runtime bounds in comparison to symmetric mutation. For the classical MST problem it is legitimate to assume that edges of low weight/rank are more likely to be in an MST than edges of high weight/rank. Such knowledge can also be leveraged in terms of biased mutation as demonstrated impressively by Raidl et al. [25] on random graphs. The authors showed that mutation, where the edge selection probability is biased towards lower rank edges, can lead to immense speedups for evolutionary algorithms for different sub-graph selection problems, inter alia the MST. Likewise, for the moMST problem, non-dominated spanning trees are more likely composed of edges which are dominated by few other edges, i. e., edges of low non-domination level or domination number. A recent study by Bossek et al. [3] confirms this assumption empirically. Both Raidl et al. and Bossek et al. consider the simple edge-exchange mutation on spanning trees: an edge is added to a spanning tree and an edge is dropped from the unique introduced cycle to obtain another spanning tree. Here, the authors introduce bias by modifying the edge selection probability favoring low-rank edges. Both studies serve as a starting point and motivation for our work.

In this paper we consider biased mutation for evolutionary algorithms for the single- and multi-objective MST problem and compare with unbiased counterparts. Specifically, we examine the effects of mutation bias on the time complexity of simple EAs until they hit an optimal solution or cover the Pareto-front for the first time. We show that bias can be both boon and bane depending on the structure of optimal solutions on example graphs. I. e., there are situations where introduced bias leads to improved upper bounds where we save a factor of n if the ranks of edges which are part of optimal solutions are $O(n)$. Contrarily, if heavy edges are frequent members of optimal solutions, bias towards lightweight edges may entail an exponential deterioration in the expected running time. Luckily, in the single-objective setting, we can combine the best of both worlds. A simple modification, which decides for unbiased or biased mutation in each step independently with probability $1/2$, leads to a guaranteed polynomial runtime bound of $O(n^3 \log(n \cdot w_{max}))$ for general graphs where w_{max} is the maximum edge weight in the graph. At the same time this strategy benefits from bias if the circumstances allow for it saving on a factor of n.

After having motivated our work we introduce the (mo)MST problem formally, establish a vocabulary and introduce the considered algorithms in Section 2. Sections 3 and 4 deal with our runtime analysis in the single-objective and multi-objective MST setting, respectively. Section 5 wraps up the work with some concluding remarks and outlook on future work.

2 PRELIMINARIES

Let $G = (V, E)$ denote a graph with vertex set V and edge set E. For convenience, we write $n = |V|$ and $m = |E|$. A spanning tree of graph G is a sub-graph $G' = (V, E')$ if and only if there exists exactly one path between any two vertices in G'. In the single-objective

---

Algorithm 1 (1+1) EA

1: Let T be a random spanning tree on $G = (V, E)$.
2: Set the edge-selection strategy.
3: **while** optimum not found **do**
4:     $T' \leftarrow T$
5:     $k \leftarrow 1 + \text{Pois}(1)$
6:     Based on the selection strategy, assign the probability $q(e)$ to each edge $e \in E$.
7:     **for** k times **do**
8:         Choose $e \in E$ with probability $q(e)$.
9:         $T' \leftarrow T' \cup \{e\}$
10:         Drop an edge from the resulting cycle in T' uniformly at random.
11:     **if** T' has no worse fitness than T **then**
12:         $T \leftarrow T'$

---

scenario, each edge $e \in E$ is assigned a positive weight $w(e)$ and the goal is to find a spanning tree with minimum total weight, called Minimum Spanning Tree (MST). In the multi-objective scenario, each edge is assigned two weights $w(e) = (w_1(e), w_2(e))$.[1] The goal is to find a spanning tree such that the total weight in both weight functions is minimized simultaneously. This may result in a set of incomparable trade-offs which are not necessarily better than each other in both weights. In order to capture this aspect mathematically we adopt the well-known notion of *Pareto dominance* [6] – a core concept in multi-objective optimization – to establish a partial order of spanning trees. Let $w_i(T) = \sum_{e \in T} w_i(e)$. We say spanning tree T_1 weakly (Pareto-)dominates spanning tree T_2, denoted by $T_1 \geq T_2$, if $w_1(T_1) \leq w_1(T_2) \wedge w_2(T_1) \leq w_2(T_2)$. The strong dominance holds when at least one of the inequalities is strict and it is denoted by $T_1 > T_2$. T_1 is called non-dominated if there is no other spanning tree that dominates T_1. Likewise, $w(T_1) = (w_1(T_1), w_2(T_1))$ is the non-dominated objective vector. The union set of all non-dominated spanning trees is called Pareto set, its image in objective space is called the Pareto front, and each solution is termed a Pareto(-optimal) solution or multi-objective MST (moMST). Our goal is to find a non-dominated spanning tree for each non-dominated objective vector. In the following, we present the algorithms that we use to tackle these problems.

2.1 Algorithms

We consider the performance of the (1+1) EA (see Algorithm 1) facing the single-objective MST problem. It is initialized with a random spanning tree T. There have been different studies on generating random spanning trees such as a rather classical randomized algorithm by Broder [4], with expected running time of $O(n \log n)$ for almost all graphs or more recently by Madry et al. [19]. Afterwards, the algorithm sets an *edge-selection strategy*, i. e., the edge-selection probability distribution that is used in Line 6. Next, the algorithm sets $k = \text{Pois}(1) + 1$, the number of edges for the mutation step, where $\text{Pois}(1)$ stems from a Poisson distribution with rate $\lambda = 1$. The constant ensures that we always perform at least one mutation and avoids counting iterations that does not

---

[1] Clearly, more than two objective functions are possible. Since we restrict our analysis to bi-objective problems in this paper we refrain from introducing the general form in favor of less notation overhead.
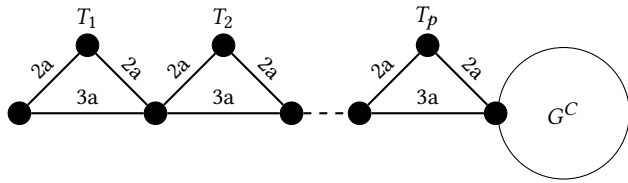
Figure 1: Triangular-tailed graph G with a chain of $p = n/4$ triangles and a giant component $G^C = K_{n/2}$. [23]

