TECHNISCHE UNIVERSITÄT DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

# Analysis of a Simple Evolutionary Algorithm for the Multiobjective Shortest Path Problem

Christian Horoba

No. CI-258/08

Technical Report ISSN 1433-3325 December 2008

Secretary of the SFB 531 · Technische Universität Dortmund · Dept. of Computer Science/LS 2 · 44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence," at the Technische Universität Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

# Analysis of a Simple Evolutionary Algorithm for the Multiobjective Shortest Path Problem

Christian Horoba*
Fakultät für Informatik, LS 2
Technische Universität Dortmund
Dortmund, Germany

December 12, 2008

**Abstract**

We present a natural fitness function $f$ for the multiobjective shortest path problem, which is a fundamental multiobjective combinatorial optimization problem known to be NP-hard. Thereafter, we conduct a rigorous runtime analysis of a simple evolutionary algorithm (EA) optimizing $f$. Interestingly, this simple general algorithm is a fully polynomial-time randomized approximation scheme (FPRAS) for the problem under consideration, which exemplifies how EAs are able to find good approximate solutions for hard problems.

## 1 Introduction

In the last decade, a lot of effort has been put into the runtime analysis of evolutionary algorithms (EAs). These results are aiming at a well-founded understanding of the working principles of EAs. One of the first papers of this line of research has investigated the runtime of a simple EA, the $(1+1)$ EA, on simple pseudo-Boolean functions [6]. Later, there have been similar investigations of basic EAs for multiobjective optimization [11]. The first runtime analyses have mainly dealt with artificial example functions. In the last few years, the analysis of EAs for classical algorithmic problems has attracted more and more attention since EAs are often applied to solve combinatorial optimization problems.

One of the first papers that deals with this subject is [19] whose authors analyze the runtime of EAs for sorting and the single-source shortest path problem. The topic of [8] is the maximum matching problem. There is a series of papers analyzing EAs for the Eulerian cycle problem [14, 3, 5, 4]

*This author was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

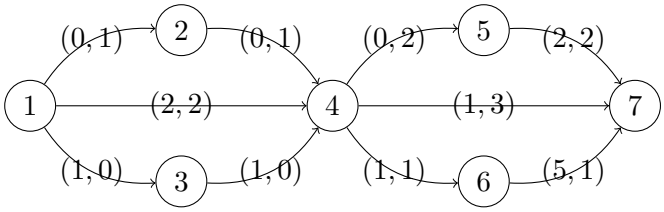


Figure 1: Example of a weighted directed graph with 2 weights per edge.

| Path | Weight |
|---|---|
| $1 \to 2 \to 4 \to 5 \to 7$ | $(2, 6)$ |
| $\mathbf{1 \to 2 \to 4 \to 7}$ | $\mathbf{(1, 5)}$ |
| $1 \to 2 \to 4 \to 6 \to 7$ | $(6, 4)$ |
| $1 \to 4 \to 5 \to 7$ | $(4, 6)$ |
| $1 \to 4 \to 7$ | $(3, 5)$ |
| $1 \to 4 \to 6 \to 7$ | $(8, 4)$ |
| $1 \to 3 \to 4 \to 5 \to 7$ | $(4, 4)$ |
| $\mathbf{1 \to 3 \to 4 \to 7}$ | $\mathbf{(3, 3)}$ |
| $\mathbf{1 \to 3 \to 4 \to 6 \to 7}$ | $\mathbf{(8, 2)}$ |

Table 1: Overview of all paths from node 1 to node 7 with the corresponding weights. The *Pareto-optimal* or *efficient* paths are shown in bold.

and the minimum spanning tree problem [16, 15]. All mentioned problems have in common that they are solvable in polynomial time. Since EAs are typically applied to find good solution for (NP-)hard problems, it would be even more interesting to find out more about the runtime on such problems. In [13] it is shown that a simple EA efficiently approximates the NP-hard biobjective minimum spanning tree problem by a constant factor of 2 and in [22] it is attested that a simple EA efficiently approximates the NP-hard partition problem by a constant factor of 4/3. While these results point out the capability of EAs to find good approximate solutions, it would be more appropriate if the user of the EA could provide the desired approximation ratio. We have named only a few results and refer to the comprehensive survey [17] for further references on this topic.

