

# Speeding up Evolutionary Algorithms by Restricted Mutation Operators

Benjamin Doerr\*, Nils Hebbinghaus<sup>†</sup>    Frank Neumann<sup>‡</sup>

May 16, 2006

## Abstract

We investigate the effect of restricting the mutation operator in evolutionary algorithms with respect to the runtime behavior. Considering the Eulerian cycle problem we present runtime bounds on evolutionary algorithms with a restricted operator that are much smaller than the best upper bounds for the general case. In our analysis it turns out that a plateau which has to be coped with for both algorithms changes its structure in a way that allows the algorithms to obtain an improvement much faster. In addition, we present a lower bound for the general case which shows that the restricted operator speeds up computation by at least a linear factor.

## 1 Introduction

Evolutionary algorithms (EAs) are randomized search heuristics, that have shown to be very successful in solving problems from combinatorial optimization as well as producing good solutions in real world applications. Especially in the case of real world applications often little is known about the structure of the problem and a standard evolutionary approach achieves good results.

In the case of combinatorial optimization problems frequently much more is known about the structure of a considered problem and EAs that are designed for the considered problem achieve better results than a standard

---

\*Max-Planck-Institut für Informatik, Saarbrücken, Germany.

<sup>†</sup>Max-Planck-Institut für Informatik, Saarbrücken, Germany.

<sup>‡</sup>Institut für Informatik und Praktische Mathematik, Kiel, Germany.

evolutionary approach. Raidl et al. [9] have recently shown that NP-hard spanning tree problems can be solved much easier by using a mutation operator that chooses edges with a small weight much more often for inclusion in a mutation step than heavier edges. They have shown that minimum spanning trees can be computed by an evolutionary algorithm within a runtime bound that is of the same magnitude as the best deterministic algorithms such as Kruskal and Prim for the mentioned problem. This result also shows that a sophisticated mutation operator can speed up computations drastically compared with a standard evolutionary approach analyzed by Neumann and Wegener [8].

In this paper, we consider a restricted mutation operator for the Eulerian cycle problem and examine the effect of the restriction on the runtime of the algorithms. The analysis of EAs with respect to their runtime behavior has become very popular in recent years. Starting with results on the optimization of pseudo boolean objective functions a lot of results have been obtained by now. First results consider the behavior of this class of randomized search heuristics on special functions that explain the behavior of evolutionary algorithms in different situations. One important issue is to analyze how evolutionary algorithms can cope with plateaus. Plateaus are regions in a search space where all search points have the same fitness. If such a plateau is large the search process frequently becomes difficult. Note that the number of different fitness values is often polynomial bounded in the input size whereas the number of search points is exponential. Then a pigeon hole argument implies that many search points have the same fitness. Plateaus have been for the first time examined by Jansen and Wegener [3] with respect to their impact on the runtime of an EA. They have investigated the effect of the structure of a plateau with respect to the runtime behavior of a simple evolutionary algorithm.

Since 2002 a lot of results concerning the runtime behavior of EAs on combinatorial optimization problems have been obtained. There are results on some of the best-known polynomial solvable problems such as sorting and shortest path (Scharnow, Tinnefeld, and Wegener [10]), maximum matchings (Giel and Wegener [1]), and minimum spanning trees (Neumann and Wegener [8]). In the case of NP-hard problems the first results have been achieved for the multi-objective minimum spanning tree problem (Neumann [6]) and a scheduling problem on two identical machines (Witt [11]).

For the Eulerian Cycle problem Neumann [7] has shown that a simple EA produces a Eulerian cycle of a Eulerian graph in expected number of  $O(m^5)$

steps if a jump operator is used for mutation. In contrast to this the expected optimization time gets exponential if the mutation operator is changed to exchange operations. This is one of the first results on the runtime behavior of evolutionary algorithms that deal with the important representation of permutations. This kind of representation is important for many important NP-hard combinatorial optimization problems such as the traveling salesman problem (see e.g. Michalewicz and Fogel [5] for different evolutionary approaches to solve this problem) or a wide class of scheduling problems (see e.g. Mattfeld and Bierwirth [4]). The analysis of Neumann [7] shows that jumps lead for his model to a plateau of constant fitness that can be left in a polynomial number of steps. Whereas in the case of exchange operations this plateau changes to local optima with a large inferior neighborhood.

