Maximum Cardinality Matchings on Trees by Randomized Local Search

Oliver Giel[*]
Fachbereich Informatik, Lehrstuhl 2
Universität Dortmund
44221 Dortmund, Germany
oliver.giel@cs.uni-dortmund.de

Ingo Wegener[*]
Fachbereich Informatik, Lehrstuhl 2
Universität Dortmund
44221 Dortmund, Germany
ingo.wegener@cs.uni-dortmund.de

ABSTRACT

To understand the working principles of randomized search heuristics like evolutionary algorithms they are analyzed on optimization problems whose structure is well-studied. The idea is to investigate when it is possible to simulate clever optimization techniques for combinatorial optimization problems by random search. The maximum matching problem is well suited for this approach since long augmenting paths do not allow immediate improvements by local changes. It is known that randomized search heuristics like simulated annealing, the Metropolis algorithm, the (1+1) EA and randomized local search efficiently approximate maximum matchings for any graph; however, there are graphs where they fail to find maximum matchings in polynomial time. In this paper, we examine randomized local search (RLS) for graphs whose structure is simple. We show that RLS finds maximum matchings on trees in expected polynomial time.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; G.3 [**Probability and Statistics**]: Probabilistic Algorithms

General Terms

Algorithms, Theory

Keywords

Evolutionary algorithms, randomized local search, maximum cardinality matchings, runtime analysis

[*]supported by the German Research Foundation (DFG) as a part of the collaborative research center "Computational Intelligence" (SFB 531)

1. INTRODUCTION

To understand how randomized search heuristics (RSHs) work we analyze their runtime. In general, we are interested in the question for what problems simple RSHs including the Metropolis algorithm (MA), simulated annealing (SA), the (1+1) EA, and randomized local search (RLS) are suitable. One approach is to analyze RSHs for optimization problems whose structure is well-known. When a certain problem turns out to be too difficult for a specific RSH, we ask for what instances of the problem the RSH is successful.

This approach to understand the working principles of RSHs was started for MA and SA for the problems *graph bisection* [8], *clique* [7], and also *maximum matching* [12, 11]. The question whether the right cooling schedule for SA is superior to all choices of a constant temperature in MA for a quite natural problem was answered in the affirmative [15]. The proof uses an instance of the combinatorial optimization problem *minimum spanning tree*. A number of combinatorial optimization problems have already been investigated in the field of evolutionary algorithms. Among them are *sorting* and *single source shortest path* [13], *maximum matching* [4], *minimum spanning tree* [10], and also the NP-hard problem *partition* [16].

The maximum matching problem is well-suited for our purpose. The problem is not trivial. Only theoretical insight has led to polynomial time combinatorial algorithms. Exact optimization is difficult for simple RSHs because small changes of a good solution do not produce better solutions in typical situations. However, even very simple RSHs find approximate solutions efficiently. But there are instances which require an exponential runtime for an optimal solution. Section 2 briefly summarizes known results regarding the maximum matching problem and RSHs. For a better understanding, we first introduce the matching problem and the heuristic RLS.

A *matching* M is a subset of the edges of an undirected graph G such that no edges of M share a node. The *maximum matching problem* is to find a matching of maximal cardinality for a given graph G. Edges not belonging to M are called *free* and nodes not belonging to an edge in M are also called *free*. If M is a matching, an M*-augmenting path* is a simple path in G connecting two free nodes such that matching edges and free edges alternate along the path. Exchanging the state of matching edges and free edges on an augmenting path improves the matching by 1 edge. By Berge's theorem [1], every non-maximum matching has an augmenting path. Consequently, maximum matchings can

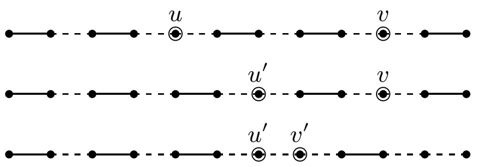


Figure 1: Flipping two bits may shorten or lengthen an augmenting path by two edges.

be constructed by iterated improvements using augmenting paths. However, searching for augmenting paths efficiently is not trivial. Efficient algorithms like the blossom algorithm [3] are clever and their correctness proofs are difficult. The best known algorithms [9, 14] are quite complicated.

We work with a fitness function $f\colon \{0,1\}^m \to \mathbb{Z}$ for graphs with m edges. A search point $s \in \{0,1\}^m$ is interpreted as the characteristic vector of the set of chosen edges. The idea is that the fitness equals the matching size so that the aim is to maximize f. Hence, f equals the number of chosen edges $s_1 + \cdots + s_m$ if s describes a matching.

There are two options to handle non-matchings. One is to start the search with the empty set and never accept non-matchings. This approach is usually chosen for SA and MA. The option we choose here is to start the search with an arbitrary search point and to define a penalty for non-matchings. The penalty directs the search towards matchings. Let $d(v,s)$ be the degree of a vertex v with respect to the edges chosen by s. The penalty $p(v,s)$ assigned to each vertex v equals $r \cdot \max\{0, d(v,s) - 1\}$ for some fixed number $r \geq m+1$. Thus, all vertex penalties are 0 for matchings. Now, $f(s)$ equals the number of chosen edges minus the sum of all vertex penalties, i. e.,

$$f(s) \;:=\; s_1 + \cdots + s_m - \sum_v p(v,s). \qquad (*)$$

The scaling factor $r \geq m+1$ ensures that the fitness of non-matchings is strictly worse than the fitness of all matchings.