This paper is dedicated to the well-known multiobjective shortest path problem, which consists in finding a set of shortest paths that captures all possible trade-offs in a weighted directed graph with $k \geq 1$ weight functions. Figure 1 shows a possible input instance for the problem at hand and Table 1 shows all (Pareto-optimal) paths from node 1 to node 7. Before we consider EAs, we summarize some general results concerning the problem. The problem is known to be NP-hard if we are confronted with two or more weights per edge [7]. Furthermore, there are instances with an exponential number of Pareto-optimal paths. Hence, approximative solutions seem to

be the only resort. The first fully polynomial-time approximation scheme (FPTAS) for the biobjective case is presented in [9]. If we restrict ourselves to acyclic directed graphs, an FPTAS for the general multiobjective case is presented in [21]. In [18] it is shown that there is for each multiobjective optimization problem a succinct approximation of the Pareto front of polynomial size. Furthermore, the authors show how to construct for several multiobjective optimization problems (including the shortest path problem) an FPTAS. An FPTAS with an improved runtime for the single-source multiobjective shortest path problem is presented in [20]. The last-mentioned paper [20] has actually motivated the runtime analysis of a simple EA for this problem since such algorithms can "simulate" the FPTAS, which relies on dynamic programming in combination with scaling and rounding.

All available papers, which investigate the runtime of EAs for the shortest path problem from a theoretical point of view, have concentrated on the singleobjective case. The above-mentioned paper [19] has analyzed the runtime of a simple EA for the single-source variant. The authors of [1] have refined this analysis by classifying the input instances according to $\ell$ where $\ell$ is the smallest integer such that any vertex can be reached from the source via a shortest path with at most $\ell$ edges. The all-pairs variant has been presented in [2] as a natural example where the use of an appropriate recombination operator can improve the runtime of an EA.

This paper provides the first runtime analysis of a simple EA for the multiobjective shortest path problem. The analysis shows that the runtime of the investigated EA is competitive with the best known problem-specific algorithm [20].

We conclude this section with a short overview of the paper. In Section 2 we give a brief introduction into multiobjective optimization and explain how to measure the runtime of an EA. Section 3 is dedicated to the multiobjective shortest path problem. The fitness function and the basic EA, which we will use in this paper, are presented in Section 4 and Section 5, respectively. In Section 6 we analyze the runtime of the EA on the fitness function. We state the used proof technique in Section 7 for possible future applications and conclude the paper with a discussion in Section 8.

## 2 Preliminaries

The goal in multiobjective optimization is to optimize several objectives simultaneously. The different objectives are often conflicting, which implies that there is no single optimal solution but a set of solutions, which represents the possible trade-offs w. r. t. the objectives. Within this paper we consider the minimization of functions $g\colon S \to \mathbb{R}^d$ where $d \in \mathbb{N}^+$ is the number of objectives.

It is common to call $g$ *objective function*, $S$ *search space*, and $\mathbb{R}^d$ *objective*

*space*. The elements of $S$ and $\mathbb{R}^d$ are called *search points* and *objective vectors*, respectively. We define that $y'$ *weakly dominates* $y$ ($y' \succeq y$) iff $y'_i \leq y_i$ for all $i \in \{1, \ldots, d\}$, and $y'$ *dominates* $y$ ($y' \succ y$) iff $y' \succeq y$ and $y \not\succeq y'$ where $y = (y_1, \ldots, y_d) \in \mathbb{R}^d$ and $y' = (y'_1, \ldots, y'_d) \in \mathbb{R}^d$. The concept of dominance is generalized via $Y' \succeq Y :\Leftrightarrow \forall y \in Y \colon \exists y' \in Y' \colon y' \succeq y$ to subsets $Y, Y' \subseteq \mathbb{R}^d$ of the objective space. The set $\mathcal{PF}(g) = \{y \in g(S) \mid \nexists y' \in g(S) \colon y' \succ y\}$ is called the *Pareto front of* $g$ and a set $\mathcal{P}(g) \subseteq S$ is called a *Pareto set of* $g$ iff $g(\mathcal{P}(g)) = \mathcal{PF}(g)$. The elements of $\mathcal{PF}(g)$ and $g^{-1}(\mathcal{PF}(g))$ are called *Pareto-optimal*. The goal is to determine an arbitrary Pareto set of $g$. The definition of a Pareto set mirrors the assumption that search points that are mapped to the same objective vector are considered as equivalent. Hence, it is sufficient to determine for all Pareto-optimal objective vectors $y \in \mathcal{PF}(g)$ at least one search point $s \in S$ with $g(s) = y$.