The aim of this paper is to show that a restricted jump operator can speed up computations of an evolutionary algorithms. EAs often do steps that only waste time. This is the price a general search heuristic usually pays in contrast to a specialized algorithm. Restricting the mutation operator to the considered problem can shorten the time until a desired step occurs. In addition such a restriction can change the behavior of an EA on a plateau. We will show that a restricted version of the jump operator leads to an EA that computes a Eulerian cycle of an Eulerian graph in an expected number of  $O(m^3)$  steps. In addition we present an example graph which shows that our analysis is tight.

After having motivated our work, we introduce the model of the Eulerian cycle that will be analyzed in Section 2. In Section 3 we introduce restricted version of simple evolutionary algorithms that will be analyzed with respect to their runtime behavior. To show the advantages of these restriction we give lower bounds on the corresponding more general algorithms in Section 4 and present a runtime analysis of the restricted algorithms in Section 5. We finish with some conclusions.

## 2 Preliminaries

The Eulerian cycle problem can be seen as the first problem in graph theory. Proposed by Euler in 1736 as the famous Seven Bridges problem, its generalization can be described as follows and is known as the Eulerian cycle problem.

Given an undirected connected graph  $G = (V, E)$  on  $n$  vertices and  $m$

edges, compute a cycle such that every edge is used exactly once. Euler proved that such a tour exists if and only if the degree of each vertex is even. Graphs that contain a Eulerian cycle are called Eulerian. In the remainder of this paper we assume that all graphs are Eulerian.

For every  $n \in \mathbb{N}$  let us define  $[n] := \{k \in \mathbb{N} : 1 \leq k \leq n\}$ . We define the search space

$$S_m := \{\pi : [m] \rightarrow E \mid \pi \text{ is a bijection}\}.$$

$S_m$  contains all permutations of the edges of  $G$ . Thus, a search point  $\pi \in S_m$  corresponds to an order of the edges of  $G$ . Looking at the edges  $\pi(1), \pi(2), \dots, \pi(m)$  we can determine a longest path  $p = \pi(1)\pi(2)\dots\pi(l)$  for an appropriate  $l \leq m$  such that it holds  $\pi(i) \cap \pi(i+1) \neq \emptyset$  for all  $i \in [l-1]$  and  $\pi(l) \cap \pi(l+1) = \emptyset$ . If  $l = m$  the permutation  $\pi$  corresponds to a Eulerian tour. We formalize this in the following fitness function. For convenience we set  $\pi(m+1) = \emptyset$  and choose  $\pi(0)$  as a fixed one-element subset of  $\pi(1) \setminus \pi(2)$ . Define:

$$\text{path}(\pi) := 1 + \max\{k \in [m] \mid \forall i \in [k] : \pi(i) \subseteq \pi(i-1) \cup \pi(i+1)\}.$$

In the rest of this paper the path  $\pi(1)\pi(2)\dots\pi(\text{path}(\pi))$  will be named by  $p$ .

The fitness function describes the processing order to use the edges for a tour starting with the edge on position 1. Another advantage of this fitness function is that it can be easily evaluated. If the resulting path is short most of the edges in the permutation do not have to be considered.

For the Eulerian cycle problem algorithms have been designed that compute a Eulerian cycle in linear time. To analyze randomized search heuristics we use the knowledge that has been put into these algorithms. The following algorithm proposed by Hierholzer [2] computes an Eulerian cycle of a given Eulerian graph  $G$  and contains ideas which will be later used in the analysis of our algorithms.