Algorithm 2 GSEMO

1: Initialize population P with a random spanning tree on $G = (V, E)$.
2: Set the edge-selection strategy.
3: **while** not all Pareto-optimal solutions found **do**
4:     Choose $T \in P$ uniformly at random.
5:     $T' \leftarrow T$
6:     $k \leftarrow 1 + \text{Pois}(1)$
7:     Based on the selection strategy, assign the probability $q(e)$ to each edge $e \in E$.
8:     **for** k times **do**
9:         Choose $e \in E$ with probability $q(e)$.
10:         $T' \leftarrow T' \cup \{e\}$
11:         Drop an edge from the resulting cycle in T' uniformly at random.
12:     **if** $\{T'' \in P \mid T'' > T'\} = \emptyset$ **then**
13:         $P = P \setminus \{T'' \in P \mid T' \geq T''\} \cup \{T'\}$

---

generate new solutions. The same approach has been used in [27]. In the mutation step, an edge is selected according to its probability $q(e)$ and is added to T. As the mutant is no longer acyclic after the edge insertion, removing a randomly chosen edge from the unique cycle is required to reestablish the tree property. This guarantees that the resulting graph is a spanning tree. The algorithm repeats this procedure k times to achieve a new solution T' and replaces T by T' if $w(T') \leq w(T)$.

We consider three versions of Algorithm 1 where the difference is in the edge-selection strategy.

(1+1) EA-UM refers to the unbiased variant of (1+1) EA in which always each edge is selected with uniform probability $q(e) = 1/m$. We also consider (1+1) EA with biased mutation called (1+1) EA-BM, in which the mutation probability of edge e has been set based on the approximation of the probability that e appears in the MST. The approximation, which is the result of experimental analyses, gives higher probability to the edges with lower weights to be selected. The details on how to calculate the approximation is given in the following sections. Note that for these versions of (1+1) EA, the edge-selection strategy does not change during the optimization process and has been set at the beginning of the algorithm. In other words, the edge-selection strategy deterministically assigns values of $q(e)$ (see line 6 in Algorithm 1), i. e., either uniform or biased mutation with probability 1. Additionally, we analyze a "hybrid" (1+1) EA, called (1+1) EA-MM (MM for <u>m</u>ixed <u>m</u>utation), where in each iteration of the outer loop the algorithm decides by fair coin-tossing which strategy (biased or unbiased) to use.

For the multi-objective scenario, our runtime analysis is based on the global simple evolutionary multi-objective algorithm (GSEMO; see Algorithm 2). GSEMO stores a set of non-dominated solutions in the population P, which is initialized with a single random spanning tree. In each iteration, it selects a solution T from P uniformly at random and sets the number of the edges to be added in the mutation step: one plus a random value sampled from a Poisson distribution with $\lambda = 1$. The mutation step is the same as the (1+1) EA and guarantees that the resulting graph T' is also a spanning tree. If there is no solution in P that strongly dominates T', T' is added to P and all the solutions that T' weakly dominates are removed from P. Similar to the single-objective setting, two versions are subject to analysis: GSEMO-UM with uniform edge-selection probability $q(e) = 1/m$ and its biased counterpart GSEMO-BM, in which edges that are dominated by fewer edges in E have higher probability to be selected for the mutation (see Section 4 for details).

3 SINGLE-OBJECTIVE PROBLEM

In this section, we consider two types of triangular-tailed graphs, G_1 and G_2, which are structurally the same but are different in the weights of the edges. A triangular-tailed graph consists of a clique, G^C, with $\nu = n/2$ vertices and a triangular tail, G^T, with $\eta = n/4$ triangles (Figure 1). In both G_1 and G_2, each triangle has 2 edges with weights $2a$ and one edge with weight $3a$, where $a := n^2$. The weights of edges in the clique are $4a$ and a in G_1 and G_2, respectively.

Neumann and Wegener proved that (1+1) EA with bit-string representation, which flips each bit with probability $1/m$ and is initialized with a random graph, finds the MST of the triangular-tailed graphs in $\Omega(n^4 \log n)$ expected time [23], i. e. the triangular-tailed graph has been used as the worst case example to prove the lower bound. This bound is proven for a fitness function that prevents the algorithm to accept solutions other than spanning trees after achieving the first spanning tree. Moreover, the most time consuming phase in their proof is finding the MST from an achieved spanning tree. Hence, their proof also holds even if (1+1) EA is initialized with a spanning tree.

Using the same worst case example, we prove that (1+1) EA-UM finds the MST in $\Theta(n^2 \log n)$. Afterwards, we improve this bound for graph G_1, in which the edges of G^T are lighter than the edges of the clique, by enhancing the biased mutation in (1+1) EA-BM. Inspired by the study of Raidl et al. [25], we use the ranking strategy to perform the biased mutation. To this aim, we assign rank r, $1 \leq r \leq |E|$, to each edge based on its placement in ascending order of the weights, ties are broken uniformly at random. For each edge $e \in E$ with rank r, we approximate the probability of e to appear in the MST with $p(r) = a^r$. Then, we set

$$q(e) = q_{\text{b}}(e) = \frac{\sqrt{p(r)}}{\sum_{i=1}^{m} \sqrt{p(r)}},$$

as the probability of selecting e for the mutation step, where $a = \frac{n-1}{n}$. We show that (1+1) EA-BM finds the MST of G_1 in expected time $\Theta(n \log n)$. However, it takes exponential time for (1+1) EA-BM to find the MST of G_2, in which the edges of G^T are heavier than the edges of G^C. In the following proofs, let $b = B(T)$ denote the set of bad selected edges in the tail of solution T, which have weight $3a$.

Lemma 3.1. *(1+1) EA-BM and (1+1) EA-UM do not increase the value of b during the optimization process.*

Proof. Let T' be the result of k subsequent edge insertions into T by mutation. Any changes in the structure of the solution in G^C does not change the weight and neither b. It is similar when an edge with weight $2a$ is added and the other edge with weight $2a$ is removed from the cycle. Therefore, we only consider the number of changes in G^T that the swap between $2a$ and $3a$ edges happen in the same triangle. Let b_i and b_d denote the number of swaps that increase and decrease $w(T)$, respectively. We have $|B(T')| = |B(T)| + b_\mathrm{i} - b_\mathrm{d}$ and $w(T') = w(T) + a(b_\mathrm{i} - b_\mathrm{d})$. On the other hand, the algorithms accept T' if and only if $w(T') \leq w(T)$, which implies that $b_\mathrm{i} \leq b_\mathrm{d}$. Thus, in an accepted move, the number of bad edges added to T is less than or equal to the number of added edges with weight $2a$. □

The following theorem considers the performance of (1+1) EA-UM on triangular-tailed graphs.

