Randomized Local Search, Evolutionary Algorithms, and the Minimum Spanning Tree Problem

Frank Neumann[1] and Ingo Wegener[2]*

[1] Inst. für Informatik und Prakt. Mathematik, Christian-Albrechts-Univ. zu Kiel, 24098 Kiel, Germany
fne@informatik.uni-kiel.de
[2] FB Informatik, LS 2, Univ. Dortmund, 44221 Dortmund, Germany
ingo.wegener@uni-dortmund.de

Abstract. Randomized search heuristics, among them randomized local search and evolutionary algorithms, are applied to problems whose structure is not well understood, as well as to problems in combinatorial optimization. The analysis of these randomized search heuristics has been started for some well-known problems, and this approach is followed here for the minimum spanning tree problem. After motivating this line of research, it is shown that randomized search heuristics find minimum spanning trees in expected polynomial time without employing the global technique of greedy algorithms.

1 Introduction

The purpose of this paper is to contribute to the growing research area where randomized search heuristics are analyzed with respect to the expected time until they consider an optimal search point. Such an approach should support the understanding how these heuristics work, should guide the choice of the free parameters of the algorithms, and should support the teaching of heuristics. This is a growing research area, some general results can be found in Papadimitriou, Schäffer, and Yannakakis (1990) for randomized local search and Beyer, Schwefel, and Wegener (2002) and Droste, Jansen, and Wegener (2002) for evolutionary algorithms.

Search heuristics are mainly applied to problems whose structure is not well understood but the analysis has to start with problems whose structure is well understood. One cannot hope to beat the best problem-specific algorithms on these problems. Hence, the main purpose is to study the behavior of randomized search heuristics which find many applications in real-world optimization problems. For combinatorial optimization, this approach has been started only

* This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531) and by the German-Israeli Foundation (GIF) in the project "Robustness Aspects of Algorithms".

recently. There are results on sorting as the minimization of unsortedness and on shortest paths problems (Scharnow, Tinnefeld, and Wegener (2002)), on maximum matchings (Sasaki and Hajek (1988) for simulated annealing and Giel and Wegener (2003) for randomized local search and evolutionary algorithms), and on minimum graph bisections (Jerrum and Sorkin (1998) for the Metropolis algorithm).

Here we study the well-known problem of computing minimum spanning trees in graphs with n vertices and m edges. The problem can be solved by greedy algorithms. The famous algorithms due to Kruskal and Prim have worst-case run times of $O((n+m)\log n)$ and $O(n^2)$, respectively, see any textbook on efficient algorithms, e.g., Cormen, Leiserson, and Rivest (1990). Greedy algorithms use global ideas. Considering only the neighborhoods of two vertices u and v, it is not possible to decide whether the edge $\{u, v\}$ belongs to some minimum spanning tree. Therefore, it is interesting to analyze the run times obtainable by more or less local search heuristics like randomized local search and evolutionary algorithms. One goal is to estimate the expected time until a better spanning tree has been found. For large weights, there may be exponentially many spanning trees with different weights. Therefore, we also have to analyze how much better the better spanning tree is. This is indeed the first paper where the expected fitness increase is estimated for problems of combinatorial optimization.

As already argued, we do not and cannot hope to beat the best algorithms for the minimum spanning tree problem. This can be different for two generalizations of the problem. First, one is interested in minimizing the weight of restricted spanning trees, e.g., trees with bounded degree or trees with bounded diameter. These problems are NP-hard, and evolutionary algorithms are competitive, see Raidl and Julstrom (2003). Second, one is interested in the multi-objective variant of the problem. Each edge has k weights, and one looks for the Pareto optimal spanning trees with respect to the weight functions, see Hamacher and Ruhe (1994) for the general problem and Zhou and Gen (1999) for the design of evolutionary algorithms. Many polynomially solvable problems have NP-hard multi-objective counterparts, see Ehrgott (2000). None of these papers contains a run time analysis of the considered search heuristics. We think that it is essential to understand how the heuristics work on the unrestricted single-objective problem before one tries to analyze their behavior on the more difficult variants.