It is known that most multiobjective optimization problems might feature a Pareto front of exponential size w. r. t. the input size. Therefore, a much better goal is to look for approximations of the Pareto front. We use the following measure for the quality of an approximation where we assume that all function values are positive to allow for a meaningful definition of relative approximation ratios. We define that $y'$ $(1+\varepsilon)$*-dominates* $y$ ($y' \succeq_{1+\varepsilon} y$) iff $y'_i \leq (1+\varepsilon) \cdot y_i$ for all $i \in \{1, \ldots, d\}$ where $y = (y_1, \ldots, y_d) \in (\mathbb{R}^+)^d$, $y' = (y'_1, \ldots, y'_d) \in (\mathbb{R}^+)^d$, and $\varepsilon \in \mathbb{R}^+$. A set $\mathcal{PF}_\varepsilon(g) \subseteq g(S)$ is called *$\varepsilon$-approximate Pareto front of* $g$ iff

$$\forall y \in g(S) \colon \quad \exists y' \in \mathcal{PF}_\varepsilon(g) \colon \quad y' \succeq_{1+\varepsilon} y,$$

and a set $\mathcal{P}_\varepsilon(g) \subseteq S$ is called *$\varepsilon$-approximate Pareto set of* $g$ iff $g(\mathcal{P}_\varepsilon(g))$ is an $\varepsilon$-approximate Pareto front of $g$.

The next point, which we would like to clarify, is the measurement of the runtime of an EA. EAs are randomized search heuristics that are based on a population (multiset) of individuals where each individual is a representation of a search point. Different randomized variation operators (e. g., recombination or mutation) are utilized to create new individuals, which emanate from the individuals in the actual population. The subsequent selection operator composes the next population where the transition is called a generation (round). For theoretical investigations, we consider a run of an EA as an infinite sequence of populations and count the number of generations until the population fulfills the desired optimization goal for the first time. This number is called the *optimization time* of the considered EA. The expected optimization time refers to the expectation of this random variable. In this paper, we are interested in the number of generations until the population has evolved into an $\varepsilon$-approximate Pareto set.

We conclude this section with the definitions of a *fully polynomial-time approximation scheme* (FPTAS) and a *fully polynomial-time randomized approximation scheme* (FPRAS) for multiobjective optimization problems.

An algorithm is called FPTAS (FPRAS) for a multiobjective optimization problem if the algorithm

- runs in polynomial time w. r. t. the size of the input and $1/\varepsilon$, and
- outputs an $\varepsilon$-approximate Pareto set with probability 1 (at least 3/4)

for any input instance and any $\varepsilon \in \mathbb{R}^+$.

# 3 Problem

The multiobjective shortest path problem is presented in this section. We are given a directed graph $G = (V, E)$ where $V$ is a set of $n$ vertices and $E \subseteq \{(u, v) \in V^2 \mid v \neq u\}$ is a set of $m$ edges. Each edge $e \in E$ is associated with $k \in \mathbb{N}^+$ positive weights $w_1(e), \ldots, w_k(e) \in \mathbb{R}^+$, i. e., we are given a weight function $w\colon E \to (\mathbb{R}^+)^k$ with $w(e) = (w_1(e), \ldots, w_k(e))$. A *(simple) path $p$ of length $\ell \in \mathbb{N}$* is a sequence $p = (v_0, \ldots, v_\ell)$ of $\ell + 1$ (different) vertices where $(v_{i-1}, v_i) \in E$ for all $1 \leq i \leq \ell$. The vertices $v_0$ and $v_\ell$ are called *source* and *sink*, respectively. Note that the length of a simple path is at most $n - 1$. The *weight $w(p)$ of a path $p = (v_0, \ldots, v_\ell)$* is the sum of the weights of the traversed edges, i. e., $w(p) = \sum_{i=1}^{\ell} w((v_{i-1}, v_i))$. We use the notation $\mathrm{PATH}_\ell(s, t)$ for the set of all *simple* paths of length $\ell$ from $s \in V$ to $t \in V$ and define further $\mathrm{PATH}_{\leq \ell}(s, t) := \bigcup_{0 \leq i \leq \ell} \mathrm{PATH}_i(s, t)$.