The optima of f are maximum matchings and the local optima are maximal matchings. A *maximal matching* is a matching without augmenting paths of length 1. Here, we call such an augmenting path, consisting of only one free edge, a *selectable edge*. If s represents a matching without selectable edges, flipping only one bit of s either produces a new search point representing a smaller matching or a search point representing a non-matching. Hence, randomized search heuristics with an elitist selection strategy need to flip at least two bits to overcome local optima. Indeed, it suffices to flip one or two bits in each step to reach a globally optimal solution. To see this, consider an augmenting path P between free nodes u and v (Figure 1 top, dashed lines indicate free edges). If the first two edges or the last two edges of P flip, P shrinks by two edges. If this happens over and over again, P finally consists of only one selectable edge $\{u', v'\}$ (Figure 1 bottom). Now, flipping only the bit of $\{u', v'\}$ improves the matching.

Motivated by these observations, we consider the following randomized local search strategy RLS.

1. Choose $s \in \{0,1\}^m$ according to some probability distribution.

2. Repeat:

   Decide by a coin flip whether s' is to be produced by a 1-bit flip or 2-bit flip from s. Flip a uniformly chosen bit of s or a uniformly chosen pair of bits accordingly. If $f(s') \geq f(s_{t-1})$, set $s := s'$.

The aim is to analyze the number of fitness evaluations until RLS produces a search point s such that $f(s)$ is maximal. This random number is called *optimization time* and we are interested in its expected value, the expected optimization time. We assume that the coin flip in the mutation step is fair; however, this is not essential. It is only required that the probabilities of both outcomes of the coin flip are $\Omega(1)$. Then, RLS is essentially the (1+1) EA restricted to 1- and 2-bit flips. Indeed, the probability that a certain bit or a certain pair of bits flip is $\Theta(1/m)$ resp. $\Theta(1/m^2)$ for both the (1+1) EA and RLS. We believe that the results presented in this paper also apply to the (1+1) EA but it is not obvious that the proofs can be adapted. For an experimental study see [2].

Next we summarize related results regarding the maximum matching problem and RSHs. Then we study the effect of 1- and 2-bit flips in Section 3, which provides some basic lemmas. In Section 4, we consider complete trees and obtain a time bound of $O(m^{7/2})$ for RLS. For arbitrary trees, our bound of $O(D^2 m^4)$ in Section 5 depends also on the diameter D of the tree.

2. RELATED WORK

Sasaki and Hajek [12] show that simulated annealing (SA) and the Metropolis algorithm (MA) efficiently find approximations for maximum matchings. These randomized search heuristics find a $(1+\varepsilon)$-optimal solution (i. e., a matching such that maximum matchings are at most by a factor of $(1+\varepsilon)$ larger) in expected polynomial time where the degree of the polynomial depends on $\varepsilon > 0$. We prove the same for the (1+1) EA and RLS in [4]. The idea is that matchings which are at most $(1+\varepsilon)$-optimal include an augmenting path whose length is bounded by $2\lceil 1/\varepsilon \rceil - 1$ (compare [6]). The probability to flip the path's edges in the desired order is then large enough.

Remarkably, SA, MA, and the (1+1) EA utilize no knowledge about the maximum matching problem; RLS utilizes no knowledge except that 1- and 2-bit flips are sufficient. They are general search heuristics, i. e., they are not designed to optimize a particular class of fitness functions, and work in the black box scenario. This includes that access to the graph instance is limited to the fitness function, i. e., the graph is assumed to be hidden to the heuristics. He and Yao [5] come up with a similar approximation result for a population based EA; however, their EA needs to access the graph instance and the variation operators seem to be tailored to the maximum matching problem to some extent.

Although all mentioned heuristics can be viewed as polynomial time approximation schemes for the maximum matching problem, they may fail when it comes to exact optimization. Sasaki and Hajek [12] prove an exponential expected optimization time for MA and SA and a particular graph class when the search starts with the empty matching. Sasaki [11] proves with another technique that there exist starting points with an exponential expected optimization time for these graphs. In [4], we prove an exponential expected optimization time for RLS and the (1+1) EA for any starting point (except the optimum). In fact, it can be shown that many starting points (including the empty matching) lead to an exponential optimization time even with an overwhelming probability. For these starting points, multi-start strategies of RLS and the (1+1) EA will also be unsuccessful.

In spite of carefully chosen worst-case examples, we believe that simple randomized search heuristics including the (1+1) EA and RLS are able to find maximum matchings for simple graphs efficiently. In [4], we present the first positive results on the optimization time of evolutionary algorithms for a class of graphs, namely path graphs. *Path graphs* are trees consisting of m edges laid end to end such that the edges form a path of length m. For every starting point, the expected optimization time of the (1+1) EA and RLS is $O(m^4)$. One can also prove that some second-best matchings, i.e., matchings which are by one edge smaller than maximum matchings, are starting points with an expected optimization time of $\Omega(m^4)$. Hence, the final optimization step from second-best to maximum matchings is essential.

Our intuition is that simple randomized search heuristics are successful if the number of potential augmenting paths is polynomially bounded in most situations during the optimization process. In a tree with n nodes and $m = n-1$ edges, there is one and only one (simple) path between any two nodes. Hence, there are at most $\binom{n}{2} = O(m^2)$ potential augmenting paths at any time.

3. PRELIMINARIES

First we show that RLS finds some matching quickly.

Lemma 1. *For any initial search point and any graph, RLS finds a matching in an expected time of $O(m \log m)$.*

Proof. Let $p = r \cdot k$ be the sum of the vertex penalties for the search point s (see equation $(*)$ in Section 1). As $r \geq m+1$, the fitness function rewards any decrease of p and penalizes any increase of p. The sum of all vertex degrees is $2m$. Hence, k is less than $2m$ and there are at least $\lceil k/2 \rceil \leq m$ edges chosen by s whose elimination decreases k. The probability of a specific 1-bit flip equals $\Theta(1/m)$. Hence, the expected waiting time to decrease k is bounded by $O(m/k)$. Summing up for $1 \leq k < 2m$ yields the claim. $\square$

In the analyses, we distinguish situations with and without selectable edges. The next lemma considers a situation with a selectable edge and states that the next step changing the situation will likely improve the matching.