**Algorithm 1 (Eulerian Cycle).**

1. Find a cycle  $C$  in  $G$
2. Delete the edges of  $C$  from  $G$
3. If  $G$  is not empty go to step 1.
4. Construct the Eulerian cycle from the cycles produced in step 1.

Neumann [7] has shown that simple randomized search heuristics are able to compute a Eulerian cycle of a Eulerian graph in expected polynomial time if a mutation operator is used that uses jump operators. The expected optimization change drastically, i.e. from polynomial to exponential, if one chooses exchange operations instead of jumps.

### 3 Randomized local search and the (1+1) EA

Randomized local search (RLS) and the (1+1) EA are perhaps the simplest randomized search heuristics that can be considered. They work on a population of size 1 and only use mutation to produce one single new individual in each generation. We consider variants of RLS and the (1+1) EA that are similar to the ones discussed by Neumann [7] for the Eulerian cycle problem. The difference is that the algorithms considered here work with a more restricted mutation operator. EAs often waste time by doing many steps that are not accepted. Our aim is to show that such a restriction of the mutation operator can speed up computations and to show how the structure of a combinatorial optimization problem can change from the EAs point of view because of such a restriction. The algorithms considered use a restricted jump operator for mutation. Jump operators have also been discussed by Scharnow, Tinnefeld, and Wegener [10] for the sorting problem. In the case of restricted jumps a jump operation  $jump(i)$  executed on a permutation  $\pi$  produces a new permutation  $\pi'$  by putting the element on position  $i$  at position 1 and shifting the remaining elements to the right. This restricted operator differs from the general jump operator that chooses two positions  $i$  and  $j$  in the permutation and puts the element at position  $i$  at position  $j$  while shifting the elements between into the appropriate direction. For RLS in one mutation step exactly one jump operation is executed. The executed jump is chosen according to the uniform distribution. We can describe the

restricted version of RLS as follows.

**Algorithm 2 (Randomized Local Search (restricted) (RLS<sub>r</sub>)).**

1. Choose  $\pi \in S_m$  uniformly at random.
2. Choose  $i \in [m]$  uniformly at random and define  $\pi'$  by jumping the element at position  $i$  to position 1 and shifting the elements between position 1 and position  $i$  one position to the right.
3. Replace  $\pi$  by  $\pi'$  if  $\text{path}(\pi') \geq \text{path}(\pi)$ .
4. Repeat Steps 2 and 3 forever.

Evolutionary algorithms use a mutation operator where more than one operation is possible in a mutation step. The (1+1) EA using the encoding of permutations is adopted from the well-known (1+1) ES (evolution strategy) and differs from RLS by the chosen mutation operator.

**Algorithm 3 (Mutation operator of (1+1) EA<sub>r</sub>).**

- 2') Define  $\pi'$  in the following way. Choose  $s$  according to a Poisson distribution with parameter  $\lambda = 1$  and perform sequentially  $s + 1$  restricted jump operations to produce  $\pi'$  from  $\pi$ .

In applications, we need a stopping criterion. For theoretical investigations it is a common use to investigate the number of fitness evaluations until an optimal solution has been achieved. This is called the runtime of the algorithm. Often the expectation of this value is considered which is called the expected optimization time of the considered algorithms.

The algorithms introduced here will be compared with variants called RLS and (1+1) EA that use the general jump operator. RLS and (1+1) EA have already been analyzed with respect to their runtime behavior on the Eulerian cycle problem. We will compare the results obtained for these algorithms with our results on the restricted versions.

## 4 A lower bound for RLS and the (1+1) EA

We consider RLS and the (1+1) EA with the general mutation operator in this section. Neumann [7] has proven that both algorithms need in expectation at most  $O(m^5)$  steps until they compute a Eulerian cycle of a Eulerian graph. We now show that both algorithms need at least  $\Omega(m^4)$  steps to find a Eulerian cycle.

Without loss of generality, let  $m$  be a multiple of 8. Consider the example graph  $G_m$  given in Figure 1, which consists of two cycles of length  $m/2$  that share exactly one vertex  $v_k$ .