After having motivated the problem to analyze randomized search heuristics on the minimum spanning tree problem, we give a survey on the rest of this paper. In Section 2, we describe our model of the minimum spanning tree problem and, in Section 3, we introduce the randomized search heuristics which will be considered in this paper. The theory on minimum spanning trees is well established. In Section 4, we deduce some properties of local changes in non-optimal spanning trees which are applied in the run time analysis presented in Section 5. After the discussion of some generalizations in Section 6, we finish with concluding remarks.

2 Minimum Spanning Trees

This classical optimization problem has the following description. Given an undirected connected graph $G = (V, E)$ on n vertices and m weighted edges, find an edge set $E' \subseteq E$ of minimal weight, which connects all vertices. The weight of an edge set is the sum of the weights of the considered edges. Weights are positive integers. Therefore, the solution is a tree on V, a so-called spanning tree. One can also consider graphs which are not necessarily connected. Then the aim is to find a minimum spanning forest, i.e., a collection of spanning trees on the connected components. All our results hold also in this case. To simplify the notation we assume that G is connected.

There are many possibilities how to choose the search space for randomized search heuristics. This problem has been investigated intensively by Raidl and Julstrom (2003). Their experiments point out that one should work with so-called "edge sets". The search space equals $S = \{0, 1\}^m$, where each position corresponds to one edge. A search point $s \in S$ corresponds to the choice of all edges e_i, $1 \leq i \leq m$, where $s_i = 1$. In many cases, many search points correspond to non-connected graphs and others correspond to connected graphs with cycles, i.e., graphs which are not trees. If all graphs which are not spanning trees get the same "bad" fitness, it will take exponential time to find a spanning tree when we apply a general search heuristic. We will investigate two fitness functions w and w'. The weight of e_i is denoted by w_i. Let $w_{\max}$ be the maximum weight. Then $w_{\text{ub}} := n^2 \cdot w_{\max}$ is an upper bound on the weight of each edge set. Let

$$w(s) := (c(s) - 1) \cdot w_{\text{ub}}^2 + (e(s) - (n - 1)) \cdot w_{\text{ub}} + \sum_{i|s_i=1} w_i$$

be the first fitness function where $c(s)$ is the number of connected components of the graph described by s and $e(s)$ is the number of edges in this graph. The fitness function has to be minimized. The most important issue is to decrease $c(s)$ until we have graphs connecting all vertices. Then we have at least $n-1$ edges, and the next issue is to decrease $e(s)$ under the condition that s describes a connected graph. Hence, we look for spanning trees. Finally, we look for minimum spanning trees.

It is necessary to penalize non-connected graphs since the empty graph has the smallest weight. However, it is not necessary to penalize extra connections since breaking a cycle decreases the weight. Therefore, it is also interesting to investigate the fitness function

$$w'(s) := (c(s) - 1)w_{\text{ub}} + \sum_{i|s_i=1} w_i.$$

The fitness function w' is appropriate in the black-box scenario where the scenario contains as little problem-specific knowledge as possible. The fitness function w contains the knowledge that optimal solutions are *trees*. This simplifies the analysis of search heuristics. Therefore, we always start with results on the fitness function w and discuss afterwards how to obtain similar results for w'.

3 Randomized Local Search and the (1+1) EA

Randomized local search (RLS) uses the following mutation operator:
– Choose $i \in \{1, \dots, m\}$ randomly and flip the ith bit.
Here we use the notion "choose randomly" for a choice according to the uniform distribution. This operator is not useful for most graph problems. Often the number of ones (or edges) is the same for all good search points, e. g., for TSP or minimum spanning trees. Then all Hamming neighbors of good search points are bad implying that we have many local optima. Therefore, we work with the larger neighborhood of Hamming distance 2. This mutation operator has already been discussed for maximum matchings by Giel and Wegener (2003). Finally, RLS can be described as follows.