Lemma 2. *Given an arbitrary tree and a matching with at least one selectable edge. In expected time $O(m)$, RLS has improved the matching or there is no selectable edge but an augmenting path of length 3. The matching is improved with a probability of at least $1/2$.*

Proof. We prove the lemma by proving the following two claims for all search points describing matchings with a selectable edge: In the next step,

– the probability of improving the matching is at least $1/(2m)$, and

– the probability of destroying all selectable edges without improving the matching is at most $1/(2m)$.

The first claim is obvious since $1/(2m)$ is the probability of flipping exactly a specified selectable edge. Now we prove the second claim. A step flipping only matching edges is not accepted. A step flipping only free edges is not accepted or it improves the matching. Hence, we only have to consider mixed 2-bit flips choosing a free edge e and a matching edge e'. Since e' must not become selectable, e has to be adjacent to e' implying that e is not selectable. To destroy a given selectable edge e^*, e must also be adjacent to e^*. In summary, the free edge e is adjacent to the selectable edge e^* at its free endpoint, and at the other endpoint, e is adjacent to the matching edge e'. This shows that an augmenting path of length 3 is created if e and e' flip. By the matching property, the choice of e determines e', and, in a tree, e' determines e between e' and the given edge e^*. Hence, the possible pairs $\{e, e'\}$ are pairwise disjoint implying that their number is bounded above by $(m-1)/2$. The probability to flip any of these pairs is at most $((m-1)/2) \cdot (1/2) \cdot \binom{m}{2}^{-1} = 1/(2m)$. $\square$

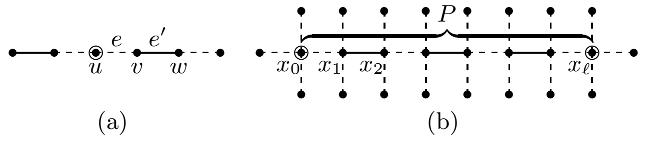


Figure 2: (a) In an accepted step, a free edge e incident upon a free node u and a matching edge e' adjacent to e flip. (b) The augmenting path P and its neighborhood. The figure only shows P and nodes and edges adjacent to P.

Hence, on average, we have to create a selectable edge at most twice before the matching is improved. To bound the expected time to improve the matching it now suffices to bound the expected time to create a selectable edge. To this end, we study a situation without selectable edges and pick an augmenting path. The aim is to bound the time until the selected path finally consists of only one selectable edge. First, we consider only one step, i.e., one iteration of the loop of RLS. Obviously, 1-bit flips will not be accepted. The next lemma describes the effect of 2-bit flips on an augmenting path and claims that it is not too unlikely that the path gets shorter.

Lemma 3. *Let $P = (x_0, x_1, \ldots, x_\ell)$ be an augmenting path in a tree with respect to a matching without a selectable edge. An accepted mutation step of RLS preserves the current matching size and either*

– *leaves P unchanged,*

– *lengthens P by two edges at one end, or*

– *shortens P by a multiple of two edges at one end.*

If P is changed, the probability that P shrinks is at least $2/(\deg(x_0) + \deg(x_\ell))$.

Proof. Because no edge is selectable, augmenting paths have at least three edges. Hence, only steps preserving the matching size are accepted. This implies that only 2-bit flips choosing a free edge e and a matching edge e' are accepted. Since e is free but not selectable, we obtain a new matching only if e' is adjacent to e, i.e., $e = \{u, v\}$ and $e' = \{v, w\}$, and u is free (see Figure 2(a)).

We investigate accepted steps changing P. There are four possibilities for the free edge $e = \{u, v\}$ (see Figure 2(b)):

– e is an "outer" edge of P, namely, $e = \{x_0, x_1\}$ or $e = \{x_{\ell-1}, x_\ell\}$,

– e is an "inner" edge of P,

– e does not belong to P but its free endpoint u does,

– neither e nor its free endpoint u belong to P.

In the first case, e' is adjacent to e and lies on P. Flipping e and e' shortens P by two edges. The second case is impossible because no endpoint of e would be free. In the third case, either $u = x_0$ or $u = x_\ell$. Flipping e and e' lengthens P. Since w becomes a free node, the length increases by exactly 2. In the last case, P is only changed if the matching edge e' connects two inner nodes of P. (To see this, observe that each edge incident upon a node of P is either free or it is a matching edge belonging to P.) The node w is then an inner node of P and becomes a free node. Either $(x_0, \ldots, w)$ or $(w, \ldots, x_\ell)$ becomes an augmenting path. Since every augmenting path has odd length, the new path is by an even number of edges shorter than the old path P.

We have seen that the length of P can only increase if $u = x_0$ or $u = x_\ell$. In such a step, the flipping free edge e incident upon x_0 or x_ℓ determines which matching edge can flip. Therefore, there are at most $\deg(x_0) + \deg(x_\ell) - 2$ 2-bit flips increasing the length of P and at least 2 2-bit flips decreasing the length of P. This proves the claimed probability. $\square$

In many situations, we will consider the number R of *relevant steps* rather than the total number of steps T. The definition of a relevant step will depend on the situation. If an expected number of $E(R)$ relevant steps is necessary to reach some target and in any situation the next step is relevant with a probability of at least p then the expected total number of steps $E(T)$ is at most $p^{-1} \cdot E(R)$.