Theorem 3.2. *(1+1) EA-UM finds the MST of triangular-tailed graph $G \in \{G_1, G_2\}$ in $\Theta(n^2 \log n)$ steps with probability $1 - o(1)$.*

Proof. Here, we follow the proof of Claim 10 in [23]. Note that we can focus on G^T since the initial solution is a random spanning tree and all weights in G^C are equal. Moreover, the MST contains all $2a$ edges and no $3a$ edge. Since (1+1) EA-UM does not increase b (Lemma 3.1), we need to calculate the expected time to achieve $b = 0$. In order to reduce b by one, the algorithm needs to insert a $2a$ edge and remove the $3a$ edge from the resulting cycle. The probability of adding only one edge is the probability of zero events in the Poisson distribution, which is equal to e^{-1}, and there are b specific $2a$ edges that need to be added. Since the maximum size of a consequent cycle is 3, removing the $3a$ edge happens with the constant probability $1/3$. Hence, the probability of swapping a $2a$ edge with the $3a$ edge in a required triangle is $b/(3em)$ that happens in expected time $3em/b$ by the waiting-time argument. Let $T_{(1+1)\,\mathrm{EA\text{-}UM}}$ denote the first hitting time that (1+1) EA-UM finds the MST. Since b is at most n, we obtain the following upper bound on the expected time with probability $1 - o(1)$

$$E[T_{(1+1)\,\mathrm{EA\text{-}UM}}] \leq \sum_{k=1}^{n} 3e \cdot \frac{m}{k} \leq 3en^2 H_n$$
$$\leq 3en^2(\log n + 1) = O(n^2 \log n).$$

Now we prove the lowe bound. Similar to the argument in the proof of the coupon collector's theorem (see, e. g., [20]), the lower bound $3en^2(\log n + 1) - cn^2$ holds with the probability $1 - e^{-e^c}$, if (1+1) EA-UM only adds one edge in each iteration. Setting $c = \frac{\log n}{2}$, the lower bound for the expected time is $\Omega(n^2 \log n)$ with probability $1 - o(1)$. Let k-step refer to the iterations that k triangle edges are chosen for the mutation step and note that $k \leq 3n/4$. It is enough to bound the contribution of k-steps on b during $\alpha n^2 \log n$ iterations for a constant $\alpha > 0$. The probability of a k-step for a constant $k \geq 1$ is

$$p_k^{\mathrm{UM}} = \frac{e^{-1}}{(k-1)!} \cdot \binom{3n/4}{k} \cdot \left(\frac{1}{m}\right)^k = \theta(n^k m^{-k}) = \theta(n^{-k}),$$

where the first term is the probability of $k - 1$ events in the Poisson distribution with $\lambda = 1$. Note that (1+1) EA-UM always adds at least one edge and $p_0^{\mathrm{UM}} = 0$. Within $\Theta(n^2 \log n)$ iterations, the expected number of 2-steps is $O(\log n)$ and there are $o(1)$ k-steps with $k > 2$. Each 2-step reduces b by at most 2. On the other hand, in a random spanning tree, each triangle contains a bad edge with probability $2/3$. Thus, b is at least $n/8 = \Theta(n)$ with probability $1 - e^{-\Omega(n)}$, using a Chernoff bound with $\delta = 2/8$. Hence, with the probability $1 - o(1)$, the expected time for (1+1) EA-UM to find the MST is

$$E[T_{(1+1)\,\mathrm{EA\text{-}UM}}] = \Theta\left(\sum_{k=1}^{b-2\log n} \frac{m}{k}\right) = \Theta\left(\sum_{k=1}^{b} \frac{n^2}{k} - \sum_{k=b-2\log n}^{b} \frac{n^2}{k}\right)$$
$$= \Theta\left(\sum_{k=1}^{b} \frac{n^2}{k} - \sum_{k=1}^{2\log n} \frac{n^2}{k}\right) = \sum_{k=1}^{\Theta(n)} \frac{n^2}{k} - \sum_{k=1}^{O(\log n)} \frac{n^2}{k}$$
$$= \Theta(n^2 \log n) - O(n^2 \log \log n) = \Theta(n^2 \log n).$$
□

Now, we consider the performance of (1+1) EA-BM on the graphs G_1 and G_2.

Lemma 3.3. *Using the biased mutation with probability $q_\mathrm{b}(e)$, the probability of selecting edge e with rank $r = O(n)$ is $\Theta(1/n)$.*

Proof. Considering the denominator of $q_\mathrm{b}(e)$, we have

$$\sum_{i=1}^{m} a^{i/2} = \frac{\sqrt{a} - a^{(m+1)/2}}{1 - \sqrt{a}} = \frac{(1 - o(1)) \cdot \sqrt{a}}{1 - \sqrt{a}}$$
$$= \frac{(1 - o(1)) \cdot \sqrt{n-1}}{\sqrt{n} - \sqrt{n-1}} = \Theta(n).$$

Since $r = O(n)$, for the numerator we have $1 \geq (1 - \frac{1}{n})^{r/2} \geq (1 - \frac{1}{n})^{cn} \geq e^{-c'}$, where $c < c'$ are constants. We conclude that $q_\mathrm{b}(e) = \frac{1 - o(1)}{2e^{c'}n} = \Theta(1/n)$. □

Lemma 3.3 shows that the edges of G^T in G_1 are more likely to be chosen in (1+1) EA-BM than in (1+1) EA-UM. In the following theorem, we show the effect of this property on the performance of (1+1) EA-BM.

Theorem 3.4. *(1+1) EA-BM finds the MST of G_1 in $\Theta(n \log n)$ with probability $1 - o(1)$.*

Proof. The proof is analogous to the proof of Theorem 3.2. However, we use Lemma 3.3 to tighten the probability of selecting edges from G_T. Hence, the expected waiting for the beneficial event in which a bad edge is removed from the tail is $\Theta(n/b)$. Thus, we obtain an upper bound of $O(n \log n)$.

