Computing Single Source Shortest Paths using Single-Objective Fitness Functions*

Surender Baswana† Somenath Biswas† Benjamin Doerr‡

Tobias Friedrich§ Piyush P. Kurur† Frank Neumann‡

† Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
208016 Kanpur, India

‡ Max-Planck-Institut für Informatik
Campus E1 4
66123 Saarbrücken, Germany

§ International Computer Science Institute
1947 Center St., Suite 600
94704 Berkeley, CA, USA

ABSTRACT

Runtime analysis of evolutionary algorithms has become an important part in the theoretical analysis of randomized search heuristics. The first combinatorial problem where rigorous runtime results have been achieved is the well-known single source shortest path (SSSP) problem. Scharnow, Tinnefeld and Wegener [PPSN 2002, J. Math. Model. Alg. 2004] proposed a multi-objective approach which solves the problem in expected polynomial time. They also suggest a related single-objective fitness function. However, it was left open whether this does solve the problem efficiently, and, in a broader context, whether multi-objective fitness functions for problems like the SSSP yield more efficient evolutionary algorithms. In this paper, we show that the single objective approach yields an efficient (1+1) EA with runtime bounds very close to those of the multi-objective approach.

Categories and Subject Descriptors: F.2 [Theory of Computation]: Analysis of Algorithms and Problem Complexity

General Terms: Theory, Algorithms, Performance

1. INTRODUCTION

Evolutionary algorithms (EAs) have been successfully applied to a wide range of combinatorial optimization problems. Understanding the success of these randomized search heuristics by rigorous analyses has gained increasing interest in recent years. One approach to analyze evolutionary algorithms is by means of carrying out a rigorous runtime analysis. The line of research has been started by analyzing the behavior of EAs on simple pseudo-Boolean functions [13, 15, 7]. Later on, some of the best known combinatorial optimization problems have been investigated [9, 14, 18].

The first problem of this kind where rigorous runtime results have been achieved is the well-known single source shortest path (SSSP) problem [17]. Computing shortest paths in a given graph is one of the fundamental problems in computer science and still an important field of research [16, 2, 11]. In the area of randomized search heuristics related problems such as vehicle routing [8] and routing problems in networks [6, 10] have been tackled. Therefore, it seems to be important to understand the basic SSSP problem from a theoretical point of view to gain new insights that will help practitioners solving related problems arising in applications.

In [17], the authors examined a simple EA together with a multi-objective fitness function which makes the EA mimic Dijkstra's algorithm for the SSSP problem [3]. Its optimization time (that is, the number of fitness evaluations used) is $\mathcal{O}(n^3)$, where n is the number of vertices of the input graph. Additionally, they have given a single-objective approach which they suppose to be efficient. However, the authors state that they were not able to analyze their approach with respect to the runtime behavior. In this paper, we point out that the multi-objective fitness function is not necessary. We consider the proposed single-objective approach and show how it solves the SSSP problem in expected polynomial time.

In the case that a simple randomized local search procedure is used, it is not too hard to prove that such an approach again follows the ideas of Dijkstra's algorithm. This follows from the fact that the new solution is constructed in the 1-neighborhood of the previous one. Simple EAs may construct solutions that are further away from the current search point. For the SSSP, this may result in that shortest paths for certain vertices can get lost in exchange for other short paths. This is a crucial difference to what happens in Dijkstra's algorithm.

In fact, it has been shown recently (for a different problem though) that this more general search behavior can increase the optimization time from polynomial for randomized local

*Work supported by the Collaborative Research Program of the Research I Foundation, IIT Kanpur. Tobias Friedrich's work was also partially supported by a postdoctoral fellowship from the German Academic Exchange Service (DAAD).

FOGA'09, January 9–11, 2009, Orlando, Florida, USA.

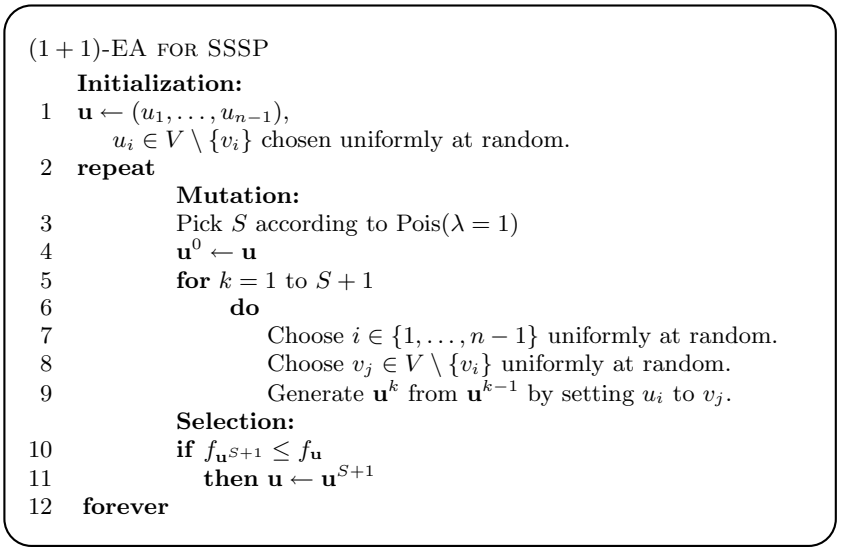

Figure 1. The $(1+1)$-EA with single-objective fitness function f for the SSSP problem.