4. COMPLETE *k*-ARY TREES

We investigate complete k-ary trees, i.e., rooted trees where inner nodes have k successors and all leaves have the same distance to the root. (Paths graphs can be considered as complete unary trees where $k = 1$.) For $k \geq 2$, the diameter D of path graphs is $\Theta(\log_k m)$ and at most $\log_k m$.

Considering path graphs and 2-bit flip again, there is one possibility to shorten an augmenting path at each end point and one possibility to lengthen it at each end point in a typical situation (see Figure 1 top). Thus, the probabilities of lengthening and shortening the path are the same; they describe a fair game. The game gets unfair for k-ary trees and $k \geq 2$. There are k possibilities to lengthen an augmenting path if the end point is an inner node. Still there is only one possibility to shorten it. On the one hand, 2-bit flips relevant to a specific augmenting path now tend to lengthen the path. One might expect that this leads to a larger optimization time. On the other hand, the length of all paths is now bounded logarithmically. The next lemma bounds the length of the shortest augmenting path with respect to a matching M. This gives rise to an upper bound which is in fact smaller than the lower bound for path graphs.

LEMMA 4. *Given a complete k-ary tree with m edges and $k \geq 2$. If M^* is a maximum matching and M a matching with $m^* \geq 1$ edges less than M^*, there is an M-augmenting path whose length is strictly less than $L := 2\log_k(2m/m^*)$.*

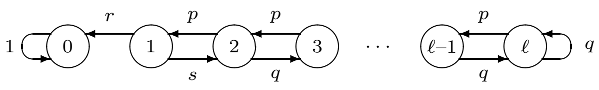


Figure 3: The Markov chain M in Lemma 5.

PROOF. The nodes of the tree in distance h from the root belong to level h. Let d denote the depth of the tree implying $d \leq \log_k m$. Considering the symmetric difference $M^* \oplus M$, we obtain a set of m^* node-disjoint M-augmenting paths (see Theorem 1 in [6]). Let us assume that the length of a shortest augmenting path in the set is ℓ. A simple path in a tree can contain at most two nodes from each level. This implies that each augmenting path in the set has an odd length of at least $\ell < 2d$ and contains at least one node on a level $d' \leq d - \lceil \ell/2 \rceil$. Hence, m^* is bounded above by the number of nodes on the levels $0, \ldots, d - \lceil \ell/2 \rceil$ implying that

$$\begin{aligned} m^* \; &\leq \; k^0 + \cdots + k^{d - \lceil \ell/2 \rceil} \; = \; \frac{k^{d-\lceil \ell/2 \rceil + 1} - 1}{k-1} \\ &< \; \frac{k}{k-1} k^{d - \lceil \ell/2 \rceil} \; \leq \; 2mk^{-\lceil \ell/2 \rceil}. \end{aligned}$$

Solving for ℓ yields the proposed bound $\ell < L$. $\square$

The next lemma gives the hitting time of a random walk that is essential in all our analyses. The random walk is similar to the random walk describing the gambler's ruin problem but has a reflecting barrier.

LEMMA 5. *Given the homogeneous Markov chain M with state space $S = \{0, \ldots, \ell\}$, initial state $\ell \geq 2$, and positive transition probabilities $p(0,0) = 1$, $p(1,0) = r$, $p(1,2) = s$, $p(i, i-1) = p$ for $i \in \{2, \ldots, \ell\}$, $p(i, i+1) = q$ for $i \in \{2, \ldots, \ell - 1\}$, $p(\ell, \ell) = q$, where $0 < p, r < 1$, $q = 1 - p$, and $s = 1 - r$ (see Figure 3). The expected time to reach state 0 for the first time starting in state ℓ is*

$$h_{\ell,0} = \begin{cases} \ell^2 + \left(\frac{2}{r} - 3\right)\ell - \frac{1}{r} + 2 & \text{if } p = q = 1/2, \\ \frac{1}{q-p}\left(\frac{\left(\frac{q}{p}\right)^\ell - 1}{\frac{q}{p} - 1} + \frac{s}{r}\left(\frac{q}{p}\right)^{\ell-1} - \ell - \frac{s}{r}\right) + \frac{1}{r} & \text{else.} \end{cases}$$

In particular, for $p = q = r = s = 1/2$, $h_{\ell,0} = \ell^2 + \ell$.

PROOF. We claim that

$$h_{1,0} \; = \; \frac{1}{p} + \frac{q}{p} h_{2,1}$$

and, for $j \geq 2$,

$$h_{j,j-1} \; = \; \begin{cases} 2(\ell - j + 1) & \text{if } p = q = 1/2, \\ \dfrac{\left(\frac{q}{p}\right)^{\ell - j + 1} - 1}{q - p} & \text{if } p \neq q. \end{cases}$$

Summing up the terms $h_{j,j-1}$ for $j \in \{1, \ldots, \ell\}$ yields the lemma. By the law of total probability, $h_{1,0} = 1 + p \cdot 0 + q \cdot (h_{2,1} + h_{1,0})$. This implies the first part of the claim. We prove the second part by induction on j. In accordance with the claim (for $p = q = 1/2$ and for $p \neq q$), $h_{\ell,\ell-1} = 1/p$ since the transition $(\ell, \ell - 1)$ is the only transition leaving state ℓ with positive probability p. For $\ell - 1 \geq j \geq 2$, $h_{j,j-1} = 1 + p \cdot 0 + q(h_{j+1,j} + h_{j,j-1})$ implying that

$$h_{j,j-1} \; = \; \frac{1}{p}(1 + q \cdot h_{j+1,j}).$$

We apply the induction hypothesis and consider the cases $p = q = 1/2$ and $p \neq q$ separately. In the first case, we obtain

$$h_{j,j-1} \;=\; \frac{1}{p}\Big(1 + q\big(2(\ell - (j+1) + 1)\big)\Big) \;=\; 2(\ell - j + 1),$$

and in the second case,

$$\begin{aligned} h_{j,j-1} \;&=\; \frac{1}{p}\left(1 + q\frac{\left(\frac{q}{p}\right)^{\ell-j} - 1}{q-p}\right) \\ &=\; \frac{\frac{q}{p} - 1}{p(\frac{q}{p} - 1)} + \frac{\left(\frac{q}{p}\right)^{\ell-j+1} - \frac{q}{p}}{q-p} \;=\; \frac{\left(\frac{q}{p}\right)^{\ell-j+1} - 1}{q-p}. \end{aligned}$$