The most general shortest path problem is the all-pairs shortest path problem, i. e., we are asked to determine a Pareto set of

$$w\colon \mathrm{PATH}_{\leq n-1}(s, t) \to (\mathbb{R}^+)^k$$

for all $s, t \in V$, $t \neq s$. Note that restricting the search space to simple paths does not affect the Pareto front and the Pareto set since all Pareto-optimal paths are simple because all weights are positive. We obtain simpler versions of the problem if we fix $s$ or $t$ in advance or if we consider an undirected graph where "simpler" means that we have to calculate less trade-off paths. We will concentrate on the single-source shortest path problem for directed graphs, i. e., we are, given a source $s \in V$, interested in finding a Pareto set of $w\colon \mathrm{PATH}_{\leq n-1}(s, t) \to (\mathbb{R}^+)^k$ for all $t \in V$, $t \neq s$. Recall that a Pareto set of $w$ is a set of paths from $s$ to $t$ that contains for all Pareto-optimal objective vectors $y \in w(\mathrm{PATH}_{\leq n-1}(s, t))$ at least one path $p$ with $w(p) = y$.

We will use the following abbreviations and assumptions to simplify the notation in the subsequent sections. Let $w_i^{\min} = \min_{e \in E} w_i(e)$ and $w_i^{\max} = \max_{e \in E} w_i(e)$ denote the extreme weights w. r. t. the $i$-th dimension. Further, we denote the extreme weights w. r. t. all dimensions by $w^{\min} = \min_{1 \leq i \leq k} w_i^{\min}$ and $w^{\max} = \max_{1 \leq i \leq k} w_i^{\max}$. We assume w. l. o. g. that $V = \{1, \ldots, n\}$, $s = 1$, and $w^{\min} \geq 1$. The first assumptions can be achieved by renaming the vertices and the last assumption can be achieved by dividing all weights by $w^{\min}$.

## 4 Fitness Function

If we want to use an EA to solve a problem, the first task is to find an appropriate representation of the solution candidates. We use

$$\mathrm{PATH}_{\leq n-1}(1,\cdot) := \bigcup_{1\leq t\leq n} \mathrm{PATH}_{\leq n-1}(1,t)$$

as search space. Note that $\mathrm{PATH}_{\leq n-1}(1,1) = \{(1)\}$.

The second task is to find an appropriate objective function. Consider two arbitrary individuals $p = (v_0, \ldots, v_\ell)$ and $p' = (v'_0, \ldots, v'_{\ell'})$. Since we are interested in shortest paths from 1 to all $2 \leq t \leq n$, the objective function has to ensure that $p$ and $p'$ are incomparable if $v'_{\ell'} \neq v_\ell$. The weight $w = (w_i)$ does therefore not serve as an appropriate objective function. Hence, we define the objective function $f = (f_{t,i})$ with

$$f_{t,i}(p) := \begin{cases} \sum_{j=1}^{\ell} w_i((v_{j-1}, v_j)) & \text{if } v_\ell = t \\ c \cdot (n-1) \cdot w^{\max} & \text{otherwise} \end{cases}$$

for all $2 \leq t \leq n$ and $1 \leq i \leq k$ where $c > 1$. Note that we have decided for a problem formulation, which involves the optimization of

$$d = (n-1) \cdot k$$

different objectives. The factor $c$ guarantees that $w_i(p) < c \cdot (n-1) \cdot w^{\max}$ for all $p$. The definition of the objective function ensures that $1 \leq f_{t,i}(p) \leq (n-1) \cdot w^{\max}$ for $t = v_\ell$ and $f_{t,i}(p) = c \cdot (n-1) \cdot w^{\max}$ for $t \neq v_\ell$. Hence, $p$ and $p'$ are incomparable if $v'_{\ell'} \neq v_\ell$.

Another reasonable approach consists in using the $k$-dimensional objective function $w = (w_i)$ and adapting the EA to consider paths with different sinks as incomparable. The behavior of the EA is in both cases the same. We decide for the first alternative, which relies on a more general EA and demonstrates how additional information can be introduced into the objective function.

## 5 Evolutionary Algorithm

We motivate and present in this section all components of the EA, which we analyze in the next section.