search to exponential for EAs, even under conditions that look highly favorable for the EA [5]. In this light, the work of [17] raises the question whether such a phenomenon also occurs for the SSSP problem.

We answer this question and prove that the more flexible mutation allowed in EAs still leads to a polynomial optimization time, more precisely, to an expected optimization time of $\mathcal{O}(n^3 \log(n + w_{\max}))$, where $w_{\max}$ is the largest of the (integral) edge weight. Our analysis uses new structural insights of the SSSP problem which point out how randomized search heuristics may achieve progress for this kind of problem even if they may not follow the ideas of Dijkstra. These positive results are later on complemented with lower bounds for the examined algorithms which show that our analyses are almost tight.

We should note that the results in [17] have recently been improved in [4]. In addition to n denoting the number of vertices of the input graph, let ℓ denote the maximum number of edges of a shortest path. Then [4] shows that the optimization time of the multi-objective EA proposed in [17] is $\mathcal{O}(n^2 \max\{\log(n), \ell\})$ with high probability. The methods of [17] would only yield an optimization time of $\mathcal{O}(n^2 \ell \log(n))$ in expectation. Hence the improvements are both a stronger bound on the expected optimization time for graphs having small diameter and a sharper concentration bound for the optimization time in general graphs. However, no progress was made on whether the SSSP can be approached via single-objective methods

The outline of the paper is as follows. In Section 2, we present the single-objective approach that will be analyzed throughout this paper. Section 3 shows that the single-objective approach solves the SSSP in expected polynomial time and Section 4 gives almost matching lower bounds. Finally, we finish with some concluding remarks.

2. ALGORITHM

We consider the well-known single source shortest paths problem (SSSP). Given a directed graph $G = (V, E)$ with $V = \{v_0, \ldots, v_{n-1}\}$ and $E = \{e_1, \ldots, e_m\}$. Additionally, a weight function $w\colon E \to \mathbb{N}$ is given which assigns integer distance values to the edges. Let $w_{\max} = \max_{e \in E} w(e)$. We may extend w to all (u, v), $u \neq v$, by setting $w((u, v)) = \infty$ iff $(u, v) \notin E$. Given a distinguished vertex $s \in V$, say $s = v_0$, the single source shortest path problem is to compute for each vertex v_i, $1 \leq i \leq n-1$, a shortest path from s to v_i. Without loss of generality, we assume that such a path exists for each v_i. It is both a well known fact and easy to see that a set of such shortest paths always forms a tree, which is therefore called a *shortest path tree*.

We examine the (1+1) EA for the SSSP problem (see Figure 1) already investigated in [17]. The search space consists of all candidate solutions $\mathbf{u} = (u_1, \ldots, u_{n-1}) \in \{v_0, \ldots, v_{n-1}\}^{n-1}$ where $u_i \neq v_i$. The goal of the algorithm is to find a $\mathbf{u} = (u_1, \ldots, u_{n-1})$ such that the edges (u_i, v_i) form a shortest path tree T rooted at s (that is, for each vertex v_i, $1 \leq i \leq n-1$, u_i is the predecessor of v_i in the shortest path from s to v_i contained in T).

The initial solution is chosen uniformly at random from the search space. In each iteration one single offspring is produced by mutation. The mutation consists of changing the predecessor of some $S+1$ vertices uniformly at random, where the value of S is chosen according to the Poisson distribution with parameter $\lambda = 1$. This distribution is as proposed in [17].

There are many invalid solutions, i. e., search points that do not represent trees. Assigning a cost of ∞ to such solutions, it is hard for a randomized search heuristic to obtain a valid solution. Due to this a multi-objective fitness function has been investigated for this problem and an upper bound of $\mathcal{O}(n^3)$ on the expected optimization time has been proven. Additionally, a single-objective fitness function has been given which leads the algorithm towards valid solutions. Instead of using the value ∞ for solutions that do not represent trees each vertex that is not connected to the source is penalized. Using penalty values is a common approach for handling constraints (see e. g. [12]) and leads the algorithm towards feasible solutions.

In [17] the authors state that they are not able to analyze

this approach. Our goal is to show that this single-objective approach indeed works and finds an optimal solution in expected polynomial time.