Algorithm 1 (Randomized Local Search (RLS))

```
1.) Choose $s \in \{0,1\}^m$ randomly.
2.) Choose $b \in \{0,1\}$ randomly. If $b = 0$, choose $i \in \{1, \dots, m\}$ randomly and
    define $s'$ by flipping the $i$th bit of $s$. If $b = 1$, choose $(i,j) \in \{(k,l) \mid 1 \le k < l \le m\}$ randomly and define $s'$ by flipping the $i$th and the $j$th bit of $s$.
3.) Replace $s$ by $s'$ if $w(s') \le w(s)$.
4.) Repeat Steps 2 and 3 forever.
```

In applications, we need a stopping criterion. Here we are interested in the expected value of T_G, which measures the number of fitness evaluations until s is a minimum spanning tree. This is the expected optimization time (sometimes called expected first hitting or passage time) of RLS. Indeed, we will estimate $E(T_G)$ with respect to the parameters n, m, and $w_{\max}$.

The simple evolutionary algorithm called (1+1) EA differs from RLS in the chosen mutation operator.

Algorithm 2 (Mutation operator of (1+1) EA)

Define s' in the following way. Each bit of s is flipped independently of the other bits with probability $1/m$.

This is the perhaps most simple algorithm which can be called evolutionary algorithm. It is adopted from the well-known (1+1) ES (evolution strategy) for the optimization in continuous search spaces. In Section 6, we will argue why we believe that larger populations will be harmful. There it will also be discussed whether genetic algorithms based on crossover can be useful.

4 Properties of Local Changes of Spanning Trees

Our aim is to show the following. In the rest of this paper we denote by w_{opt} the weight of minimum spanning trees. For a non-optimal tree s, there are either many weight-decreasing local changes which, on the average, decrease $w(s)$ by an amount which is not too small with respect to $w(s) - w_{\text{opt}}$, or there are few of these local changes which, on the average, cause a larger decrease of the weight. This statement will be made precise in the following lemma.

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

Proof. This result follows directly from results in the literature on spanning trees. Kano (1987) has proved the following result by an existence proof and Mayr and Plaxton (1992) have proved the same result by an explicit construction procedure.

Let s^* be a search point describing a minimum spanning tree T^*. Let $E(T)$ and $E(T^*)$ be the edge sets of T and T^*, respectively. Let $k := |E(T^*) - E(T)|$. Then there exists a bijection $\alpha : E(T^*) - E(T) \to E(T) - E(T^*)$ such that $\alpha(e)$ lies on the cycle created in T by including e into T and the weight of $\alpha(e)$ is not smaller than the weight of e.

We consider the k 2-bit flips flipping e and $\alpha(e)$ for $e \in E(T^*) - E(T)$. They are accepted since e creates a cycle which is destroyed by the elimination of $\alpha(e)$. Performing all the k 2-bit flips simultaneously changes T into T^* and leads to a weight decrease of $w(s) - w_{\text{opt}}$. Hence, the average weight decrease of these steps is $(w(s) - w_{\text{opt}})/k$. □

The analysis performed in Section 5 will be simplified if we can ensure that we always have the same parameter k in Lemma 1. This is easy if we allow also non-accepted 2-bit flips whose weight decrease is defined as 0. We add $n - k$ non-accepted 2-bit flips to the set of the k accepted 2-bit flips whose existence is proven in Lemma 1. Then we obtain a set of exactly n 2-bit flips. The total weight decrease is at least $w(s) - w_{\text{opt}}$ since this holds for the k accepted 2-bit flips. Therefore, the average weight decrease is bounded below by $(w(s) - w_{\text{opt}})/n$. We state this result as Lemma 2.

Lemma 2. *Let s be a search point describing a spanning tree T. Then there exists a set of n 2-bit flips such that the average weight decrease of these flips is at least $(w(s) - w_{\text{opt}})/n$.*

When analyzing the fitness function w' instead of w, we may accept non-spanning trees as improvements of spanning trees. Non-spanning trees can be improved by 1-bit flips eliminating edges of cycles. A 1-bit flip leading to a non-connected graph is not accepted and its weight decrease is defined as 0.