To prove the lower bound, similar to the proof of Theorem 3.2, we use the argument of coupon collector's theorem with a similar approach used in [12]. However, it must be noted that we argue on the minimum number of edge selections such that all the bad edges are chosen for the mutation at least once. According to Lemma 3.3, the probability of selecting an edge in G^T is at least $1/cn$ for a constant c. Moreover, we have the initial number of bad edges is at least $n/8$ after the random initialization with probability $1 - o(1)$. Note that (1+1) EA-BM selects at least one edge in each iteration.

Therefore, $(1 - 1/cn)^t$ is the probability of no triangle edge is selected after t iterations. Consequently, the probability of flipping at least one triangle edge in t iterations is $1 - (1 - 1/cn)^t$ that implies $(1 - (1 - 1/cn)^t)^{n/8}$ is the probability of selecting all of the $n/8$ bad edges at least once. Hence, the probability that at least one bad edge has never been selected in t iterations is $1 - (1 - (1 - 1/cn)^t)^{n/8}$. Finally, the probability that (1+1) EA-BM does not attempt to remove at least one bad edge in $t = (n - 1) \ln n$ steps is $1 - (1 - (1 - 1/cn)^{(n-1)\ln n})^{n/8} \geq 1 - e^{-1/8c}$.

Therefore, (1+1) EA-BM needs $\Omega(n \log n)$ iterations to find the MST with probability of $1 - e^{-1/8c} - o(1)$, which completes the proof. □

Although (1+1) EA-BM efficiently finds the MST of G_1, the next argument shows that, in graphs similar to G_2, finding the MST takes exponential time.

LEMMA 3.5. *The probability of selecting an edge with rank $r = \Omega(n^2)$ is exponentially small.*

PROOF. According to the proof of Lemma 3.3, it is enough to show that the enumerator of $p(r)$ is exponentially small when $r > cn^2$ for some constant c. To this aim, we have

$$\left(1 - \frac{1}{n}\right)^{\frac{r}{2}} \leq \left(1 - \frac{1}{n}\right)^{\frac{c}{2}n^2} \leq e^{-\frac{c}{2}n} = O(e^{-n}).$$

□

THEOREM 3.6. *The expected time for (1+1) EA-BM to find the MST of G_2 is exponential.*

PROOF. In G_2, edges of G^T have higher weights than the edges of G^C. Since there are $\Omega(n^2)$ edges in G^C, the rank of edges of G^T is $\Omega(n^2)$. Using the result of Lemma 3.5, the probability of selecting any of the edges of G^T is $O(e^{-n})$. Hence, the expected time to select each of these edges for the mutation step is $\Omega(e^n)$. This implies that, in expectation, (1+1) EA-BM needs exponential time to reduce the value of b by one; consequently, it needs exponential time to find the MST of G_2. □

In the following, we analyze the effect of using both mutation strategies simultaneously in (1+1) EA-MM. Note that in every t iterations, (1+1) EA-MM performs $t/3$ uniform mutations and $t - t/3$ biased mutations with probability of $1 - o(1)$. This implies that repeating (1+1) EA-MM $c \geq 4$ times, the results of Theorems 3.2 and 3.4 also hold for the (1+1) EA-MM. However, since (1+1) EA-MM benefits from the uniform mutation in half of the iterations, it is also able to find the MST of G_2 in $\Theta(n^2 \log n)$.

This is the motivation to analyze the performance of (1+1) EA-MM on general graphs. For arbitrary graph G, let $w(T^i)$ be the weight of T^i, the spanning tree achieved by the algorithm in iteration i, and T^* be the minimum spanning tree. We define

$$g(T^i) = w(T^i) - w(T^*),$$

the weight gap that the algorithm needs to cover to reach the MST. Note that a MST is not necessarily unique but its weight is unique. We also redefine 1-step as an iteration that the algorithm adds only one edge and removes a random edge from the resulting cycle. Using a similar representation of Lemma 1 in [23], the following lemma presents how 1-steps contribute to reduce the value of $g(T)$.

LEMMA 3.7. *Let solution T be an arbitrary spanning tree. There exists a set of $k \in \{1, \cdots, n - 1\}$ different accepted 1-steps that if happen in any order transform T to T^* and reduce $w(T)$ by $g(T)/k$ on average.*

PROOF. Let $E(T)$ and $E(T^*)$ denote the edges of T and T^*, respectively. Using an existence proof, Kano [16] proved that there is a bijection $\alpha : E(T^*)\backslash E(T) \rightarrow E(T)\backslash E(T^*)$ such that $w(e) \leq w(\alpha(e))$ and adding e to T creates a cycle that includes $\alpha(e)$. Let $k = |E(T^*)\backslash E(T)|$. Swapping all the edges $e \in E(T^*) \setminus E(T)$ with α transforms T to T^* and reduces $g(T)$ to zero. Thus, each of these good swaps decreases the value of $g(T)$ on average by $g(T)/k$. Moreover, any 1-step that does a good swap is accepted since it results in a solution that is not worse than T. □

Using the result of Lemma 3.7 we prove a performance bound on (1+1) EA-MM on arbitrary graphs.

THEOREM 3.8. *Starting from a random spanning tree, (1+1) EA-MM finds the minimum spanning tree in expected time $O(n^3 \log(n \cdot w_{\max}))$, where $w_{\max}$ is the maximum weight of the edges.*

PROOF. Let $\Delta(g) = g(T^i) - g(T^{i+1})$ be the contribution of the algorithm in reducing the value of g in one iteration. The probability of having a 1-step equals to the probability of having zero events in the Poisson distribution which is $1/e$. Thus, with the probability of $1/(2enm)$, the uniform strategy causes a 1-step such that a specific edge e is added and a specific edge from the created cycle is removed. From Lemma 3.7 there are k good swaps. Therefore, the probability of a good swap in a 1-step with uniform strategy is $k/(2enm)$. Since a good swap reduces the value of $g(T)$ on average by $g(T)/k$, for $\Delta(g)$ we have

$$E[\Delta(g)] = \frac{g(T)}{k} \cdot \frac{k}{2enm}.$$

Since the the maximum value of $g(T)$ is $n \cdot w_{\max}$, using the multiplicative drift theorem [8] with $\delta = 1/(2enm)$, the expected first hitting time that $g(T) = 0$ is upper bounded by

$$\frac{\ln(n \cdot w_{\max}) + 1}{\left(\frac{1}{2enm}\right)} = O(n^3 \log(n \cdot w_{\max})).$$

□

Although (1+1) EA-MM guarantees a polynomial expected time to find T^* for any arbitrary graph, experiments by Raidl et al. showed that in many random graphs, all the edges of T^* have rank $O(n)$. This implies that the expected time for (1+1) EA-MM to find the MST improves to $O(n^2 \log(n \cdot w_{\max}))$ in many applications, since the probability performing a beneficial step improves to $1/(2em)$.