Now we describe the single objective function that is investigated in the rest of this paper. Consider a candidate solution $\mathbf{u} = (u_1, \ldots, u_{n-1})$ where u_i is supposed to be the immediate predecessor of the vertex v_i. Associated with $\mathbf{u}$ consider the subgraph $T_{\mathbf{u}}$ of the input graph G consisting of those pairs (u_j, v_j) which are edges in G. For a vertex v, if there is a path in $T_{\mathbf{u}}$ from the source s to v it has to be unique. Let γ_v denote the unique path in such cases. Whenever such unique path γ_v exists for a vertex v we define its cost $f_{\mathbf{u}}(v)$ to be the sum of the weights of the edges in γ_v. On the other hand if v is unreachable from s in $T_{\mathbf{u}}$, then the cost $f_{\mathbf{u}}(v)$ is set to $d_{\text{penalty}} := n \cdot w_{\max}$. The fitness $f_{\mathbf{u}}$ of a candidate solution $\mathbf{u}$ is given by

$$f_{\mathbf{u}} := \sum_{i=1}^{n-1} f_{\mathbf{u}}(v_i).$$

In our analysis of the runtime behavior of the algorithm, we bound the number of evaluations of the fitness function required to reach an optimal solution. The expected optimization time refers to the expectation of this value. The difficulty in analyzing the stated approach lies in the occurrence of mutation steps that change more than one predecessor. In this case, shortest paths found during the run of the algorithm may get lost. Assuming that in each mutation step just one vertex changes its predecessor it is easy to prove an upper bound of $\mathcal{O}(n^3)$ as for the multi-objective approach given in [17] by following the ideas of Dijkstra's algorithm.

3. UPPER BOUND

We now prove an upper bound on the running time of the (1+1) EA. As mentioned before we will assume that the input graph is a complete graph. We start the algorithm with each vertex picking one of the edges incident on it at random. It then proceeds in stages where in each stage it decides to *mutate* $k+1$ vertices, k picked according to the Poisson distribution of mean 1, accepting the mutation if and only if it does not lead to an increase in the objective function.

Let $\mathbf{u}_i$ denote the candidate solution after i mutations. Recall the definition of the objective function $f_{\mathbf{u}_i}$. For ease of notation let $f_i(v)$ denote the cost $f_{\mathbf{u}_i}(v)$ associated with the vertex v and let f_i denote the fitness $f_{\mathbf{u}_i}$ of the candidate solution $\mathbf{u}_i$. Similarly let T_i denote the subgraph $T_{\mathbf{u}_i}$ of valid edges (i.e. edges of the input graph G) present in the candidate solution $\mathbf{u}_i$.

The key idea behind our upper bound is the following: We prove that in each mutation the value of the objective function reduces at least by a factor of $1 - \Omega(n^{-3})$ with high probability. Thus in time $\mathcal{O}(n^3)$ the value of the objective function reduces by a multiplicative factor of at most $c < 1$. The value of the objective function is less than $n \cdot d_{\text{penalty}}$ initially. Assuming the edge weights are integers, after $\mathcal{O}\left(n^3 \cdot (\log n + \log d_{\text{penalty}})\right)$ mutations the algorithm will find the shortest distance tree with high probability. Note, that this approach is similar to the expected multiplicative weight decrease for the analysis of the (1+1) EA and the minimum spanning tree problem [14]. The difference of our approach to this one is that we do not give a set

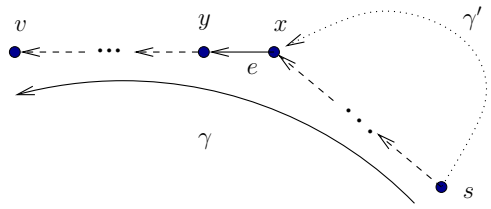

Figure 2. How a good edge enters the solution.

of operations that turn the current solution into an optimal one. Instead of this we give a set of operations that lead to a solution whose distance to an optimal one is by a factor of $(1 - 1/n)$ smaller than the distance of the current solution to an optimal one.

We now define two parameters called delay and gap which we use to analyze the rate at which the objective function decreases. Let T be the shortest path tree in the graph (if there are more than one pick one and fix it for the rest of the section). Let $\delta(s, v)$ denote the distance of v from s in T. The *delay* of the vertex v after i mutations is defined to be $d_i(v) = f_i(v) - \delta(s, v)$. By *gap* g_i of the candidate solution after i mutations we mean $g_i = f_i - \sum_v \delta(s, v)$. Notice that $g_i = \sum_v d_i(v)$. When the gap drops to 0 we have the desired shortest distance tree. Thus gap measures how far the fitness function is currently from the final optimal value $\sum_v \delta(s, v)$.