Lemma 3. *Let s be a search point describing a connected graph. Then there exist a set of n 2-bit flips and a set of $m - (n - 1)$ 1-bit flips such that the average weight decrease of these flips is at least $(w(s) - w_{\text{opt}})/(m + 1)$.*

Proof. We consider all 1-bit flips concerning the non-T^* edges. If we try them in some arbitrary order we obtain a spanning tree T. If we consider their weight decrease with respect to the graph G' described by s, this weight decrease can be only larger. The reason is that a 1-bit flip, which is accepted in the considered sequence of 1-bit flips, is also accepted when applied to s. Then we apply Lemma 2 to T. At least the same weight decrease is possible by adding e_i and deleting a non-T^* edge with respect to G'. Altogether, we obtain at least a weight decrease of $w(s) - w_{\text{opt}}$. This proves the lemma, since we have chosen $m + 1$ flips. □

5 The Analysis of RLS and (1+1) EA for the Minimization of Spanning Trees

First, it is rather easy to prove that RLS and (1+1) EA construct spanning trees efficiently.

Lemma 4. *The expected time until RLS or (1+1) EA working on one of the fitness function w or w' has constructed a connected graph is bounded by $O(m \log n)$.*

Proof. The fitness functions are defined in such a way that the number of connected components will never be increased in accepted steps. For each edge set leading to a graph with k connected components, there are at least $k-1$ edges whose inclusion decreases the number of connected components by 1. Otherwise, the graph would not be connected. The probability of a step decreasing the number of connected components is at least $\frac{1}{2} \cdot \frac{k-1}{m}$ for RLS and $\frac{1}{e} \cdot \frac{k-1}{m}$ for (1+1) EA. Hence, the expected time until s describes a connected graph is bounded above by

$$em\left(1 + \cdots + \frac{1}{n-1}\right) = O(m \log n).$$

□

Lemma 5. *If s describes a connected graph, the expected time until RLS or (1+1) EA has constructed a spanning tree for the fitness function w is bounded by $O(m \log n)$.*

Proof. The fitness function is defined in such a way that, starting with s, only connected graphs are accepted and that the number of edges does not increase. If s describes a graph with N edges, it contains a spanning tree with $n-1$ edges, and there are at least $N-(n-1)$ edges whose exclusion decreases the number of edges. If $N = n-1$, s describes a spanning tree. Otherwise, by the same arguments as in the proof of Lemma 4, we obtain an upper bound of

$$em\left(1 + \cdots + \frac{1}{m-(n-1)}\right) = O(m \log(m-n+1)) = O(m \log n).$$

□

This lemma holds also for RLS and the fitness function w'. RLS does not accept steps only including an edge or only including two edges if s describes a connected graph. Since RLS does not affect more than two edges in a step, it does not accept steps in which the number of edges of a connected graph is increased. This does not hold for (1+1) EA. It is possible that the exclusion of one edge and the inclusion of two or more edges creates a connected graph whose weight is not larger than the weight of the given graph.

Before we analyze the expected time to turn a spanning tree into a minimum spanning tree, we investigate an example (see Figure 1).

The example graph consists of a connected sequence of p triangles and the last triangle is connected to a complete graph on q vertices. The number of

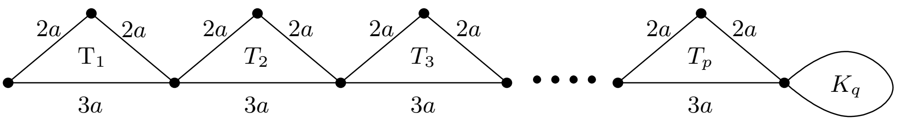

Fig. 1. An example graph with p connected triangles and a complete graph on q vertices with edges of weight 1.

vertices equals $n := 2p + q$ and the number of edges equals $m := 3p + q(q-1)/2$. We consider the case of $p = n/4$ and $q = n/2$ implying that $m = \Theta(n^2)$. The edges in the complete graph have the weight 1 and we set $a := n^2$. Each triangle edge has a weight which is larger than the weight of all edges of the complete graph altogether. Theorem 1 and Theorem 2 prove that this graph is a worst-case instance with polynomial weights.

Theorem 1. *The expected optimization time until RLS and (1+1) EA find a minimum spanning tree for the example graph equals* $\Theta(m^2 \log n) = \Theta(n^4 \log n)$ *with respect to the fitness functions* w *and* w'.