This proves the claim and finishes the proof. □

Now we are prepared to analyze RLS on totally balanced trees.

THEOREM 1. *For any choice of the first search point, the expected time until RLS finds a maximum matching on a complete k-ary tree, $k \geq 2$, is bounded by $O(m^{7/2})$.*

PROOF. By Lemma 1, the expected time to find a matching is small enough. Afterwards, we estimate the expected number of relevant steps to improve the matching. In each step, we choose some shortest augmenting path P. Now, in a situation without selectable edges, a step is called relevant if it is accepted and changes P. In a situation with at least one selectable edge, any step improving the matching or destroying all selectable edges is relevant. The expected number of all steps is then only by a factor of $O(m^2)$ larger than the expected number of relevant steps because the probability of a relevant step is $\Omega(1/m^2)$ in either situation.

Assume we can guarantee that there is always some augmenting path of length at most ℓ. In situations with a selectable edge, we apply Lemma 2. In a step changing the situation, the probability to improve the matching is at least $r = 1/2$, and the probability to create a path of length three is at most $1/2$. In situations without a selectable edge, we pessimistically replace shortenings of the considered path P with shortenings by only two edges. By Lemma 3, this leads to a probability of at least $p = 1/(k+1)$ of shortening this path in relevant steps. We can now represent the current length j of the path by the state $\lceil j/2 \rceil$ of the Markov chain in Lemma 5 with state space $\{0, \ldots, \lceil \ell/2 \rceil\}$. If we pessimistically start with a path of length ℓ, the expected number of relevant steps until the matching is improved (state 0 is reached) is at most

$$\frac{k+1}{k-1}\Big(\frac{k^{\lceil \ell/2 \rceil} - 1}{k-1} + k^{\lceil \ell/2 \rceil - 1} - (\lceil \ell/2 \rceil + 1)\Big) + 2 \;=\; O(k^{\ell/2}).$$

Until $m^* \leq 2m^{1/2}$, Lemma 4 guarantees that there is some augmenting path P of length at most $\ell < \log_k m$. This leads to an expected number of $O(m^{1/2})$ relevant steps for an improvement. Since $|M^*| \leq m$, we apply this bound only $O(m)$ times implying that the expected number of relevant steps until $m^* \leq 2m^{1/2}$ is $O(m^{3/2})$. Afterwards, we use the bound $\ell \leq 2\log_k m$ which is a trivial bound on the diameter D of the tree. Then we obtain an upper bound of $O(m)$ on the number of relevant steps to improve the matching. Since we apply the last bound only $\lfloor 2m^{1/2} \rfloor$ times, this leads to an additional expected number of $O(m^{3/2})$ relevant steps. This proves the theorem as argued at the beginning of this proof. □

5. GENERAL TREES

Path graphs have turned out to be the hardest k-ary trees for RLS. The upper bound of $O(m^4)$ for path graphs in [4] cannot be improved if the search starts with an arbitrary search point (see Section 2). Indeed, we conjecture that path graphs are the hardest trees and the best possible bound for RLS on trees is $O(m^4)$. Here, we prove a weaker upper bound of $O(D^2m^4)$ on the expected optimization time where D denotes the diameter of the tree.

We investigate an arbitrary search point s describing a non-maximum matching. We are interested in the expected time to obtain a search point describing a matching of the same size with a selectable edge. We choose an arbitrary augmenting path P connecting u and v. The endpoints u and v of P move around. A step is called *u-relevant,* if the endpoint u moves.

LEMMA 6. *Given a tree and a non-maximum matching without selectable edges. Let u denote an endpoint of an arbitrary augmenting path. RLS creates a selectable edge in an expected number of $O(D^2m)$ u-relevant steps.*

The proposed bound on the expected optimization time for trees follows from Lemma 6 and our results in Section 3.

THEOREM 2. *For any choice of the first search point, the expected time until RLS finds a maximum matching in a tree is bounded by $O(D^2m^4)$.*

PROOF. The expected time to create a matching is small enough (Lemma 1). Once we have a matching, it takes an expected number of $O(D^2m)$ u-relevant steps to produce a selectable edge (Lemma 6) and each step is u-relevant with a probability of $\Omega(1/m^2)$. Hence, the expected number of steps to produce a selectable edge is $O(D^2m^3)$. When a selectable edge exists, a decision is made within an expected number of $O(m)$ steps: With a probability of at least $1/2$ the matching is improved (Lemma 2). Using Markov's inequality we obtain that some $O(D^2m^3)$ steps improve the matching with a probability of $\Omega(1)$; otherwise, we start over and wait for a new selectable edge. Hence, an expected number of $O(D^2m^3)$ steps improve the matching. For a maximum matching, $O(m)$ improvements are sufficient. This leads to the claimed $O(D^2m^4)$ bound. □

It remains to prove Lemma 6. We focus on the distance between the endpoints u and v. If this distance is only 1, the edge $\{u, v\}$ is selectable but we may be lucky and produce a selectable edge somewhere else earlier. To obtain precise statements we fix some node w as the root of the tree. For a fixed root w, $T(r)$ is the subtree rooted at node r and $|T(r)| = O(m)$ is its number of nodes.