LEMMA 1. *Let g_i denote the gap after i mutations then the conditional expectation $\mathbb{E}[g_{i+1} \mid g_i = g]$ is given by.*

$$\mathbb{E}[g_{i+1} \mid g_i = g] \leq g\left(1 - \frac{1}{3 \cdot n^3}\right)$$

Proof. Let the current gap g_i be g. Since the total delay $\sum_v d_i(v)$ is g, there is at least one vertex v such that its delay $d_i(v)$ is at least $\frac{g}{n}$. Let T be the shortest path tree in the graph that we have fixed for our analysis. Let $\gamma = \langle s = v_0, \ldots, v_\ell = v \rangle$ be the path from s to v in T. Denote by $E(\gamma)$ the set of its edges. For an edge $e = (v_k, v_{k+1}) \in E(\gamma)$ let $D(e) = d_i(v_{k+1}) - d_i(v_k)$ be the difference of the delays of the end points of e. Since $d_i(s) = 0$ we have

$$\begin{aligned} D(\gamma) &:= \sum_{e \in E(\gamma)} D(e) \\ &= d_i(v_\ell) - d_i(v_{\ell-1}) + \ldots + d_i(v_1) - d_i(v_0) \\ &= d_i(v_\ell) - d_i(v_0) \\ &\geq \frac{g}{n}. \end{aligned} \tag{1}$$

We define an edge e in γ to be *positive* if the difference of the delays of its end points, $D(e)$, is positive. Let $E^+(\gamma)$ denote the set of positive edges in γ. Then

$$D(E^+(\gamma)) := \sum_{e \in E^+(\gamma)} D(e) \geq \sum_{e \in E(\gamma)} D(e) \geq \frac{g}{n} \tag{2}$$

Consider any positive edge $e = (x, y)$ in the path γ. We claim that x has to be connected to the source s in the graph T_i. Otherwise the value of the fitness function $f_i(x) =$

d_{penalty}. Since $f_i(y) \leq d_{\text{penalty}}$, it follows that $D(e) = d(y) - d(x) \leq 0$ and hence e is not positive.

Having proved that x is reachable from s we prove that the edge e is not present in the current candidate solution. Otherwise e will be present in the graph T_i and since the vertex x is reachable from s in T_i, so will be y. Let γ' be the path from s to x in T_i. Then the path to the vertex y from the source s in T_i is the path γ' followed by the edge e (see Figure 2).

Notice that since the edge e is present in the shortest path tree T we have $\delta(s,y) = \delta(s,x)+w(e)$. As a result the delay of y is given by $d_i(y) = f_i(x)+w(e)-(\delta(s,x)+w(e)) = d_i(x)$ and hence $D(e) = d_i(y) - d_i(x) = 0$. This contradicts the fact that the edge e is positive and hence it is not present in the graph T_i.

Let M_i be the event that in the ith mutation only one vertex is mutated. The events M_i's are mutually independent. In the ith mutation step $k_i + 1$ vertices are mutated, where k_i follows a Poisson distribution with mean 1. Therefore, we have $\mathrm{P}[M_i] = \frac{1}{e} \geq \frac{1}{3}$. Consider any positive edge $e = (x, y)$. Given that the event M_{i+1} has occurred the probability that the vertex y switches its predecessor to x is at least $1/n^2$. If such a switch occurs the gap reduces by an amount equal to $D(e)$. This is because, as noted in the previous paragraph, x is reachable from s in the current solution, and the edge e between x and y is in the shortest path tree T, the delay of y will become the same as that of x as e comes into the solution. Therefore given M_{i+1} the expected decrease in gap, Δg, is given by

$$\mathbb{E}[\Delta g \mid g_i = g, M_{i+1}] \geq \sum_{e \in E^+(\gamma)} \frac{1}{n^2} D(e) \geq \frac{1}{n^3} g.$$

Since the $\mathrm{P}[M_{i+1}] \geq \frac{1}{3}$, the expected decrease in gap is at least $\frac{1}{3 \cdot n^3} g$. Hence the expected gap after the $(i+1)$-st mutation will be less than or equal to $(1 - \frac{1}{3 \cdot n^3})g$ provided the gap after i mutations is g. □

As a corollary of Lemma 1 we have:

Corollary 2.

$$\mathbb{E}[g_{i+j} \mid g_i = g] \leq g \left(1 - \frac{1}{3 \cdot n^3}\right)^j.$$

Theorem 3. *The expected optimization time of the (1+1) EA on the shortest path problem with integer edge weights is $\mathcal{O}\left(n^3 \cdot (\log n + \log w_{\max})\right)$, where $w_{\max}$ is the maximum of the weights of all edges in the graph.*

Proof. Recall that the gap g_0 at the beginning of the algorithm is at most $n \cdot d_{\text{penalty}}$ and we have set d_{penalty} to be $n \cdot w_{\max}$. We partition the sequence of iterations into epochs. Each epoch consists of a sequence of $6n^3$ iterations in the algorithm. Consider any such epoch. It follows from Corollary 2 that conditioned on the event that the epoch begins with gap $g_{begin} = g$, the expected value of the gap g_{final} after the epoch is given by

$$\mathbb{E}[g_{final} \mid g_{begin} = g] \leq g \left(1 - \frac{1}{3n^3}\right)^{6n^3} \leq \frac{g}{e^2}.$$