Proof. The upper bound is contained in Theorem 2. Here we prove the lower bound by investigating typical runs of the algorithm. We use the following notation. We partition the graph G into its triangle part T and its clique part C. Each search point x describes an edge set. We denote by $d(x)$ the number of triangles that are disconnected with respect to the edges chosen by x, by $b(x)$ the number of bad triangles (exactly one $2a$-edge and the $3a$-edge are chosen), by $g(x)$ the number of good triangles (exactly the two $2a$-edges are chosen), by $c(x)$ the number of complete triangles (all three edges are chosen), and by $\text{con}_G(x)$, $\text{con}_T(x)$, and $\text{con}_C(x)$ the number of connected components in the different parts of the graph. We investigate four phases of the search. The first phase of length 1 is the initialization step producing the random edge set x. In the following, all statements hold with probability $1 - o(1)$.

Claim. After initialization, $b(x) = \Theta(n)$ and $\text{con}_C(x) = 1$.

Proof. The statements can be proved independently since the corresponding parts of x are created independently. The probability that a given triangle is bad equals $1/4$. There are $n/4$ triangles and $b(x) = \Theta(n)$ by Chernoff bounds. We consider one vertex of C. It has $n/2 - 1$ possible neighbors. By Chernoff bounds, it is connected to at least $n/6$ of these vertices. For each other vertex, the probability to be not connected to at least one of these $n/6$ vertices is $(1/2)^{n/6}$. This is unlikely even for one of the remaining vertices. Hence, $\text{con}_C(x) = 1$. □

For the following phases, we distinguish the steps by the number k of flipping triangle edges and call them k-steps. Let p_k be the probability of a k-step. For RLS, $p_1 = \Theta(n^{-1})$, $p_2 = \Theta(n^{-2})$ and $p_k = 0$, if $k \geq 3$. For (1+1) EA and constant k

$$p_k = \binom{3n/4}{k} \left(\frac{1}{m}\right)^k \left(1 - \frac{1}{m}\right)^{3n/4-k} = \Theta(n^k m^{-k}) = \Theta(n^{-k}).$$

For a phase of length $n^{5/2}$, the following statements hold. The number of 1-steps equals $\Theta(n^{3/2})$, the number of 2-steps equals $\Theta(n^{1/2})$, and there is no k-step, $k \geq 3$.

Claim. Let $b(x) = \Theta(n)$ and $\text{con}_C(x) = 1$. In a phase of length $n^{5/2}$, a search point y where $b(y) = \Theta(n)$ and $\text{con}_G(y) = 1$ is produced.

Proof. By Lemma 4, the probability of creating a connected graph is large enough. Let y be the first search point where $\text{con}_G(y) = 1$. We prove that $b(y) = \Theta(n)$. All the 2-steps can decrease the b-value by at most $O(n^{1/2})$. A 1-step has two possibilities to destroy a bad triangle.

– It may destroy an edge of a bad triangle. This increases the con_G-value. In order to accept the step, it is necessary to decrease the con_C-value.
– It may add the missing edge to a bad triangle. This increases the weight by at least $2a$. No triangle edge is eliminated in this step. In order to accept the step, it is necessary to decrease the con_C-value.

However, $\text{con}_C(x) = 1$. In order to decrease this value, it has to be increased before. A step increasing the con_C-value can be accepted only if the con_T-value is decreased in the same step at least by the same amount. This implies that triangle edges have to be added. For a 1-step, the total weight is increased without decreasing the con_G-value and the step is not accepted. Hence, only the $O(n^{1/2})$ 2-steps can increase the con_C-value. By Chernoff bounds, the number of clique edges flipping in these steps is $O(n^{1/2})$. This implies that the number of bad triangles is decreased by only $O(n^{1/2})$. □

Claim. Let $b(y) = \Theta(n)$ and $\text{con}_G(y) = 1$. In a phase of length $n^{5/2}$, a search point z where $b(z) = \Theta(n)$, $\text{con}_G(z) = 1$, and $T(z)$ is a tree is produced.