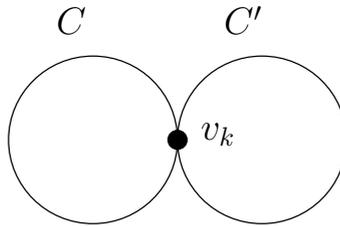


Figure 1: Instance  $G_m$ : Two cycles of length  $m/2$  sharing one vertex

**Theorem 1.** *The expected optimization time of RLS and the (1+1) EA on the graph  $G_m$  is lower bounded by  $\Omega(m^4)$ .*

*Proof.* Let us consider the situation for RLS where  $|p| \geq 3$  for the first time. With probability  $1 - o(1)$  all edges of  $p$  belong to only one of the two cycles of  $G_m$ . W.l.o.g. we may assume that  $p$  is contained in  $C$ . The first possibility that an edge of the other cycle  $C'$  can be integrated in the path  $p$  is given if  $v_k$  is the first vertex of  $p$ . Before one of the edges of  $C'$  adjacent to  $v_k$  is integrated in  $p$ ,  $p$  cannot contain any other edge of  $C'$ . But with probability  $1/3$  (under the condition that the first small path is contained in  $C$ ), the two edges of  $C$  adjacent to  $v_k$  are integrated in  $p$  before one of the two edges of  $C'$  adjacent with  $v_k$  is contained in  $p$ . If this is the case, there is a moment where the current path  $p$  is the cycle  $C$ . The expected time until  $C$  has been produced is upper bound by  $O(m^3)$  as the path has to be lengthened at most  $m/2$  times and the expected waiting time for such a mutation step is  $O(m^2)$ . Thus, with probability  $1/3 - o(1)$  there is a moment where  $p$  is one of the two cycles (w.l.o.g.  $C$ ), and until that moment no edge of  $C'$  was contained in  $p$ . Let us denote the first vertex of the path  $p$  as  $x_0$  and the other vertices of  $C$

clockwise as  $x_1, x_2, \dots, x_{\frac{m}{2}-1}$ . We do this modulo  $m/2$ , in particular, by the vertex  $x_{\frac{m}{2}}$  we mean the vertex  $x_0$ . Let  $d \in \{0, 1, \dots, \frac{m}{2} - 1\}$  be chosen such that  $x_d = v_k$ . By the symmetry of  $C$  and since the cycle  $C'$  can be ignored until this moment, with probability at least a half,  $d \in \{\frac{m}{8}, \frac{m}{8} + 1, \dots, \frac{3m}{8}\}$  and thus, the distance (measured in the number of edges) of  $x_0$  to  $v_k$  is at least  $m/8$ . But for the next improvement of RLS  $v_k$  has to be the first vertex of  $p$ . Therefore, we are interested in the expected number of accepted steps until the first vertex of  $p$  is  $x_d = v_k$ .

**Claim 1.** *The expected number of accepted steps until  $x_d$  is the first vertex of the path  $p$  is  $d(\frac{m}{2} - d)$ .*

*Proof.* Let  $t_k$  be the expected number of accepted steps until the first vertex of  $p$  is  $x_k$  for the first time (after  $p$  has become the cycle  $C$ ) for all  $0 \leq k \leq \frac{m}{2} - 1$ . Clearly,  $t_0 = 0$ . Because of the symmetry of  $C$ ,  $t_k$  is also the expected number of accepted steps until the first vertex is  $x_0$  for the first time, starting at the vertex  $x_k$ . Then, in the first step the first vertex of  $p$  changes from  $x_k$  to the adjacent vertices  $x_{k-1}$  or  $x_{k+1}$  each with probability  $1/2$ . Thus,

$$t_k = 1 + \frac{1}{2}t_{k-1} + \frac{1}{2}t_{k+1}$$

holds for all  $0 \leq k \leq \frac{m}{2} - 1$ . Note, that the vertices are named modulo  $m/2$  and therefore  $t_{\frac{m}{2}} = t_0 = 0$ . This is a linear system of equations with  $t_0, t_1, \dots, t_{\frac{m}{2}-1}$  as variables. It is easy to see that this is a regular linear system of equations. Thus, it has a unique solution. One verifies that  $t_k = k(\frac{m}{2} - k)$  for all  $0 \leq k \leq \frac{m}{2} - 1$  solves this linear system of equations.  $\square$