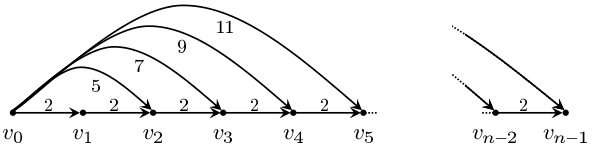

Figure 3. The worst case graph. The edges at the bottom form the shortest path tree. All edges not drawn have weight $w_{\max}$.

Using Markov's Inequality it follows that

$$\Pr\left[g_{final} \geq \frac{g_{begin}}{e}\right] \leq 1/e$$

We call an epoch *good* if the gap at the end of the epoch is less than $1/e$ times the gap in the beginning of the epoch. It follows that an epoch is good with probability at least $1-1/e$. Furthermore, each epoch is good or bad independent of other epochs (owing to the independence of mutations in each iteration). Thus it follows that $\lceil \ln g_0 \rceil$ good epochs suffice to reduce the gap below 1. Since each epoch consists of $6n^3$ iterations, the expected number of iterations in the (1+1) EA is $\mathcal{O}(n^3 \log g_0)$. □

4. LOWER BOUND

In this section we show a lower bound matching the upper bound presented in the previous section up to the logarithmic factor. More precisely, for any $n \in \mathbb{N}$ we define a graph G_n on n vertices for which the algorithm has an optimization time of at least $\Omega(n^3)$ with high probability.

The worst case graph G_n is a weighted complete graph (V, E) with $V = \{v_0, \ldots, v_{n-1}\}$, $E = V \times V \setminus \{(v, v) \mid v \in V\}$, and edge weights $w(v_i, v_j), i, j \in [0..n-1], i \neq j$ defined by

$$w(v_i, v_j) := \begin{cases} 2, & \text{if } 0 \leq i, j < n,\ j = i+1, \\ 2j+1, & \text{if } i = 0,\ 2 \leq j < n, \\ w_{\max}, & \text{otherwise.} \end{cases}$$

We choose $w_{\max} = 4n$. This implies that $(v_0, v_1, v_2, \ldots, v_{n-2}, v_{n-1})$ is the unique shortest path tree starting from $s = v_0$. Figure 3 illustrates the graph. We prove the following theorem.

Theorem 4. *The optimization time of the $(1+1)$-EA on G_n is $\Omega(n^3)$ with probability $1 - e^{-\Omega(n)}$.*

To prove this lower bound, we need the following Chernoff-type inequalities.

Theorem 5. *Let $X_i, i \in [1..n]$, be independent random variables $X := \sum_{i=1}^n X_i$. Let $0 < p < 1$ and $\delta > 0$.*

a) If $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$ for all $i \in [1..n]$, then

$$\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq \exp\left(-\frac{\delta^2 \mathbb{E}[X]}{2}\right),$$

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq \exp\left(-\frac{\min\{\delta, \delta^2\}\mathbb{E}[X]}{3}\right).$$

b) If the X_i are geometrically distributed random variables with $\Pr[X_i = j] = (1-p)^{j-1}p$ for all $j \in \mathbb{N}$, then

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq \exp\left(-\frac{\delta^2}{2}\frac{(n-1)}{1+\delta}\right).$$

Proof. Part a) is a classical Chernoff bound, cf. [1]. To prove part b), let $Y_1, Y_2, \ldots$ be an infinite sequence of independent, identically distributed biased coin tosses (binary random variables) such that Y_i is one with probability $\Pr[Y_i = 1] = p$ and zero with probability $\Pr[Y_i = 0] = 1 - p$. Note that the random variable "smallest j such that $Y_j = 1$" has the same distribution as each X_i. In consequence, X has the same distribution as "smallest j such that exactly n of the variables $Y_1, \ldots, Y_j$ are one". In particular, $\Pr[X \geq j] = \Pr[\sum_{i=1}^{j-1} Y_i \leq n-1]$ for all $j \in \mathbb{N}$. This manipulation reduces our problem to the analysis of independent Bernoulli trials and will enable us to use the classical Chernoff bounds.