Proof. Only search points x describing connected graphs are accepted, in particular, $d(x) = 0$. Let z be the first search point where $T(z)$ is a tree. Then $\text{con}_G(z) = 1$ and we have to prove that $b(z) = \Theta(n)$ and that z is produced within $n^{5/2}$ steps. A 1-step can be accepted only if it turns a complete triangle into a good or bad triangle. Such a step is accepted if no other edge flips. Moreover, $c(x)$ cannot be increased. In order to increase $c(x)$ it is necessary to add the missing edge to a good or bad triangle. To compensate this weight increase, we have to eliminate an edge of a complete triangle. Remember that we have no k-steps for $k \geq 3$. If $c(x) = l$, the probability of decreasing the c-value is at least $3l/(em)$ and the expected time to eliminate all complete triangles is $O(m \log n) = O(n^2 \log n)$. Hence, $n^{5/2}$ steps are sufficient to create z. The number of bad triangles can be decreased only in the $O(n^{1/2})$ 2-steps implying that $b(z) = \Theta(n)$. □

Claim. Let $b(z) = \Theta(n)$, $\text{con}_G(z) = 1$, and $T(z)$ be a tree. The expected time to find a minimum spanning tree is $\Omega(n^4 \log n)$.

Proof. First, we assume that only 2-steps change the number of bad triangles. Later, we complete the arguments. The expected waiting time for a 2-step flipping those two edges of a bad triangle that turn it into a good one equals $\Theta(n^4)$.

The expected time to decrease the number of bad triangles from b to $b-1$ equals $\Theta(n^4/b)$. Since b has to be decreased from $\Theta(n)$ to 0, we obtain an expected waiting time of

$$\Theta(n^4 \sum_{1\le b\le\Theta(n)} (1/b)) = \Theta(n^4 \log n). \qquad (*)$$

Similarly to the proof of the coupon collector's theorem we obtain that the optimization step if only 2-steps can be accepted equals $\Theta(n^4 \log n)$ with probability $1-o(1)$. Hence, it is sufficient to limit the influence of all k-steps, $k \neq 2$, within a time period of $\alpha n^4 \log n$ for some constant $\alpha > 0$. Again with probability $1-o(1)$, the number of 4-steps is $O(\log n)$ and there are no k-steps for $k \ge 5$. The 4-steps can decrease the number of bad triangles by at most $O(\log n)$. Because of the weight increase, a k-step, $k \le 4$, can be accepted only if it eliminates at least $\lceil k/2 \rceil$ triangle edges. Moreover, it is not possible to disconnect a good or a bad triangle. Hence, a 4-step cannot create a complete triangle. As long as there is no complete triangle, a 3-step or a 1-step has to disconnect a triangle and is not accepted. A 2-step can only be accepted if it changes a bad triangle into a good one. Hence, no complete triangles are created. The 4-steps eliminate $O(\log n)$ terms of the sum in $(*)$. The largest terms are those for the smallest values of b. We only have to substract a term of $O(n^4 \log\log n) = o(n^4 \log n)$ from the bound $\Theta(n^4 \log n)$ and this proves the claim. □

We have proved Theorem 1 since the sum of all failure probabilities is $o(1)$. □

In the following , we prove an upper bound of size $O(m^2(\log n + \log w_{\max}))$ on the expected optimization time for arbitrary graphs. This bound is $O(m^2 \log n)$ as long as $w_{\max}$ is polynomially bounded and it is always polynomially bounded with respect to the bit length of the input. Theorem 1 shows that the bound is optimal.

Theorem 2. *The expected time until RLS or (1+1) EA working on the fitness function w constructs a minimum spanning tree is bounded by $O(m^2(\log n + \log w_{\max}))$.*

Proof. By Lemmas 4 and 5, it is sufficient to investigate the search process after having found a search point s describing a spanning T. Then, by Lemma 2, there always exists a set of n 2-bit flips whose average weight decrease is at least $(w(s) - w_{\text{opt}})/n$. The choice of such a 2-bit flip is called a "good step". The probability of performing a good step equals $\Theta(n/m^2)$ and each of the good steps is chosen with the same probability. A good step decreases the difference between the weight of the current spanning tree and w_{opt} on average by a factor not larger than $1 - 1/n$. This holds independently from previous good steps. Hence, after N good steps, the expected difference of the weight of T and w_{opt} is bounded above by $(1-1/n)^N \cdot (w(s) - w_{\text{opt}})$. Since $w(s) \le (n-1)\cdot w_{\max}$ and $w_{\text{opt}} \ge 0$, we obtain the upper bound $(1-1/n)^N \cdot D$, where $D := n \cdot w_{\max}$.