4 MULTI-OBJECTIVE PROBLEM

In this section we consider the multi-objective version of the minimum spanning tree problem. Firstly, we introduce the ranking of the edges in multi-objective space and experimentally show a considerably good approximation for the appearance of edges in an moMST according to their ranks. Using the approximation, we analyze the performance of GSEMO-UM and GSEMO-BM dealing with two different types of graphs.

4.1 Experimental Approximation

The work by Raidl. et al. [25] considered the single-objective scenario and lays the groundwork for our empirical study. As a reminder: the authors showed that low rank edges have a much higher probability to be part of MSTs. In the setting of multiple conflicting objectives similar assumptions are reasonable, i. e., that non-dominated spanning trees are more likely composed of "low-rank" edges for an appropriate definition of "rank". In a recent study Bossek et al. [3] considered different ranking definitions in the bi-objective case. More precisely, they considered (1) the non-domination level and (2) the domination number of an edge to define the rank and established a total order on the edges with low ranks being favored. Similar to Raidl's work, they conducted an empirical study and estimated the probability $p^{\mathrm{m}}(r)$ of edges to be part of at least one spanning tree as a function of its rank r for different graph classes (more details in the following). They, next, empirically evaluated the convergence speed of biased edge selection strategies in comparison to the baseline of random uniform selection. They obtained significant improvements, particularly in the case where the domination number $d(e) = |\{e' \in E \mid w(e') \geq w(e)\}|$ was adopted for the definition of rank and the probability of choosing an edge with edge r for insertion was set to

$$q_{\mathrm{b}}^{\mathrm{m}}(r) = \frac{p^{\mathrm{m}}(r)}{\sum_r p^{\mathrm{m}}(r)},$$

i. e., proportional to its probability of appearance in non-dominated solutions. We catch up on their work and illustrate empirically, that $\beta \cdot ((n-1)/n)^r$ – similar to Raidl's results – is indeed a good approximation for the probability $p^{\mathrm{m}}(r)$. In line with Bossek et al., our empirical study is based on different graph types reflecting different levels of density and edge weight distribution. Complete graphs (CEG for Complete Edge Generation) with n nodes placed uniformly at random in $[0, 100]^2$ are studied alongside graphs where the interconnection of nodes is based on a Delauney triangulation of the point cloud in the Euclidean plane (Delauney Edge Generation). Note that in the latter case $m = \Theta(n)$. Edge weights $w_i(e)$, $i = 1, 2$ either both are realizations of uniform random numbers stemming from a $\mathcal{U}[5, 200]$-distribution (RNDRND; in consistence with [17, 31]) or the first weight corresponds to the Euclidean distance between the nodes in the plane and the second weight is sampled from a $\mathcal{U}[5, 200]$ distribution (EUCRND). For each graph type, i. e., CEG-RNDRND, CEG-EUCRND, DEG-RNDRND and DEG-EUCRND we consider $n \in \{25, 50, 100, 250\}$.

The estimation of $p^{\mathrm{m}}(r)$ follows [3]. Here, we describe the procedure in a nutshell and refer the interested reader to the original work. First consider a single random graph $G = (V, E)$ of a given graph type and problem size n. For each edge e, we calculate the number of non-dominated spanning trees that e is part of [2], termed the share $s(e)$, and estimate the probability of r-ranked edges by the average of all shares of the corresponding rank. We repeat this process for 1000 random graphs of the corresponding graph type and $n \in \{25, 50, 100, 250\}$ and use the mean probability over all 1000 instances as the final estimate for $p^{\mathrm{m}}(r)$.

Figure 2 shows the estimations of $p^{\mathrm{m}}(r)$, the probability of rank-r edges to be part of at least one non-dominated spanning tree, separated by graph class and number of nodes. We present results for $n \in \{100, 250\}$ due to space limitations[3]. The estimations are accompanied by fitted regression models of the form $\beta \cdot ((n-1)/n)^r$. We observe that the model mostly adheres quite well to the data. These observations are supported by the results of a regression analysis. Here, the R^2 values – a measure for the fraction of variance in the data explained by the model – takes values close to 1 with a minimum of 0.8893 for CEG-EUCRND graphs with $n = 250$ nodes. Additionally, the root mean squared error (RMSE) values, i. e., the mean deviation of the model predictions to the data, are very low consistently. All in all the experiments support our parametric model assumption for different dense and sparse graphs. As a consequence, we use this empirical estimate for our upcoming theoretical runtime analysis.

4.2 Theoretical Analysis

Motivated by the experimental results, we use $p^{\mathrm{m}}(r) = \beta \cdot ((n-1)/n)^r$ as the approximation for the probability of an edge with domination number r appears in the moMST. As β consistently takes values in $(0, 1)$ throughout the experiments, we drop this constant factor in subsequent investigations. Note that we break rank ties randomly. Hence, we have $m = |E|$ different edges with m different probabilities. Using Bossek et al. [3] approach, for each edge e with domination number r we set

$$q(e) = q_{\mathrm{b}}^{\mathrm{m}}(r)$$

for the probability of choosing e in the mutation step in Algorithm 2. Using the same arguments as in Lemma 3.3, we have the following lemma for biased mutation in the multi-objective setting.

LEMMA 4.1. *Using the biased mutation with probability $q(e) = q_{\mathrm{b}}^{\mathrm{m}}(r)$, the probability of selecting edge e with domination count $r = O(n)$ is $\Theta(1/n)$.*

Again, we consider the triangular-tailed graph in two versions G_1^{m} and G_2^{m}. Both graphs contain η triangles in the tail. In each triangle, the two upper edges have weights $(1, 2)$ while the bottom edge has weight $(2, 1)$. The difference lies in the composition of the clique part G^C. Here, in G_1^{m} all edges have the same weight (k, k), $k > 2$ while in G_2^{m} there exists a subset $G^S = \{e_1, \ldots, e_l\} \subseteq G^C$ of size $l \leq (n/2 - 1)$ with $w(e) = (u, u)$, $u > 2$, for each edge $e \in G^S$ and $w(e) = (k, k)$, $k > u + n + 1$, for all remaining clique edges. We also assume that the edges in G^S do not create any cycle. Let us at this point retain the following: every non-dominated spanning tree of G_1^{m} contains an arbitrary spanning tree on G^C as a subgraph. In contrast, in G_2^{m} every non-dominated spanning tree must necessarily contain G^S as a sub-graph.