We decide to utilize a natural initialization of the population and to start off with all paths of length 1, as it is proposed in [2], i. e., $P := \{(u,v) \in E \mid u = 1\}$.

The question of how to choose the variation operators is much more delicate. We decide on a simple variant by restricting the algorithm to a single mutation operator. In each generation we choose an individual uniformly at

random for mutation. It is a classical guideline that the application of a mutation operator should result in most cases in a small change of the selected individual. On the other hand, it should be possible to create any other individual in a single mutation step with positive probability to guarantee convergence. For the classical case of bit-strings of length $n$, the standard bit-flip mutation operator implements these guidelines by independently flipping each bit with probability $1/n$. It is obvious that this mutation operator is not applicable to our representation. Therefore, we decide to simulate this behavior. Flipping a single bit corresponds in our case to a minimal modification of the path. We decide similar to [2] to remove the last vertex of the path or append a vertex to the path. The following case distinction makes the probabilities for these elementary mutation steps precise. If we want to mutate a path $p = (v_0, \ldots, v_\ell)$, we choose an alternative $b \in \{0, 1\}$ uniformly at random and create the mutated path $p'$ as follows.

- If $b = 0$ and $\ell \geq 1$, set $p' = (v_0, \ldots, v_{\ell-1})$.
- If $b = 1$ and $\{v \in V \mid (v_\ell, v) \in E\} \neq \emptyset$, choose $v' \in \{v \in V \mid (v_\ell, v) \in E\}$ uniformly at random and set $p' = (v_0, \ldots, v_\ell, v')$.
- Otherwise, set $p' = p$.

Afterwards we remove all cycles from $p'$ to ensure that the mutated path is simple. To simulate the global behavior of the above-mentioned mutation operator, we have to allow multiple elementary mutation steps per mutation. The number of flipping bits for the standard bit-flip mutation operator is distributed according to the binomial distribution where $f(i; n, p) = \binom{n}{i} p^i (1-p)^{n-i}$ is the probability mass function for the binomial distribution for $n \in \mathbb{N}$, $0 \leq p \leq 1$, and $0 \leq i \leq n$. It is known that this distribution converges to the Poisson distribution with $\lambda = n \cdot p$ as $n$ converges to infinity where $f(i; \lambda) = \lambda^i e^{-\lambda}/i!$ is the probability mass function for the Poisson distribution for $\lambda > 0$ and $i \in \mathbb{N}$. Therefore, it is natural to use the Poisson distribution with $\lambda = 1$ to determine the number of elementary mutation steps to be applied, when simulating the standard bit-flip mutation operator. The same distribution has been used in [19, 1, 2]. The proposed mutation operator is almost the same as the one used in [2]. However, it differs from the mutation operator used in [19, 1], which modifies individuals $(v_2, \ldots, v_n) \in \{1, \ldots, n\}^{n-1}$, $v_i \neq i$ where $v_i$ is interpreted as the predecessor of $i$. Above representation is due to the fact that in the singleobjective case all shortest paths form a tree with root 1, which does not transfer to the general case.

How to implement the environmental selection? The simplest approach is to store all nondominated individuals in the population. A drawback of this approach is that the population size might become large although all individuals concentrate on a certain part of the Pareto front, which would

**Algorithm 1** DEMO($r$) optimizing $g$

set $P \leftarrow \{(u, v) \in E \mid u = 1\}$
**loop**
    choose an individual $p \in P$ uniformly at random
    create a mutant $p'$ of $p$
    **if** $\nexists p'' \in P\colon g(p'') \succ g(p') \vee b_r(g(p'')) \succ b_r(g(p'))$ **then**
        set $P \leftarrow (P \setminus \{p'' \in P \mid b_r(g(p')) \succeq b_r(g(p''))\}) \cup \{p'\}$
    **end if**
**end loop**

lead to a poor overall approximation. Hence, controlling the population size and achieving a diverse population are important issues. We resort to dividing the objective space into boxes and storing in the population at most one individual from each box, as it has been proposed in [10]. Therefore, we define the so called *box index*.