Since with probability at least a half  $d \in \{\frac{m}{8}, \frac{m}{8} + 1, \dots, \frac{3m}{8}\}$ , the expected number of accepted steps until  $x_d = v_k$  is the first vertex in the path  $p$  is at least of order  $\Omega(m^2)$ .

The (1+1) EA has the possibility of executing more than one jump operation in a mutation step. We estimate the probability that the (1+1) EA accepts a mutation step with  $r \geq 2$  jump operations that are relevant for the fitness function. There may be additional jump operations in the part of  $\pi$  not belonging to the path  $p$  that do not effect the fitness function. The probability that at least  $r$  jump operations are performed in a mutation step is at most  $e$  times the probability that exactly  $r$  jump operations are performed. Thus, this probability is bounded by  $\frac{1}{(r-1)!}$ . In order to be accepted at least half of the  $r$  edges have to jump into the path  $p$ . The probability for a single jump into respectively out of the path  $p$  to be accepted is of order

$O(\frac{1}{m^2})$  respectively  $O(\frac{1}{m})$ . And the  $r$  relevant jumps can be executed in  $r!$  different orders. Thus, the probability that  $r$  relevant jumps are accepted in a mutation step is of order  $O\left(\frac{r!}{(r-1)!m^{2\cdot\frac{r}{2}}m^{\frac{r}{2}}}\right) = O(rm^{-\frac{3r}{2}})$ . Hence, with probability  $1 - o(1)$  there is no mutation consisting of more than two relevant jump operations within  $O(m^4)$  steps. Since there is no possibility that edges of  $p$  – except for the two edges at the beginning and the end of  $p$  – can be replaced by other edges through mutations with at most two relevant jump operations, the only difference between RLS and the (1+1) EA is, that the (1+1) EA can operate up to a factor of two faster.

Thus, RLS and the (1+1) EA need in expectation at least  $\Omega(m^2)$  accepted mutation steps. Moreover, the expected waiting time of RLS and the (1+1) EA until such a mutation is accepted is  $\Omega(m^2)$ . Hence, both algorithms need  $\Omega(m^4)$  steps to produce an optimal solution for the example graph  $G_m$ .  $\square$

## 5 Analysis of the restricted operator

The analysis in the previous section has shown that there are situations where RLS and the (1+1) EA need an expected number of  $\Omega(m^4)$  steps to reach an improvement. We will investigate the corresponding algorithms  $RLS_r$  and the (1+1)  $EA_r$  that work with the restricted mutation operator and show that in each situation an improvement is found in an expected number of  $O(m^2)$  steps. One reason for this improvement is that the plateau that has to be coped with changes its structure such that an improvement is easier to obtain. On the other hand the expected waiting time for an accepted offspring can be reduced by a factor of  $m$  using the restricted mutation operator. This leads to an expected optimization time for  $RLS_r$  and the (1+1)  $EA_r$  which is upper bound by  $O(m^3)$ . Additionally, we show that there are graphs for which  $RLS_r$  and the (1+1)  $EA_r$  need in expectation a runtime of the same magnitude with probability close to 1.