Let us briefly state our goals here. We denote by $\mathcal{T}^*$ the set of non-dominated spanning trees for a given graph and by $\mathcal{F} = w(\mathcal{T}^*)$ its image, i. e., the set of all Pareto-optimal objective vectors. We seek to locate for each $f \in \mathcal{F}$ a spanning tree $T^* \in \mathcal{T}^*$.

LEMMA 4.2. *For both G_1^{m} and G_2^{m} we have $|\mathcal{F}| = \Theta(n)$.*

---

[2]The set of non-dominated spanning trees is approximated by a simple weighted-sum approach minimizing $\lambda w_1(T) + (1 - \lambda)w_2(T)$ for equidistantly sampled $\lambda = k/1000$, $k = 0, \ldots, 1000$.

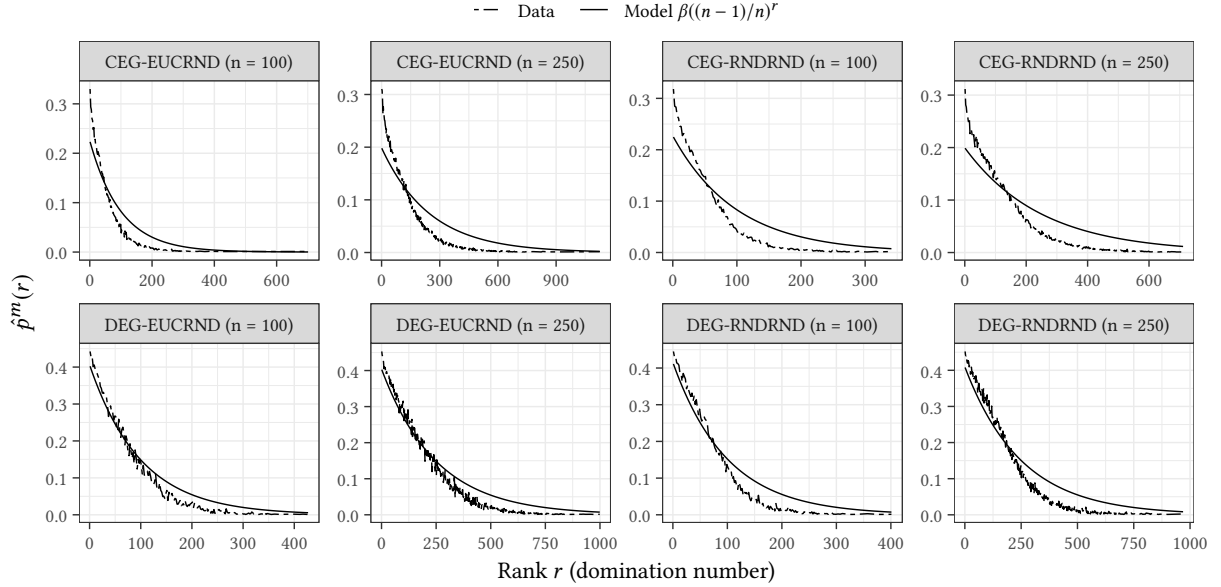[3]Omitted results for $n \in \{25, 50\}$ show the same patterns.

Figure 2: Empirical probabilities $p^{\mathrm{m}}(r)$ of edges to be part of at least one non-dominated spanning tree as a function of its rank r measured by the domination number (lower is better). The empirical data is accompanied by regression models of the form $\beta \cdot ((n-1)/n)^r$.
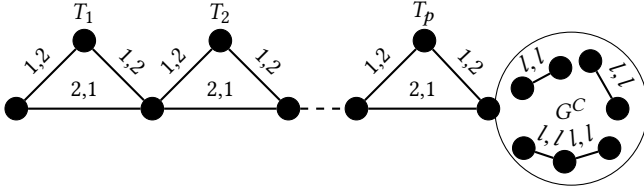

Figure 3: Triangular-tailed graph G with a chain of $p = n/4$ triangles and a giant component $G^C = K_{n/2}$. [23]

PROOF. Let us first consider the clique part. In G_1^{m} each spanning tree of G^C has equal weight, we may fix an arbitrary one. In G_2^{m} the non-dominated spanning tree of G^C must include all the edges of G^S. Thus, for each graph type G_1^{m} and G_2^{m}, the contribution of the edges of G^C in objective values are the same. Since the triangular tail is identical for both G_1^{m} and G_2^{m}, the following observations hold for both versions. Every non-dominated spanning tree contain exactly two edges of each triangle, in particular at least one edge with weight $(1, 2)$, i. e. there are at least η edges of weight $(1, 2)$ in each Pareto solution. Hence, for each non-dominated spanning tree the weight of the triangular part is

$$\eta \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + r \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + (\eta - r) \cdot \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3\eta - r \\ 3\eta + r \end{bmatrix},$$

where $0 \leq r \leq \eta$ is the number of triangles that have two upper edges in the spanning tree. Together with our observations in the clique part, this implies $r \in \Theta(n)$ and, as a direct consequence, $|\mathcal{F}| = \Theta(n)$. □

Let $f_0, f_1, \ldots, f_p \in \mathcal{F}$ be the objective vectors in ascending order of the first weight (and thus in descending order of the second weight). In the following, we show that we can easily move between Pareto-optimal spanning trees with distinct weights. We use the notation $d(T, T') := |T \setminus T'|$ and speak about distance of spanning trees in terms of the necessary edge exchange operations needed to transform T to T'.