**Definition 1.** *Let $r > 1$. The* box index *of an $y \in \mathbb{R}^+$ is defined as*

$$b_r(y) := \lfloor \log_r(y) \rfloor = \lfloor \log(y)/\log(r) \rfloor$$

We generalize the last definition to objective vectors $(y_1, \ldots, y_d) \in (\mathbb{R}^+)^d$ by $b_r((y_1, \ldots, y_d)) := (b_r(y_1), \ldots, b_r(y_d))$ and subsets $Y \subseteq (\mathbb{R}^+)^d$ of the objective space by $b_r(Y) := \{b_r(y) \mid y \in Y\}$. A new individual $p'$ is accepted iff there is no $p'' \in P$ with $g(p'') \succ g(p')$ or $b_r(g(p'')) \succ b_r(g(p'))$. If a new individual is accepted, all individuals $p'' \in P$ with $b_r(g(p')) \succeq b_r(g(p''))$ are removed from $P$ and $p'$ is added to $P$. The last step ensures that one individual from each nondominated box is stored in the population.

It remains to assemble the presented components. The resulting algorithm that is called *Diversity maintaining Evolutionary Multiobjective Optimizer* (DEMO($r$)) is presented in Algorithm 1. The single parameter of the algorithm is the ratio $r > 1$, which determines the size of the boxes.

We conclude this section with a simple upper bound on the population size of DEMO($r$).

**Lemma 1.** *Let $r > 1$ be a ratio, $g = (g_1, \ldots, g_d)\colon S \to (\mathbb{R}^+)^d$ an objective function, and $\bigcup_{i=1}^{n} S_i$ a partition of the search space $S$. Then the population size of* DEMO($r$) *optimizing $g$ is at most*

$$\sum_{i=1}^{n} \prod_{j=1}^{d_i - 1} n_{j,i} \prod_{j=d_i+1}^{d} n_{j,i}$$

*where $n_{j,i} := |b_r(g_j(S_i))|$ and $d_i \in \{1, \ldots, d\}$.*

*Proof.* Consider $S_i$. The image of $S_i$ under $g$ consists of objective vectors from

$$|b_r(g(S_i))| \leq \prod_{j=1}^{d} |b_r(g_j(S_i))| = \prod_{j=1}^{d} n_{j,i}$$

different boxes. Since the update mechanism of DEMO($r$) ensures that the population consists of one individual from each *nondominated* box, the population contains at most $\prod_{j=1}^{d_i-1} n_{j,i} \prod_{j=d_i+1}^{d} n_{j,i}$ individuals from $S_i$. Hence, the population size is at most

$$\sum_{i=1}^{n} \prod_{j=1}^{d_i-1} n_{j,i} \prod_{j=d_i+1}^{d} n_{j,i}. \qquad \square$$

# 6 Runtime Analysis

Within this section we analyze the EA from Section 5 on the fitness function from Section 4. We start with two lemmas, which are used in the proof of the following theorem.

The next technical lemma constitutes the core of the proof of Theorem 1. It describes how an $r^i$-approximate Pareto set of $f\colon \mathrm{PATH}_{\leq i}(1,\cdot) \to (\mathbb{R}^+)^d$ can be evolved into an $r^{i+1}$-approximate Pareto set of $f\colon \mathrm{PATH}_{\leq i+1}(1,\cdot) \to (\mathbb{R}^+)^d$.

**Lemma 2.** *Let $c > 1$ and $1 < r < c^{1/(n-2)}$. Further, $1 \leq i \leq n-2$ and $P \subseteq \mathrm{PATH}_{\leq n-1}(1,\cdot)$ with $f(P) \succeq_{r^i} f(\mathrm{PATH}_{\leq i}(1,\cdot))$. Consider a path*

$$p_1 = (1, v_{1,1}, \ldots, v_{1,i}, v_{1,i+1}) \in \mathrm{PATH}_{i+1}(1,\cdot).$$

*Then there is a path $p_2 = (1, v_{2,1}, \ldots, v_{2,\ell}) \in P$ with $v_{2,\ell} = v_{1,i}$ such that*

1. *$f(\hat{p_2}) \succeq_{r^i} f(p_1)$, and*
2. *$f(p_3) \succeq_{r^{i+1}} f(p_1)$ for each path $p_3$ with $b_r(f(p_3)) \succeq b_r(f(\hat{p_2}))$*

*where $\hat{p_2} = (1, v_{2,1}, \ldots, v_{2,\ell}, v_{1,i+1})$.*

*Proof.* We prove each statement in turn.