**Theorem 2.** *The expected time until  $RLS_r$  and the (1+1)  $EA_r$  have computed a Eulerian cycle is bounded by  $O(m^3)$ .*

*Proof.* We distinguish two cases. In the first case the current path is not a cycle. Then there exists at least one edge that can jump to position 1 and lengthen the path. In the case that  $p$  represents a cycle  $C$  which is not a Eulerian cycle, the complement  $G^C$  of  $C$  in  $G$  is a subgraph whose

components are all Eulerian. Let  $d'(v)$  denote the degree of  $v$  in  $G^C$  for every vertex  $v$  that is in the cycle  $C$  and in the subgraph  $G^C$ . For all vertices that are in the cycle  $C$  but not in the subgraph  $G^C$  we set  $d'(v) := 0$ . Since  $G$  is Eulerian,  $G^C$  has at least one vertex in common with  $C$ . Thus, there is at least one vertex  $v$  in  $C$  with  $d'(v) \geq 2$ . Let  $l$  be the length of the path  $p$  and let  $e_i(p)$ ,  $1 \leq i \leq l$ , denote the  $i$ th edge of  $p$ . We call  $v_1(p) := e_1(p) \setminus e_2(p)$  the starting point of the path  $p$ . If  $d'(v_1(p)) = 0$ , the only accepted jump is  $jump(l)$ . But after at most  $l-1$  such jumps  $d'(v_1(p)) \geq 2$ . In such a situation there are  $d'(v_1(p))+1$  accepted jumps. Besides  $jump(l)$ , all  $d'(v_1(p))$  edges of  $G^C$  containing  $v_1(p)$  can jump to the first position of  $p$  (shifting all the edges of  $p$  by one). Consequently, the probability that the path is lengthened is  $\frac{d'(v_1(p))}{d'(v_1(p))+1} \geq \frac{2}{3}$ . Hence, the expected number of jumps until the path is improved is at most  $\frac{3}{2}l \leq \frac{3}{2}m$ . Since in average at least every  $m$ th jump is accepted, this implies an expected time for an improvement of  $O(m^2)$ . The number of improvements is at most  $m-1$ , which completes the proof.  $\square$

To prove a matching lower bound, we consider the graph  $G'$  consisting of  $m/4$  cycles  $C_i$ ,  $0 \leq i \leq m/4 - 1$  of length 4.  $C_i$  consists of the vertices  $v_{3i}, v_{3i+1}, v_{3i+2}, v_{3(i+1)}$  and the edges  $\{v_{3i}, v_{3i+1}\}, \{v_{3i}, v_{3i+2}\}, \{v_{3i+1}, v_{3(i+1)}\}$  and  $\{v_{3i+2}, v_{3(i+1)}\}$ . The number of vertices in  $G'$  is  $3m/4 + 1$ . Note that cycle  $C_i$ ,  $0 \leq i \leq m/4 - 2$ , and  $C_{i+1}$  intersect in  $v_{3(i+1)}$ .

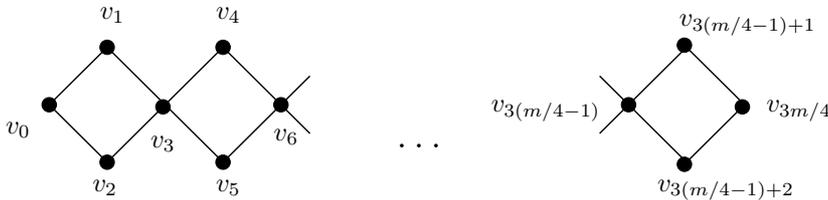


Figure 2: Example graph  $G'$

**Theorem 3.** *With probability  $1 - o(1)$ ,  $RLS_r$  and the  $(1+1)$   $EA_r$  need  $\Omega(m^3)$  steps to find a Eulerian cycle in  $G'$ .*

*Proof.* We call the vertices  $v_{3i}$ ,  $1 \leq i \leq m/4 - 1$ , turning points of the graph  $G'$ . As in the proof of Theorem 2, we call  $v_1(p) := e_1(p) \setminus e_2(p)$  the starting point of the path  $p$ , where  $e_1$  denotes the first and  $e_2$  the second edge of the path  $p$ . At the beginning of  $RLS_r$  the path  $p$  consists of only a few edges.

More precisely, the probability that the length of the path is shorter than for example  $m/20$  is clearly  $1 - e^{-\Omega(m)}$ . We now show the following claim.