LEMMA 4.3. *For each non-dominated spanning tree T in G_1^{m} and G_2^{m} with $w(T) = f_i, 0 \leq i \leq \eta$, there is a non-dominated spanning tree T' with $d(T, T') = 1$ such that*

- $w(T') = f_{i+1}$ *for $0 \leq i \leq \eta - 1$ or*
- $w(T') = f_{i-1}$ *for $1 \leq i \leq \eta$.*

PROOF. We only prove the first case. The proof for the other case is similar. Consider a non dominated spanning tree T with $w(T) = f_i, 0 \leq i \leq \eta - 1$. T contains exactly $(\eta - i)$ edges of weights $(2, 1)$ in the triangular-tail part. Now we obtain T' by including one of the $\eta - (\eta - i) = i$ remaining edges of weight $(2, 1)$ and dropping a $(1, 2)$ weighted edge on the resulting cycle. It follows that $w(T') = f_{i+1}$ and clearly $d(T, T') = 1$. □

Lemma 4.3 states that once we found a single non-dominated spanning tree it is easy to obtain the others.

THEOREM 4.4. *On G_1^{m}, given an initial spanning tree T, GSEMO-UM needs expected time $O(n^3 \log n)$ to cover the Pareto front.*

PROOF. Let T be a spanning tree with $w(T) = f_i$ and $b(T)$ denote the number of triangle edges with weight $(2, 1)$ for T. We shall refer those edges *bottom edges* in the following. Since, $w(T) = f_i$ clearly $b(T) = i$. By Lemma 4.3 we can move to a tree with weight vector f_{i+1} or f_{i-1} by adding or removing a bottom edge. In GSEMO (see Algorithm 2) achieving $f(i + 1)$ happens with probability at least

$$\left(\frac{1}{i+1} \right) \cdot \left(e^{-1} \cdot \frac{(\eta - i)}{m} \right) \cdot \left(\frac{2}{3} \right) = \frac{2(\eta - i)}{3em(i+1)}.$$

Here, the first term is the probability to select the individual T with $w(T) = f_i$ such that T' with $w(T') = f_{i+1}$ is not included in the population yet, the second term is the probability for the 0 event of a Pois($\lambda = 1$) distribution, i. e., to add exactly one edge to the sampled solution, and the third term is the probability to remove one of the non-bottom edges from the resulting cycle. Adopting waiting time arguments, the expected number of iterations until f_{i+1} is achieved is bounded from above by $3em(i+1)/2(p-i)$. Hence, the total time until the population of GSEMO contains each one solution for each Pareto-optimal objective vector $f_i, i = 0, \dots, n$ – only by adding bottom edges and starting with a solution with trade-off f_0 in the worst case – is bounded by the sum

$$\sum_{i=0}^{\eta-1} \frac{3em(i+1)}{2(\eta-i)} = \frac{3em}{2} \cdot \sum_{i=0}^{\eta-1} \frac{(i+1)}{(\eta-i)} \leq \frac{3em}{2} \cdot \sum_{i=0}^{\eta-1} \frac{\eta}{(\eta-i)}$$
$$= \frac{3em\eta}{2} \cdot H_\eta = O(n^3 \log n).$$

On the other hand, to include f_{i-1}, the algorithm must choose a non-bottom edge from the triangles that include one, with probability $i/(em)$, and remove the bottom edge with probability $1/3$. Thus, the probability of this event is $i/3em(i+1)$, i. e., the expected number of iterations for such event happen is $3em(i+1)/i$. Therefore – only by decreasing the number of bottom edges and starting from f_η in the worst case – the expected time for GSEMO to achieve all the objective vectors in the Pareto front is upper bounded by the sum

$$\sum_{i=1}^{\eta} \frac{3em(i+1)}{i} = 3em \cdot \sum_{i=1}^{\eta} \frac{(i+1)}{i} \leq 3em \cdot \sum_{i=1}^{\eta} \frac{\eta}{i}$$
$$= 3em\eta \cdot H_\eta = O(n^3 \log n).$$

All together, since one of the cases is always available, the total upper bound is $O(n^3 \log n)$. □

Next we consider the performance of GSEMO-UM on G_2^m. In an arbitrary moMST T of G_2^m, let $s = |G^S \cap T|$ denote the number of optimal edges G^S in T.

LEMMA 4.5. *For two solutions $T_1, T_2 \in G_2^m$, $T_1 > T_2$ if and only if $s_1 > s_2$.*

PROOF. Considering the proof of lemma 4.2, the difference between the objective values of T_1 and T_2 that is caused because of the chosen tail edges is at most n. On the other hand, increasing s improves both objective values by at least $n + 1$. Thus any solution that has larger s have strictly better objective value in both objectives. □

Lemma 4.5 results in the fact that all the solutions in the population set of GSEMO have the same value of s. Note that GSEMO starts with a spanning tree and any offspring is also a spanning tree.

THEOREM 4.6. *On G_2^m, given an initial spanning tree T, GSEMO-UM needs expected time $O(n^3 \log n)$ to cover the Pareto-front.*

PROOF. We consider two phases in this proof. The first phase is to find the solution T with $s = l$, i. e. T contains all the edges of G^S.

After this phase, we know that any offspring is a Pareto-optimal solution. The next phase is to cover the whole Pareto front.

Note that all the solutions in the current population have the same value of $s < l$. Thus, the probability of choosing solution T with highest s is 1. To increase s, the algorithm adds edge $e \in G^S \setminus T$ with probability $(l - s)/m$. Adding e can cause a cycle with size at most n. In the worst case, there is only one edge in the cycle that can be removed without removing another optimal edge. Hence, a beneficial removing happens with probability of $1/n$. Therefore, the probability of increasing s by 1 is at least $e^{-1} \cdot \frac{l-s}{m} \cdot \frac{1}{n}$, where e^{-1} is the probability that GSEMO adds only one edge. Such mutation step happens after $O(mn/(l-s))$ iterations in expectation. The minimum initial value for s is zero and l is at most $n - 1$. Thus, the expected time for GSEMO to finish phase one is upper bounded by

$$\sum_{i=0}^{l-1} \frac{mn}{e(l-s)} \leq n^3 \sum_{i=1}^{n} \frac{1}{n} = O(n^3 \log n).$$

In the second phase, GSEMO does not accept a solution with $s < l$. Hence, the same argument as in Theorem 4.4 proves that GSEMO finishes the second phase in $O(n^3 \log n)$ expected time and this completes the proof. □

Now we consider GSEMO-BM algorithm with biased mutation that select the edges with $q(e) = q_b^m(r)$. The number of edges in the tail of G_1^m and G_2^m is the same and equal to $3n/4$. In both of the graphs, these edges dominate every other edges and consequently have lower non-domination ranks, i. e. each edge have a unique random rank within $\{1, \cdots, 3n/4\}$. Moreover, in G_2^m, edges of G^S dominate other edges of G^C. Hence, ranks $3n/4, \cdots, (3n/4) + l$ belong to the edges of G^S. Therefore, as Lemma 4.1 shows, all the edges that belong to the moMSTs in G_1^m and G_2^m have the selection probability $\Theta(1/n)$. Using the same arguments as in Theorems 4.4 and 4.6, the following result hold for the performance of GSEMO on the graphs G_1^m and G_2^m.