1. Consider the subpath $\hat{p_1} := (1, v_{1,1}, \ldots, v_{1,i})$ of $p_1$. There is a path $p_2 = (1, v_{2,1}, \ldots, v_{2,\ell}) \in P$ with $f(p_2) \succeq_{r^i} f(\hat{p_1})$ since $\hat{p_1} \in \mathrm{PATH}_i(1,\cdot)$ and $f(P) \succeq_{r^i} f(\mathrm{PATH}_{\leq i}(1,\cdot))$. Due to $f(p_2) \succeq_{r^i} f(\hat{p_1})$,

   $$f_{v_{1,i},j}(p_2) \leq r^i \cdot f_{v_{1,i},j}(\hat{p_1})$$

   holds for all $1 \leq j \leq k$. Using $i \leq n-2$ and $r < c^{1/(n-2)}$, we get

   $$r^i \cdot f_{v_{1,i},j}(\hat{p_1}) \leq r^{n-2} \cdot (n-1) \cdot w^{\max} < c \cdot (n-1) \cdot w^{\max}.$$

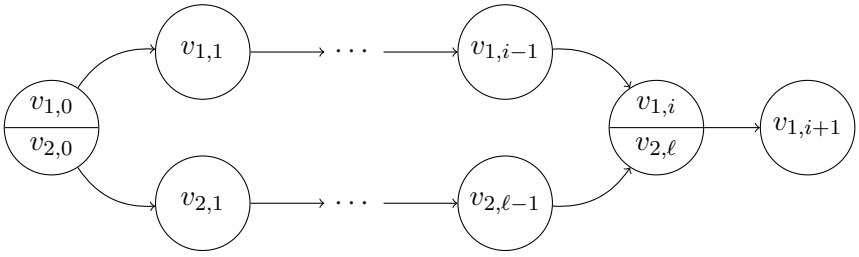


Figure 2: Relation between the paths $p_1 = (p_{1,0}, \ldots, p_{1,i+1})$ and $p_2 = (p_{2,0}, \ldots, p_{2,\ell})$.

Due to the function definition, the assumption $v_{2,\ell} \neq v_{1,i}$ would lead to $f_{v_{1,i},j}(p_2) = c \cdot (n-1) \cdot w^{\max}$, which contradicts the derived inequality $f_{v_{1,i},j}(p_2) < c \cdot (n-1) \cdot w^{\max}$. Hence, $v_{2,\ell} = v_{1,i}$. The relation between $p_1$ and $p_2$ is depicted in Figure 2. Consider the superpath $\hat{p_2} := (1, v_{2,1}, \ldots, v_{2,\ell}, v_{1,i+1})$ of $p_2$. It holds that $f(\hat{p_2}) \succeq_{r^i} f(p_1)$ since

$$\begin{aligned} f_{v_{1,i+1},j}(\hat{p_2}) &= f_{v_{2,\ell},j}(p_2) + w_j((v_{2,\ell}, v_{1,i+1})) \\ &\leq r^i \cdot f_{v_{1,i},j}(\hat{p_1}) + w_j((v_{1,i}, v_{1,i+1})) \\ &\leq r^i \cdot f_{v_{1,i+1},j}(p_1), \end{aligned}$$

for all $1 \leq j \leq k$, and

$$f_{t,j}(\hat{p_2}) = c \cdot (n-1) \cdot w^{\max} = f_{t,j}(p_1)$$

for all $2 \leq t \leq n$, $t \neq v_{1,i+1}$, and $1 \leq j \leq k$.

2. We conclude from the prerequisite $b_r(f(p_3)) \succeq b_r(f(\hat{p_2}))$ that

$$\lfloor \log_r(f_{t,j}(p_3)) \rfloor \leq \lfloor \log_r(f_{t,j}(\hat{p_2})) \rfloor.$$

The last inequality implies

$$\frac{\log(f_{t,j}(p_3))}{\log(r)} - 1 \leq \frac{\log(f_{t,j}(\hat{p_2}))}{\log(r)},$$

which is equivalent to $f_{t,j}(p_3) \leq r \cdot f_{t,j}(\hat{p_2})$. Using $f_{t,j}(\hat{p_2}) \leq r^i \cdot f_{t,j}(p_1)$, we deduce that

$$f_{t,j}(p_3) \leq r \cdot r^i \cdot f_{t,j}(p_1) = r^{i+1} \cdot f_{t,j}(p_1). \qquad \square$$

The next lemma upper bounds the population size of DEMO($r$) optimizing $f$.