**Claim 2.** *Let the current path  $p_1$  at a certain time  $t_1$  in  $RLS_r$  be not a cycle. And let  $t_2 > t_1$  be the first time such that the current path  $p_2$  is a cycle. Then*

$$|\{0 \leq i \leq m/4 - 1 : C_i \cap p_1 = \emptyset \neq C_i \cap p_2\}| \leq \frac{m}{40} \quad (1)$$

with probability  $1 - e^{-\Omega(m)}$ .

*Proof.* Assume that (1) does not hold. Then at least  $m/40$  times between  $t_1$  and  $t_2$  the starting point  $v_1(p)$  was a turning point. In each of this situations the probability that the next edge that jumps at the first position of the path  $p$  is in the same cycle  $C_i$  as the edge that was in the first position of  $p$  before this jump is  $1/3$ . And with probability  $2/3$  this two edges are in different cycles  $C_i$  and  $C_{i+1}$ . Since the path  $p$  is not closed until the time  $t_2$ , only up to two times between  $t_1$  and  $t_2$  the starting point  $v_1(p)$  was a turning point and these two edges were in the same cycle. But at least  $m/40$  times the starting point  $v_1(p)$  was a turning point and those two edges were in different cycles. The probability for this is  $e^{-\Omega(m)}$  which proves the claim.  $\square$

With probability  $1 - e^{-\Omega(m)}$  the path  $p$  at the beginning of  $RLS_r$  is shorter than  $m/20$ . That means, it contains edges of at most  $m/40$  cycles. Using the claim, we get the following: when the path  $p$  is a cycle for the first time, it contains edges of at most  $m/40 + m/40 = m/20$  cycles with probability  $1 - e^{-\Omega(m)}$ . Thus, the length of  $p$  is bounded by  $m/5$  at this time with probability  $1 - e^{-\Omega(m)}$ . Another consequence of the claim is the following: Let  $l_1$  be the length of the current path  $p$  at a certain time when  $p$  is a cycle. And let  $l_2$  be the length of the current path  $p$  at the first time when the current path is a cycle again after  $p$  was not a cycle for a while. Then  $l_2 - l_1 \leq m/10$  with probability  $1 - e^{-\Omega(m)}$ , since only edges from at most  $m/40$  cycles  $C_i$  can jump into  $p$  (with probability  $1 - e^{-\Omega(m)}$ ).

Therefore with probability  $1 - e^{-\Omega(m)}$  there exists a moment when the current path  $p$  is a cycle of length at least  $m/3$  and at most  $m/3 + m/10 \leq m/2$ . Hence, there are at least  $m/8$  cycles left that have to be integrated. Each time when the starting point  $v_1(p)$  is a turning point and only one edge  $e$  of the four edges adjacent to  $v_1(p)$  is contained in  $p$  we have the following situation: the probability that the next edge that jumps at the first position of  $p$  is in the same cycle  $C_i$  as the edge  $e$  is  $1/3$ . Thus, this will occur in

average in  $1/3$  of the  $m/8$  times. Using Chernoff bounds, with probability  $1 - e^{-\Omega(m)}$  this happens at least  $m/30$  times. After such an event, at least  $m/6$  edges have to jump to the first position of the path  $p$  until the starting point  $v_1(p)$  of  $p$  is a turning point, such that only one of the four edges adjacent to  $v_1(p)$  is contained in  $p$ . Therefore, with probability  $1 - e^{-\Omega(m)}$ , at least  $(m/6)(m/30) = m^2/180$  edges have to jump at the first position of the path  $p$  until the Eulerian cycle is found by  $\text{RLS}_r$ . Since the maximal degree of  $G'$  is 4, in average at most  $3/m$  of all jumps is accepted. Thus, using the Chernoff bounds once more, the number of jumps until  $\text{RLS}_r$  has found a Eulerian cycle is  $\Omega(m^3)$  with probability  $1 - e^{-\Omega(m)}$ .