If $N := \lceil (\ln 2)\cdot n \cdot (\log D + 1)\rceil$, this bound is at most $\frac{1}{2}$. Since the difference is not negative, by Markov's inequality, the probability that the bound is less than 1 is at least 1/2. The difference is an integer implying that the probability of having found a minimum spanning tree is at least 1/2. Repeating the same arguments, the expected number of good steps until a minimum spanning tree is found is bounded by $2N = O(n \log D) = O(n(\log n + \log w_{\max}))$.

By our construction, there are always exactly n good 2-bit flips. Therefore, the probability of a good step does not depend on the current search point. Hence, the expected time until r steps are good equals $\Theta(rm^2/n)$. Altogether, the expected optimization time is bounded by

$$O(Nm^2/n) = O(m^2(\log n + \log w_{\max})).$$

□

Applying Lemma 3 instead of Lemma 2, it is not too difficult to obtain the same upper bound for the fitness function w'. The main difference is that a good 1-bit flip has a larger probability than a good 2-bit flip.

Theorem 3. *The expected time until RLS or (1+1) EA working on the fitness function w' constructs a minimum spanning tree is bounded by $O(m^2(\log n + \log w_{\max}))$.*

Proof. By Lemma 4, it is sufficient to analyze the phase after having constructed a connected graph. We apply Lemma 3. The total weight decrease of the chosen 1-bit flips and 2-bit flips is at least $w(s) - w_{\text{opt}}$ if s is the current search point. If the total weight decrease of the 1-bit flips is larger than the total weight decrease of the chosen 2-bit flips, the step is called a 1-step. Otherwise, it is called a 2-step.

If more than half of the steps are 2-steps, we adapt the proof of Theorem 2 with $N' := 2N$ since we guarantee only an expected weight decrease by a factor of $1 - 1/(2n)$. Otherwise, we consider the good 1-steps which have an expected weight decrease by a factor of $1 - 1/(2m')$ for $m' = m - (n-1)$. Choosing $M := \lceil 2 \cdot (\ln 2) \cdot m' \cdot (\log D + 1) \rceil$, we can apply the proof technique of Theorem 2 where M takes the role of N. The probability of performing a good 1-bit flip equals $\Theta(m'/m)$. In this case, we obtain the bound

$$O(Mm/m') = O(m(\log n + \log w_{\max}))$$

for the expected number of steps which is even smaller than the proposed bound.

□

6 Generalizations

Theorems 1, 2, and 3 contain matching upper and lower bounds for RLS and (1+1) EA with respect to the fitness functions w and w'. The bounds are worst-case bounds and one can hope that the algorithms are more efficient for many graphs. Here we discuss what can be gained by other randomized search heuristics.

First, we introduce more problem-specific mutation operators. It is easy to construct spanning trees. Afterwards, it is good to create children with the same number of edges. The new mutation operators are:

- If RLS flips two bits, it chooses randomly a 0-bit and randomly a 1-bit.
- If s contains k 1-bits, (1+1) EA flips each 1-bit with probability $1/k$ and each 0-bit with probability $1/(m-k)$.

For spanning trees, the probability of a specific edge exchange is increased from $\Theta(1/m^2)$ to $\Theta(1/(n(m-n+1)))$. It is easy to obtain the following result.

Theorem 4. *For the modified mutation operator, the bounds of Theorems 1, 2, and 3 can be replaced by bounds of size $\Theta(mn \log n)$ and $O(mn(\log n + \log w_{\max}))$ respectively.*

Using larger populations, we have to pay for improving all members of the population. This holds at least if we guarantee a large diversity in the population. The lower bound of Theorem 1 holds with overwhelming probability. Hence, we do not expect that large populations help. The analysis in the proof of Theorems 2 and 3 is quite precise in most aspects. There is only one essential exception. We know that the weight distance to w_{opt} is decreased on average by a factor of at most $1-1/n$ and we work under the pessimistic assumption that this factor equals $1-1/n$. For large populations or multi-starts the probability of having sometimes much larger improvements may increase for many graphs.