The expected value of each X_i is $\mathbb{E}[X_i] = \frac{1}{p}$, thus $\mathbb{E}[X] = \frac{n}{p}$. Let $Y := \sum_{i=1}^{\lceil (1+\delta)\mathbb{E}[X]-1 \rceil} Y_i$. By the above,

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] = \Pr[Y \leq n-1].$$

The expected value of Y is bounded by

$$\mathbb{E}[Y] = \lceil (1+\delta)\mathbb{E}[X] - 1 \rceil p \geq (1+\delta)n - p > (1+\delta)(n-1).$$

Now let $\delta' := 1 - \frac{n-1}{\mathbb{E}[Y]}$. Then $0 < \delta' \leq 1$ and $\Pr[Y \leq n-1] = \Pr[Y \leq (1-\delta')\mathbb{E}[Y]]$. Hence we can apply the classical Chernoff bound from part a) to get

$$\begin{aligned}
&\Pr[X \geq (1+\delta)\mathbb{E}[X]] \\
&= \Pr[Y \leq (1-\delta')\mathbb{E}[Y]] \\
&\leq \exp\left(-\frac{1}{2}\mathbb{E}[Y]\left(1 - \frac{n-1}{\mathbb{E}[Y]}\right)^2\right) \\
&\leq \exp\left(-\frac{1}{2}\mathbb{E}[Y]\left(1 - \frac{1}{1+\delta}\right)^2\right) \\
&\leq \exp\left(-\frac{1}{2}(n-1)(1+\delta)\left(\frac{\delta}{1+\delta}\right)^2\right). \qquad \square
\end{aligned}$$

As the last preparation to prove Theorem 4, we show the following lemma. We use $\mathbf{u}$ to denote an individual. The predecessor of a vertex v is then called $\mathbf{u}(v)$.

LEMMA 6. *Let $\mathbf{u}$ be an individual, $v \neq s$ be a vertex and $y \in V \setminus \{v, \mathbf{u}(v)\}$. Let $\mathbf{u}'$ be the outcome of applying a mutation step (without selection) to $\mathbf{u}$. Then $\Pr[\mathbf{u}'(v) = y] \leq 2e(n-1)^{-2}$.*

Let $v_{i_1}, \ldots, v_{i_k} \in V \setminus \{s\}$ pairwise different and $y_1, \ldots, y_k \in V$ such that $y_j \notin \{v_{i_j}, \mathbf{u}(v_{i_j})\}$ for all $j \in [1..k]$. Then $\Pr[\forall j \in [1..k] \colon \mathbf{u}'(v_{i_j}) = y_j] \leq 2e(n-1)^{-k-1}$.

Proof. To have $\mathbf{u}'(v) \neq \mathbf{u}(v)$, at least one of the $S+1$ elementary mutations performed in the mutation step has to regard the vertex v ("first event"), and to have $\mathbf{u}'(v) = y$, the last one of these elementary mutations has to change $\mathbf{u}(v)$ to y ("second event"). The probability of the first event is at most

$$\begin{aligned}
\sum_{S=0}^{\infty} \frac{1}{S!}\frac{S+1}{n-1} &= 1/(n-1)\left(\sum_{S=1}^{\infty}\frac{S}{S!} + \sum_{S=0}^{\infty}\frac{1}{S!}\right) \\
&= 1/(n-1)\left(\sum_{S=1}^{\infty}\frac{1}{(S-1)!} + e\right) \\
&= 2e/(n-1).
\end{aligned}$$

Conditional on the first event, the second happens with probability exactly $1/(n-1)$.

For the second claim, note that the above shows that $\Pr[\mathbf{u}'(v_{i_1}) = y_1] \leq 2e(n-1)^{-2}$. Even assuming that v_{i_j}, $j \leq 2$, is touched by the mutation, the probability that the last change of its predecessor is to y_j, is exactly $(n-1)^{-1}$, and this event is independent of all other random decisions. $\square$

Clearly, the second part of the lemma is not best possible—the probability should be of order n^{-2k}—but sufficient for our purposes. We can now prove Theorem 4.

Proof of Theorem 4. As we are only interested in asymptotic bounds, implicit we assume that n is sufficiently large. Also, we will not try to find the best possible constants. To prove the claim, we analyze how long it takes until the individual $\mathbf{u}$ for the first time is the path $P := (s = v_0, v_1, \ldots, v_{n-1})$. To this aim, we analyze how the length $L = L(\mathbf{u})$ of the longest subpath of P starting in s that is contained in $\mathbf{u}$ grows. Note that, contrary to the multi-objective setting, this length L may decrease. We shall adopt the proof of [4] for the multi-objective setting to deal with this issue. In particular, we denote by L_t the maximum length ℓ such that the path $(v_0, \ldots, v_\ell)$ was contained in the individual at some time $t' \leq t$.

We first convince ourselves that for all times t and all $i > L_t + 1$, we have $\Pr[\mathbf{u}(v_i) = v_{i-1}] \leq \nu := 2e^2/(2e^2+1) \approx 0.936$. This is clearly true for $t = 0$, since the initial individual satisfies $\Pr[\mathbf{u}(v_i) = v] = 1/(n-1)$ for all $v \neq v_i$.