In contrast to  $\text{RLS}_r$  more than one jump operation per mutation is possible in the mutation operator of the  $(1+1)$   $\text{EA}_r$ . The probability that a jump is accepted is at most  $3/m$  in the example graph  $G'$ , since the maximum degree of  $G'$  is 4. Thus, with probability  $1 - o(1)$  there is no accepted mutation with more than three jumps within  $O(m^3)$  steps. Therefore, none of the circles can be integrated in one mutation with probability  $1 - o(1)$ . Hence, the only difference between  $\text{RLS}_r$  and the  $(1+1)$   $\text{EA}_r$  is that  $(1+1)$   $\text{EA}_r$  is up to a factor of 3 faster than  $\text{RLS}_r$ . This shows the claim also for the  $(1+1)$   $\text{EA}_r$ .  $\square$

## 6 Conclusions

In the case of combinatorial optimization problems often a lot is known about the structure of a given problem. This knowledge can lead to more sophisticated evolutionary algorithms. For the Eulerian cycle problem we have considered the runtime behavior of simple evolutionary algorithms that work with a restricted mutation operator. We have proven an upper bound of  $O(m^3)$  for the expected runtime of these restricted algorithms. This beats the up to now best upper bound on the more general versions of these algorithms by a factor  $m^2$ . The lower bound on the algorithms with the restricted mutation operators shows that there are graphs whose optimization time is  $\Omega(m^3)$  with probability close to 1. We also have obtained a lower bound of  $\Omega(m^4)$  for the more general algorithms which strengthens our claim that restricting the mutation operator can help to come up with faster evolutionary algorithms in some cases.

## References

- [1] O. Giel and I. Wegener. Evolutionary algorithms and the maximum matching problem. In *Proc. of 20th STACS*, volume 2607 of *Lecture Notes in Computer Science*, pages 415–426, 2003.
- [2] C. Hierholzer. Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. In *Math. Ann.*, volume 6, pages 30–32, 1873.
- [3] T. Jansen and I. Wegener. Evolutionary algorithms — how to cope with plateaus of constant fitness and when to reject strings of the same fitness. In *IEEE Trans. on Evolutionary Computation*, volume 5, pages 589–599, 2001.
- [4] D. C. Mattfeld and C. Bierwirth. An Efficient Genetic Algorithm for Job Shop Scheduling with Tardiness Objectives. In *European Journal of Operational Research*, volume 155, pages 616–630, 2004.
- [5] Z. Michalewicz and D. B. Fogel *How to solve it*. Springer-Verlag, Berlin, 2004.
- [6] F. Neumann. Expected Runtimes of a Simple Evolutionary Algorithm for the Multi-objective Minimum Spanning Tree Problem. In Yao et al., editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 80–89, Springer, Berlin, 2004.
- [7] F. Neumann. Expected Runtimes of evolutionary algorithms for the Eulerian cycle problem. In *Proc. of the Congress on Evolutionary Computation 2004 (CEC 2004)*, volume 1, IEEE Press, pages 904–910, 2004.
- [8] F. Neumann and I. Wegener. Randomized Local Search, Evolutionary Algorithms, and the Minimum Spanning Problem. In Deb et al., editors, *Genetic and Evolutionary Computation Conference - GECCO 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 713–724, Springer, Berlin, 2004.
- [9] G. R. Raidl, G. Koller, and B. A. Julstrom. Biased mutation operators for subgraph-selection problems. In *IEEE Transactions on Evolutionary Computation*, 2006 (to appear).

- [10] J. Scharnow, K. Tinnefeld, and I. Wegener (2002). Fitness landscapes based on sorting and shortest paths problems. In *Proc. of Parallel Problem Solving from Nature — PPSN VII*, volume 2939 of *Lecture Notes in Computer Science*, pages 54–63, Springer, Berlin, 2002.
- [11] C. Witt (2005). Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS '05)*, volume 3404 of *Lecture Notes in Computer Science*, pages 44–56, Springer, Berlin, 2005.