It is more interesting to "parallelize" the algorithms by producing more children in parallel. The well-known algorithm (1+λ) EA produces independently λ children from the single individual from the current population. The selection procedure selects an individual with the smallest w-value (or w'-value) among the parent and its children. In a similar way, we obtain λ-PRLS (parallel RLS) from RLS. In the proofs of Theorem 2 and Theorem 3 we have seen that the probability of a good step is $\Theta(n/m^2)$. Choosing $\lambda = \lceil m^2/n \rceil$, this probability is increased to a positive constant. We have seen that the expected number of good steps is bounded by $O(n(\log n + \log w_{\max}))$. This leads to the following result.

Theorem 5. *The expected number of generations until λ-PRLS or the (1+λ) EA with $\lambda := \lceil m^2/n \rceil$ children constructs a minimum spanning tree is bounded by $O(n(\log n + \log w_{\max}))$. This holds for the fitness functions w and w'.*

If we use the modified mutation operator defined above, the probability of a good step is $O(1/m)$ and we obtain the same bound on the expected number of generations as in Theorem 5 already for $\lambda := m$.

One-point crossover or two-point crossover are not appropriate for edge set representations. It is not possible to build blocks of all edges adjacent to a vertex. For uniform crossover, it is very likely to create graphs which are not spanning trees. Hence, only problem-specific crossover operators seem to be useful. Such operators are described by Raidl and Julstrom (2003). It is difficult to analyze heuristics with these crossover operators.

7 Conclusions

The minimum spanning tree problem is one of the fundamental problems which are efficiently solvable. Several important variants of this problem are difficult, and evolutionary algorithms have a good chance to be competitive on these problems. As a first step toward the analysis of evolutionary algorithms on these problems, randomized local search and simple evolutionary algorithms have been analyzed on the basic minimum spanning tree problem. The asymptotic worst-case (with respect to the problem instance) expected optimization time has been obtained exactly. The analysis is based on the investigation of the expected multiplicative weight decrease (with respect to the difference of the weight of the current graph and the weight of a minimum spanning tree).

References

1. Beyer, H.-G., Schwefel, H.-P., and Wegener, I. (2002). How to analyse evolutionary algorithms. Theoretical Computer Science 287, 101–130.
2. Cormen, T.H., Leiserson, C.E., and Rivest, R.L. (1990). *Introduction to Algorithms.* MIT Press.
3. Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81.
4. Ehrgott, M. (2000). Approximation algorithms for combinatorial multicriteria optimization problems. Int. Transactions in Operational Research 7, 5–31.
5. Giel, O. and Wegener, I. (2003). Evolutionary algorithms and the maximum matching problem. Proc. of 20th STACS. LNCS 2607, 415–426.
6. Hamacher, H.W. and Ruhe, G. (1994). On spanning tree problems with multiple objectives. Annals of Operations Research 52, 209–230.
7. Jerrum, M. and Sorkin, G.B. (1998). The Metropolis algorithm for graph bisection. Discrete Applied Mathematics 82, 155–175.
8. Kano, M. (1987). Maximum and kth maximal spanning trees of a weighted graph. Combinatorica 7, 205–214.
9. Mayr, E.W. and Plaxton, C.G. (1992). On the spanning trees of weighted graphs. Combinatorica 12, 433–447.
10. Papadimitriou, C.H., Schäffer, A.A., and Yannakakis, M. (1990). On the complexity of local search. Proc. of 22nd ACM Symp. on Theory of Computing (STOC), 438–445.
11. Raidl, G.R. and Julstrom, B.A. (2003). Edge sets: an effective evolutionary coding of spanning trees. IEEE Trans. on Evolutionary Computation 7, 225–239.
12. Sasaki, G. and Hajek, B. (1988). The time complexity of maximum matching by simulated annealing. Journal of the ACM 35, 387–403.
13. Scharnow, J., Tinnefeld, K., and Wegener, I. (2002). Fitness landscapes based on sorting and shortest paths problems. Proc. of Parallel Problem Solving from Nature – PPSN VII. LNCS 2939, 54–63.
14. Zhou, G. and Gen, M. (1999). Genetic algorithm approach on multi-criteria minimum spanning tree problem. European Journal of Operational Research 114, 141–152.