Assume that the claim is correct for some $t \geq 0$. Let $i > L_t + 1$. Fix a mutation chosen by the algorithm. We may assume $i > L_{t+1} + 1$ (otherwise there is nothing to show). Denote by $\mathbf{u}'$ the individual resulting from this iteration, that is, from applying the mutation to $\mathbf{u}$ in case this does not worsen the fitness. With the help of Lemma 6 we compute

$$\begin{aligned}
&\Pr[\mathbf{u}'(v_i) = v_{i-1}] \\
&= \Pr[\mathbf{u}'(v_i) = v_{i-1} \mid \mathbf{u}(v_i) = v_{i-1}]\Pr[\mathbf{u}(v_i) = v_{i-1}] \\
&\quad + \Pr[\mathbf{u}'(v_i) = v_{i-1} \mid \mathbf{u}(v_i) \neq v_{i-1}]\Pr[\mathbf{u}(v_i) \neq v_{i-1}] \\
&\leq (1 - (1/e)(n-1)^{-2})\Pr[\mathbf{u}(v_i) = v_{i-1}] \\
&\quad + 2e(n-1)^{-2}(1 - \Pr[\mathbf{u}(v_i) = v_{i-1}]) \\
&\leq \nu(1 - (1/e)(n-1)^{-2} - 2e(n-1)^{-2}) + 2e(n-1)^{-2} \\
&= \nu.
\end{aligned}$$

Note that the above holds independent of the values of $\mathbf{u}(v)$, $v \in V \setminus \{s, v_i\}$ (of course, still assuming $i > L_t + 1$). For this reason, the claim $\Pr[\mathbf{u}(v_i) = v_{i-1}] \leq \nu$ also holds if we condition on arbitrary values for these $\mathbf{u}(v)$. In consequence, for any $k \geq 1$, $i_1, \ldots, i_k$ pairwise distinct element of $[L_t + 1..n-1]$, we have

$$\begin{aligned}
&\Pr[\forall j \in [k] \colon \mathbf{u}'(v_{i_j}) = v_{i_j - 1}] \\
&= \Pr[\mathbf{u}'(v_{i_1}) = v_{i_1 - 1}] \\
&\quad \Pr[\mathbf{u}'(v_{i_2}) = v_{i_2 - 1} \mid \mathbf{u}'(v_{i_1}) = v_{i_1 - 1}] \ldots \\
&\quad \Pr[\mathbf{u}'(v_{i_k}) = v_{i_k - 1} \mid \mathbf{u}'(v_{i_1}) = v_{i_1 - 1}, \ldots, \\
&\qquad\qquad \mathbf{u}'(v_{i_{k-1}}) = v_{i_{k-1} - 1}] \\
&\leq (2e/(2e+1))^k.
\end{aligned}$$

We start our analysis of the growth of L_t by noting that with high probability L_0 is constant. More precisely, the probability that in the initial individual some vertex $v_i \in \{v_1, \ldots, v_{n-1}\}$ is already linked to v_{i-1}, is exactly $\frac{1}{n-1}$ independent of all other values of $\mathbf{u}$. Hence the probability that $L_0 \geq k$, is $(n-1)^{-k}$ for all $k \in \mathbb{N}$.

We now analyze the growth $D_t := L_t - L_{t-1}$ of the path in iteration t. For D_t to be positive, the mutation has to change $\mathbf{u}(v_{L_t+1})$ to v_{L_t}. By Lemma 6, this happens with probability at most $2e(n-1)^{-2}$. D_t can be larger than one due to two effects: (i) the mutation can change $\mathbf{u}(v_j)$ to v_{j-1} for further vertices on the path, and (ii), some vertices v_j not touched at all by the mutation can already be connected to v_{j-1}. Again by Lemma 6, the probability of an event of type (i) is at most $2e(n-1)^{-2}$ independently of all other random decisions now or in the past. By the above reasoning, the probability of an event of type (ii), i. e., that certain k vertices not touched by the mutation in this iteration were already connected with their natural predecessor, is at most ν^k.

Hence $\Pr[D_t \geq k] \leq \frac{2e}{(n-1)^2}(2e(n-1)^{-2} + \nu)^{k-1} \leq 2en^{-2}\,(0.95)^{k-1}$ for n sufficiently large. For all t, D_t is dominated by a random variable $\overline{D}$ with distribution $\Pr[\overline{D} = k] = 2en^{-2}\,(0.95)^{k-1}0.05$ and $\Pr[\overline{D} = 0] = 1 - 2en^{-2}$ as this implies $\Pr[\overline{D} \geq k] = 2en^{-2}\,(0.95)^{k-1}$. Similarly, we see that $L_0 - 1$ is dominated by a random variable with distribution $\overline{D}$. Hence $L_t - 1 = L_0 - 1 + \sum_{k=1}^{t} D_k$ is dominated by the sum of t independent random variables with distribution $\overline{D}$.