CLAIM 1. *Assume there is no selectable edge, $r \neq w$ is the initial position of u, and $v \notin T(r)$. The expected number of u-relevant steps until one of the following three events happens is bounded by $|T(r)|$: u leaves $T(r)$, v enters $T(r)$, a selectable edge is created.*

PROOF. It suffices to bound the number of u-relevant steps until the first event "u leaves $T(r)$" happens assuming that the other two events do not happen. The proof investigates the random walk of u influenced by steps where an edge adjacent to u flips. Other steps moving u are only advantageous to us since u moves closer to r or leaves $T(r)$

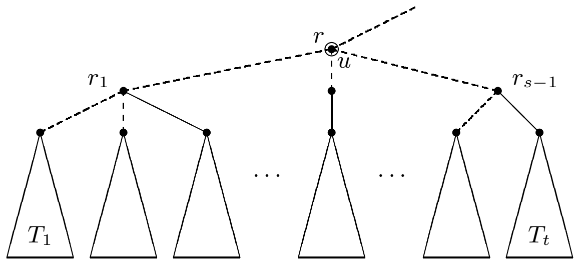


Figure 4: The subtree $T(r)$. Initially, the endpoint u is at the root of $T(r)$.

(see Lemma 3). The proof implicitly uses the argument that a large degree of u implies that not many subtrees of $T(u)$ can be large. In the considered situation, the conditions of Lemma 3 hold and we may pessimistically assume that the path is never shortened by more than two edges in u-relevant steps. It is important to observe that u can only visit nodes in $T(r)$ with an even distance to its initial position, the root r of $T(r)$. The proof is by induction on $|T(r)|$. If $|T(r)| = 1$, then $T(r)$ is a leaf and the expected number of u-relevant steps is at most 1. Now, let $|T(r)| > 1$ and let s denote the degree of the root r. Let $T_1, \dots, T_t$ denote those subtrees of $T(r)$ such that there are two edges between r and the root of each T_i (Figure 4). Whenever u is at r, r is connected to $s-1$ nodes $r_1, \dots, r_{s-1}$ in $T(r)$ and to its parent by free edges. Each node r_j may be adjacent to several roots of the subtrees $T_1, \dots, T_t$, but each r_j is connected to at most one of these roots by a matching edge. Hence, at most $s-1$ roots of subtrees T_i are reachable for u. When u leaves a subtree T_i, it can only return to r or some node on the path outside $T(r)$. W.l.o.g. let $T_1, \dots, T_{s-1}$ be $s-1$ largest subtrees in $\{T_1, \dots, T_t\}$. For a subtree T of $T(r)$ and u starting at the root of T, let $E(T)$ denote the expected number of u-relevant steps until u leaves T (or a selectable edge is created). By the law of total probability,

$$E(T(r)) \le \frac{1}{s} \cdot 1 + \frac{1}{s}\bigl(1 + E(T_1) + E(T(r))\bigr) + \dots + \frac{1}{s}\bigl(1 + E(T_{s-1}) + E(T(r))\bigr).$$

Solving for $E(T(r))$ and using the induction hypothesis yields $E(T(r)) \le s + |T_1| + \dots + |T_{s-1}|$. Because $T(r)$ includes all subtrees T_i, the nodes $r_1, \dots, r_{s-1}$, and r, the right-hand side of the last inequality is at most $|T(r)|$. □

Claim 2. *Assume there is no selectable edge, r is the initial position of u, and $v \in T(r)$. The expected number of u-relevant steps until one of the following three events occurs is bounded by $O(|T(r)|)$: u leaves $T(r)$ and v remains in $T(r)$, u and v are in the same proper subtree of $T(r)$, a selectable edge is created.*

Proof. It is essential that the distance between u and r remains even. (Then the distance between v and r is odd.) In the following, T_u and T_v denote subtrees of $T(r)$ rooted by a node on the level below r. If u is in a proper subtree of $T(r)$, T_u denotes the subtree of $T(r)$ currently containing u; analogously for v (see Figure 5). First, we show that v cannot leave $T(r)$ in the time we consider. If u is at r, then v cannot leave $T(r)$. If u leaves $T(r)$, the second stopping criterion is fulfilled. If u moves to a node in T_v (Figure 5(a)),

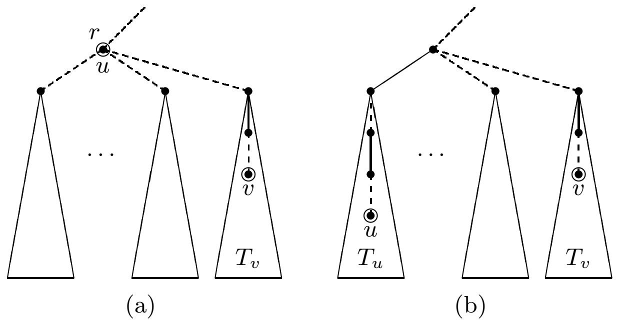


Figure 5: The subtree $T(r)$. (a) A special situation and (b) the situation when u has moved into a subtree of $T(r)$.

the second stopping criterion is fulfilled. The endpoint v might only leave $T(r)$ if u first moves to a subtree $T_u \neq T_v$ (Figure 5(b)). If this is the case, the path P between u and v visits r. By Lemma 3, relevant steps can increase and decrease the length of P only by a multiple of two edges at either end. Since u initially had an even distance to r, namely 0, u keeps an even distance and v keeps an odd distance to r. Now, if v leaves its subtree T_v in a shortening step, then v moves to some node on the path P. Since v cannot visit r (odd distance), the new location of v could only be in T_u and the last stopping criterion would be fulfilled first. Hence, v cannot leave $T(r)$ before one of the stopping criteria is fulfilled.