COROLLARY 4.7. *On G_1^m and G_2^m, given an initial spanning tree T, GSEMO-BM needs expected time $O(n^2 \log n)$ to cover the Pareto-front.*

5 CONCLUSION

We performed a rigorous asymptotic runtime analysis of evolutionary algorithms with biased mutation for the classic Minimum Spanning Tree problem. Bias in this context means that edges of low weight in the single-objective case and of low domination number in the multi-objective case are assigned a higher probability of mutation. Our findings reveal that bias is blessing and curse at the same time. While a significant time complexity speedup can be achieved in some cases, bias may also lead to exponential expected optimization time if edges of high rank are part of optimal solutions. We showed that using the biased and unbiased mutations simultaneously is the key to avoid the extreme cases of bias. We will consider the generalization of the achieved results to more general graph classes in future work.

ACKNOWLEDGMENT

REFERENCES

[1] Auger, A. and Doerr, B. (2011). *Theory of randomized search heuristics: Foundations and recent developments*, volume 1. World Scientific.

[2] Bossek, J. and Grimme, C. (2017). A pareto-beneficial sub-tree mutation for the multi-criteria minimum spanning tree problem. In *Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 3280–3287, Honolulu, Hawai. IEEE.

[3] Bossek, J., Grimme, C., and Neumann, F. (2019). On the benefits of biased edge-exchange mutation for the multi-criteria spanning tree problem. In Auger, A. and Stützle, T., editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pages 516–523. ACM.

[4] Broder, A. Z. (1989). Generating random spanning trees. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 442–447. IEEE Computer Society.

[5] Chiong, R., Weise, T., and Michalewicz, Z. (2011). *Variants of Evolutionary Algorithms for Real-World Applications*. Springer Publishing Company, Incorporated.

[6] Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[7] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.

[8] Doerr, B. and Goldberg, L. A. (2013). Adaptive drift analysis. *Algorithmica*, 65(1):224–250.

[9] Doerr, B., Hebbinghaus, N., and Neumann, F. (2006). Speeding Up Evolutionary Algorithms Through Restricted Mutation Operators. In Runarsson, T. P., Beyer, H.-G., Burke, E., Merelo-Guervós, J. J., Whitley, L. D., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX*, pages 978–987, Berlin, Heidelberg. Springer Berlin Heidelberg.

[10] Doerr, B., Hebbinghaus, N., and Neumann, F. (2007). Speeding Up Evolutionary Algorithms Through Asymmetric Mutation Operators. *Evolutionary Computation*, 15(4):401–410.

[11] Doerr, B., Johannsen, D., and Winzen, C. (2012). Multiplicative drift analysis. *Algorithmica*, 64(4):673–697.

[12] Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theor. Comput. Sci.*, 276(1-2):51–81.

[13] Friedrich, T., Göbel, A., Quinzan, F., and Wagner, M. (2018a). Heavy-Tailed Mutation Operators in Single-Objective Combinatorial Optimization. In Auger, A., Fonseca, C. M., Lourenço, N., Machado, P., Paquete, L., and Whitley, D., editors, *Parallel Problem Solving from Nature – PPSN XV*, pages 134–145, Cham. Springer International Publishing.

[14] Friedrich, T., Quinzan, F., and Wagner, M. (2018b). Escaping Large Deceptive Basins of Attraction with Heavy-tailed Mutation Operators. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 293–300, New York, NY, USA. ACM.

[15] Jansen, T. and Sudholt, D. (2010). Analysis of an Asymmetric Mutation Operator. *Evolutionary Computation*, 18(1):1–26.

[16] Kano, M. (1987). Maximum and k-th maximal spanning trees of a weighted graph. *Combinatorica*, 7(2):205–214.

[17] Knowles, J. and Corne, D. (2001). A Comparison of Encodings and Algorithms for Multiobjective Minimum Spanning Tree Problems. *Evolutionary Computation*, 1:544–551.

[18] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

[19] Madry, A., Straszak, D., and Tarnawski, J. (2015). Fast generation of random spanning trees and the effective resistance metric. In Indyk, P., editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 2019–2036. SIAM.

[20] Motwani, R. and Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press. Page 61.

[21] Neumann, F. (2007). Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620–1629.

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

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

[24] Neumann, F. and Witt, C. (2010). Ant colony optimization and the minimum spanning tree problem. *Theoretical Computer Science*, 411(25):2406–2413.

[25] Raidl, G. R., Koller, G., and Julstrom, B. A. (2006). Biased mutation operators for subgraph-selection problems. *IEEE Trans. Evolutionary Computation*, 10(2):145–156.

[26] Reichel, J. and Skutella, M. (2009). On the size of weights in randomized search heuristics. In *Foundations of Genetic Algorithms, 10th ACM SIGEVO International Workshop, FOGA 2009, Orlando, Forida, USA, January 9-11, 2009, Proceedings*, pages 21–28.

[27] Roostapour, V., Pourhassan, M., and Neumann, F. (2019). Analysis of baseline evolutionary algorithms for the packing while travelling problem. In Friedrich, T.,

Doerr, C., and Arnold, D. V., editors, *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA 2019, Potsdam, Germany, August 27-29, 2019*, pages 124–132. ACM.

[28] Rudolph, G. (1994). Convergence of non-elitist strategies. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Orlando, Florida, USA, June 27-29, 1994*, pages 63–66. IEEE.

[29] Ruzika, S. and Hamacher, H. W. (2009). A survey on multiple objective minimum spanning tree problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5515 LNCS:104–116.

[30] Witt, C. (2014). Revised analysis of the (1+1) ea for the minimum spanning tree problem. In *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*, pages 509–516.

[31] Zhou, G. and Gen, M. (1999). Genetic Algorithm Approach on Multi-Criteria Minimum Spanning Tree Problem. *European Journal of Operational Research*, 114:141–152.