The probability that after $t = (2e)(1/80)(n-2)n^2$ steps the optimal solution is found is $\Pr[L_t = n-1]$. This is at most $\Pr[\sum_{i=1}^{t} X_i \geq n-2]$ by the above considerations, where the X_i are independent random variables with distribution $\overline{D}$. Denote by x the number of X_i that are positive. Then $\mathbb{E}[x] = 2en^{-2}t = (1/80)(n-2)$ and thus $\Pr[x > 4en^{-2}t] \leq \exp(-\mathbb{E}(x)/12) = \exp(-(n-2)/960)$. We compute

$$\begin{aligned}
&\Pr[L_t = n-1] \\
&\leq \Pr\left[\sum_{i=1}^{t} X_i \geq n-2\right] \\
&\leq \Pr\left[\sum_{i=1}^{t} X_i \geq n-2 \mid x \leq (1/40)(n-2)\right] \\
&\quad + \Pr[x > 2 \cdot 2en^{-2}t] \\
&\leq \Pr\left[\sum_{i=1}^{t} X_i \geq n-2 \mid x = (1/40)(n-2)\right] \\
&\quad + \exp(-(n-2)/960).
\end{aligned}$$

For all i, define the random variable $Y_i = (X_i \mid X_i \geq 1)$. Then Y_i has the geometric distribution $\Pr[Y_i = k] = \Pr[X_i = k]/\Pr[X_i \geq 1] = (0.95)^{k-1}\,0.05$ and $\mathbb{E}[Y_i] = 20$. Hence by Theorem 5, we have

$$\begin{aligned}
&\Pr\left[\sum_{i=1}^{t} X_i \geq n-2 \mid x = (1/40)(n-2)\right] \\
&= \Pr\left[\sum_{i=1}^{(1/40)(n-2)} Y_i \geq n-2\right] \\
&\leq \Pr\left[\sum_{i=1}^{(1/40)(n-2)} Y_i \geq 2 \cdot 20 \cdot (1/40)(n-2)\right] \\
&\leq \exp(-((1/40)(n-2)-1)/4).
\end{aligned}$$

This shows that

$$\begin{aligned}
&\Pr[L_t = n-1] \\
&\leq \exp(-((1/40)(n-2)-1)/4) + \exp(-(n-2)/960) \\
&= \exp(-\Omega(n)).
\end{aligned}$$

□

5. CONCLUSION

The single source shortest path problem is one of the fundamental problems in computer science and the first combinatorial optimization problem for which a rigorous runtime analysis of evolutionary algorithms has been carried out. We have shown that a multi-objective approach is not necessary to solve this problem efficiently by evolutionary algorithms and analyzed the single-objective one given in [17]. The upper bound obtained is similar to the multi- objective result. Additionally, our analyses give new insights how evolutionary algorithms may achieve progress towards an optimal solution even if the proof ideas can not follow the run of Dijkstra's algorithm.

References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, 2nd edition, 2000.

[2] H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, volume 1, pages 269–271. Mathematisch Centrum, Amsterdam, The Netherlands, 1959.

[4] B. Doerr, E. Happ, and C. Klein. A tight bound for the (1+1)-EA on the single source shortest path problem. In *Proc. of the IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 1890–1895, 2007.

[5] B. Doerr, T. Jansen, and C. Klein. Comparing global and local mutations on bit strings. In *Proc. of the 10th annual conference on Genetic and evolutionary computation (GECCO '08)*, pages 929–936, 2008.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[7] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1–2):51–81, 2002.

[8] A. El-Fallahi, C. Prins, and R. W. Calvo. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & OR*, 35(5):1725–1741, 2008.

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

[10] S. J. Kim and M. K. Choi. Evolutionary algorithms for route selection and rate allocation in multirate multicast networks. *Appl. Intell.*, 26(3):197–215, 2007.

[11] S. Knopp, P. Sanders, D. Schultes, F. Schulz, and D. Wagner. Computing many-to-many shortest paths using highway hierarchies. In *Proc. of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX '07)*, 2007.

[12] Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In *Evolutionary Programming*, pages 135–155, 1995.

[13] H. Mühlenbein. How genetic algorithms really work: mutation and hillclimbing. In *Proc. 2nd International Conference Parallel Problem Solving from Nature (PPSN II)*, pages 15–26, 1992.

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

[15] G. Rudolph. How mutation and selection solve long path problems in polynomial expected time. *Evolutionary Computation*, 4(2):195–205, 1996.

[16] P. Sanders and D. Schultes. Engineering highway hierarchies. In *Proc. of the 14th Annual European Symposium on Algorithms (ESA '06)*, pages 804–816, 2006.

[17] J. Scharnow, K. Tinnefeld, and I. Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, 3(4):349–366, 2004. (Conference version appeared in Proc. 7th International Conference Parallel Problem Solving from Nature 2002 (PPSN VII)).

[18] C. Witt. 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)*, pages 44–56, 2005.