Now it is clear that we can upper bound the expected number of u-relevant steps until one of the stopping criteria is fulfilled by the expected number of u-relevant steps until one of the following events happens: Given that v stays in its subtree T_v either u leaves $T(r)$ or u moves into T_v . This can be done in the same way as in the proof of Claim 1. The only difference is that the terms for subtrees contained in T_v can now be bounded by 0. This can only decrease the upper bound $O(|T(r)|)$ on the expected number of u-relevant steps. Note that the case $r = w$ where u cannot leave $T(r)$ is included. □

In the following let r be the node where the path from u to the root w of the tree and the path from v to w meet, i.e., r is the root of the smallest subtree containing both u and v. Let $d(r)$ be the depth of r, i.e., the distance between r and the root w. We consider the random variable $d(r)$ with respect to time. A selectable edge is created if the distance between u and v equals 1. This is certainly the case if r is the root of a subtree with only two levels. This happens if $d(r) \ge D - 1$. The idea is to prove that $d(r)$ tends to grow.

The situation depicted in Figure 5(a) is of particular interest in our analysis. There is no selectable edge, one endpoint of P is located at the root r of $T(r)$, and the other endpoint is contained in $T(r)$. We call this a *special situation.* W.l.o.g. the endpoint at r is named u and the other v.

Claim 3. *In a special situation it is possible to define disjoint events $E_1, \dots, E_4$ such that*

- *E_1 is the event that $d(r)$ decreases (and it can decrease by at most 2).*
- *E_2 implies that $d(r)$ increases by at least 2,*

– E_3 *implies that* $d(r)$ *increases by at least 1, and*

– E_4 *is the event that a selectable edge is created.*

The expected number of u*-relevant steps for one of these events to happen is* $O(m)$*. The probability that* E_1 *happens first is less than or equal to the probability that* E_2 *happens first. When one of the events* $E_1, \ldots, E_3$ *has happened, the expected number of* u*-relevant steps until a special situation is reached again (or* E_4 *happens) is* $O(m)$*. In this time,* $d(r)$ *cannot decrease.*

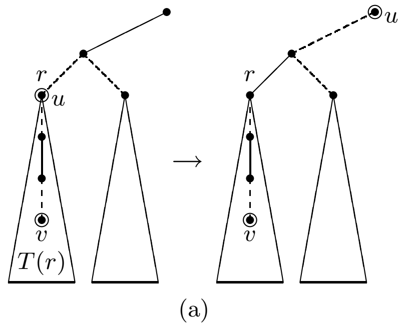
Proof. The initial situation is assumed to be a special situation. Hence, the path P connecting u and v has a minium length of 3 edges and $T(r)$ includes at least 4 levels. We consider only situations without selectable edges. Otherwise, E_4 has happened and we can stop. Furthermore, u (initially located at r) keeps an even distance to r until a new special situation is reached.

By Claim 2, after an expected number of $O(m)$ u-relevant steps, both endpoints are in the same proper subtree of $T(r)$, or u has left $T(r)$. First assume that v finishes such a phase, i. e., v leaves T_v (see Figure 5(b) and the proof of Claim 2 for the definition of T_u and T_v). From the proof of Claim 2, we also know that v cannot reach r (odd distance) but a node in T_u on the path P. If this happens (event E_3), the depth $d(r)$ increases by at least 1 and the new situation is a special situation.

Now assume that u finishes the phase and consider the last step. If the length of P decreases by more than 2 in the last step then u enters T_v and reaches a node on the path P (event E_2'). The depth $d(r)$ increases by at least 2 and a special situation is reached. Otherwise, u visits r before the last move. Given that u finishes the phase in the next step, it either moves to T_v with a probability of at least 1/2 or leaves $T(r)$ with a probability of at most 1/2 (Figure 5(a)). (If $r = w$, event E_1 is not possible.) In the first case (event E_2''), $d(r)$ increases by 2 and a special situation is reached. In the second case (event E_1), $d(r)$ decreases by at most 2: The endpoint u may move to its pre-predecessor in the tree (Figure 6(a)). Then a special situation is reached. Another option is that u moves to some sibling of r, say r' (Figure 6(b)). Then the new situation is *not* a special situation. In the following, we again assume that no selectable edge is created. By Claim 1, after an additional waiting time of an expected number of $O(|T(r')|) = O(m)$ u-relevant steps, u has left $T(r')$ and is back at r (or at some node on the path inside $T(r)$) such that a special situation is reached, or v is also in $T(r')$. In the last case, v must have visited the parent of r and r' (or a node on the path inside $T(r')$) first such that a special situation has been reached. Hence, there is no additional decrease of the depth $d(r)$ in the additional waiting time for a new special situation.

The events E_2'' and E_1 can only occur in the same situation (where u finishes the phase and u visits r before the last move). Now, given that one of these two events happens in the next step, $\text{Prob}(E_2'') \geq 1/2$. Hence, the probability that the event $E_2 := E_2' \cup E_2''$ finishes the phase is only larger than the probability that event E_1 finishes the phase. Moreover, we have shown that after an event E_1, E_2, or E_3 has happened, a special situation is reached within an expected number of $O(m)$ u-relevant steps where $d(r)$ is not decreased. □

Now we can combine our results and prove Lemma 6.



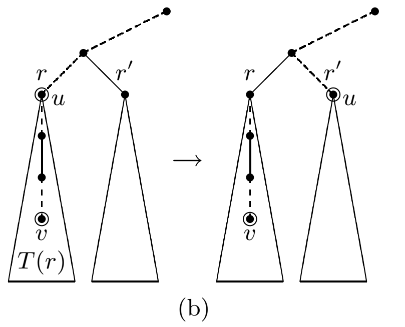


Figure 6: If $d(r)$ decreases, u moves to the pre-predecessor of r (a) or to a sibling r' of r (b).

Proof of Lemma 6. Initially, we choose $w := u$ to be the root of the tree. Then $d(r) = 0$ and we start in a special situation. This implies that $T(r)$ includes at least 4 levels. When a situation is reached where $T(r)$ includes only two levels, the considered path P is a selectable edge since its length must be odd. In a situation where $T(r)$ includes only three levels, the augmenting path is a selectable edge or the the next P-relevant step creates a selectable edge. To see this, observe that the length of P is then 1 or 3. There is nothing to show if the length is 1. If the length is 3, neither u nor v can be at the root of $T(r)$ and the path could be lengthened by at most one edge at u or v. Hence, by Lemma 3, the next P-relevant step can only shorten P. This implies that the expected number of u-relevant steps to create a selectable edge is by at most 1 larger than the expected number of u-relevant steps for the event $d(w) \geq D - 2$. By the $O(m)$ bounds in Claim 3, it is sufficient to prove an upper bound of $O(D^2)$ on the expected number of the events E_1, E_2, and E_3 until the event $d(w) \geq D - 2$ occurs.

We slow down this stochastic process by assuming that E_1 decreases $d(r)$ by exactly 2, E_2 increases $d(r)$ by exactly 2, and $\text{Prob}(E_1 \text{ happens first}) = \text{Prob}(E_2 \text{ happens first})$. We prove that the probability of the event $d(r) \geq D - 2$ in a phase including $\lceil D^2/2 \rceil + 2D$ of the events $E_1, \ldots, E_4$ is at least 1/2. The claim then follows since we may start over and choose a new root w if the phase were unsuccessful. If the number of E_3-events in the phase is at least $D - 2$, the result follows since, with probability 1/2, event E_2 happens at least as often as event E_1. Otherwise, there are at least $D^2/2 + D$ events E_1 and E_2, and we disregard the help of E_3-events. Now the events E_1 and E_2 describe a symmetric random walk as analyzed in Lemma 5 for $p = q = r = s = 1/2$. Starting there in state $\ell = \lceil (D-2)/2 \rceil$ (standing for $d(r) = 0$), the expected number of steps to reach the state 0 is less than $D^2/4 + D/2$. By Markov's inequality, $D^2/2 + D$ steps are successful with a probability of at least $1/2$. $\square$

6. CONCLUSIONS

Randomized search heuristics can be fooled by carefully constructed graphs but RLS finds maximum matchings for simple graphs (trees) in expected polynomial time. The results indicate that randomized search heuristics can use algorithmic ideas not known to the designer of the algorithm. Our results contribute to the understanding of how simple randomized heuristics work. It would be interesting to see how efficient related heuristics like SA and MA perform on trees.

7. REFERENCES

[1] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43:842–844, 1957.

[2] P. Briest, D. Brockhoff, B. Degener, M. Englert, C. Gunia, O. Heering, T. Jansen, M. Leifhelm, K. Plociennik, H. Röglin, A. Schweer, D. Sudholt, S. Tannenbaum, and I. Wegener. Experimental supplements to the theoretical analysis of EAs on problems from combinatorial optimization. In *Proceedings of the 8th Conference on Parallel Problem Solving from Nature (PPSN '04)*, volume 3242 of *Lecture Notes in Computer Science*, pages 21–30. Springer, 2004.

[3] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[4] O. Giel and I. Wegener. Evolutionary algorithms and the maximum matching problem. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS '03)*, volume 2607 of *LNCS*, pages 415–426. Springer, 2003.

[5] J. He and X. Yao. Time complexity analysis of an evolutionary algorithm for finding nearly maximum cardinality matching. *Journal of Computer Science and Technology*, 19(4):450–458, 2004.

[6] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[7] M. Jerrum. Large cliques elude the Metropolis process. *Random Structures and Algorithms*, 3(4):347–359, 1992.

[8] M. Jerrum and G. B. Sorkin. The Metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82:155–175, 1998.

[9] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science (FOCS '80)*, pages 17–27. IEEE, 1980.

[10] F. Neumann and I. Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In *Proceedings of the 6th Genetic and Evolutionary Computation Conference (GECCO '04)*, volume 3102 of *LNCS*, pages 713–724. Springer, 2004.

[11] G. Sasaki. The effect of the density of states on the Metropolis algorithm. *Information Processing Letters*, 37(3):159–163, 1991.

[12] G. H. Sasaki and B. Hajek. The time complexity of maximum matching by simulated annealing. *Journal of the ACM*, 35:387–403, 1988.

[13] J. Scharnow, K. Tinnefeld, and I. Wegener. Fitness landscapes based on sorting and shortest paths problems. In *Proceedings of the 7th Conference on Parallel Problem Solving from Nature (PPSN '02)*, volume 2439 of *LNCS*, pages 54–63. Springer, 2002.

[14] V. V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ maximum matching algorithm. *Combinatorica*, 14:71–109, 1994.

[15] I. Wegener. Simulated annealing beats Metropolis in combinatorial optimization. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP '05)*, volume 3580 of *LNCS*, pages 589–601, 2005.

[16] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS '05)*, volume 3404 of *LNCS*, pages 44–56. Springer, 2005.