**Lemma 3.** *Let $c > 1$ and $r > 1$. Then the population size of* DEMO($r$) *optimizing $f$ is at most*

$$(n-1) \cdot (\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 1)^{k-1} + 1.$$

*Proof.* Consider the partition

$$\bigcup_{j=1}^{n} \mathrm{PATH}_{\leq n-1}(1,j)$$

of the search space $\mathrm{PATH}_{\leq n-1}(1,\cdot)$. Due to Lemma 1, the population size of DEMO($r$) optimizing $f$ is at most

$$\sum_{j=1}^{n} \prod_{(t,i)|(t,i)\neq(j,1)} n_{(t,i),j}$$

where $n_{(t,i),j} = |b_r(f_{t,i}(\mathrm{PATH}_{\leq n-1}(1,j)))|$. Furthermore, it holds for all $p \in \mathrm{PATH}_{\leq n-1}(1,j)$ that

$$1 \leq f_{t,i}(p) \leq (n-1) \cdot w^{\max}$$

if $t = j$, and

$$f_{t,i}(p) = c \cdot (n-1) \cdot w^{\max}$$

if $t \neq j$. Hence, it holds for all $(t,i)$ that

$$n_{(t,i),j} \leq \lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 1$$

if $t = j$, and

$$n_{(t,i),j} = 1$$

if $t \neq j$ where the first inequality stems from

$$b_r((n-1) \cdot w^{\max}) - b_r(1) + 1 = \lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 1.$$

Hence, the population size is at most

$$1^{d-1} + (n-1) \cdot (\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 1)^{k-1} \cdot 1^{d-k}. \qquad \square$$

Note that Lemma 3 is based on the lower bound 1 and the upper bound $(n-1) \cdot w^{\max}$ on the weight of a path. The lower bound stems from the assumption that all weights are at least 1 and the upper bound stems from the restriction to *simple* paths.

We are now well-equiped to prove the main result.

**Theorem 1.** *Let* $0 < \varepsilon \leq 1$ *and* $c > (1+\varepsilon)^{(n-2)/(n-1)}$. *Then* DEMO($r$) *with* $r = (1+\varepsilon)^{1/(n-1)}$ *optimizing* $f$ *achieves an* $\varepsilon$*-approximate Pareto set within an expected number of*

$$\mathrm{O}\left(n^3 \cdot \left(\frac{n \cdot \log(n \cdot w^{\max})}{\varepsilon}\right)^{k-1} \cdot \log\left(\frac{n \cdot \log(n \cdot w^{\max})}{\varepsilon}\right)\right)$$

*generations.*

*Proof.* Let $P$ denote the actual population. We divide a run of the algorithm into $n-1$ phases where the $i$-th phase, $1 \leq i \leq n-1$, lasts until $f(P) \succeq_{r^i} f(\text{PATH}_{\leq i}(1,\cdot))$. After the $(n-1)$-th phase $f(P) \succeq_{r^{n-1}} f(\text{PATH}_{\leq n-1}(1,\cdot))$ and therefore $f(P) \succeq_{1+\varepsilon} f(\text{PATH}_{\leq n-1}(1,\cdot))$ since

$$r^{n-1} = ((1+\varepsilon)^{1/(n-1)})^{n-1} = 1+\varepsilon.$$

The initialization of the population ensures that $f(P) \succeq_{r^1} f(\text{PATH}_{\leq 1}(1,\cdot))$ because of $f(P) = f(\text{PATH}_{\leq 1}(1,\cdot))$.

Let $f(P) \succeq_{r^i} f(\text{PATH}_{\leq i}(1,\cdot))$, $1 \leq i \leq n-2$. We derive an upper bound for the expected number of generations until $f(P) \succeq_{r^{i+1}} f(\text{PATH}_{\leq i+1}(1,\cdot))$. Consider the nondominated box indices of $b_r(f(\text{PATH}_{i+1}(1,\cdot)))$. There are at most

$$\nu := (n-1) \cdot (\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 1)^{k-1} + 1$$

such indices. We fix from each of the corresponding boxes a path $p_j \in \text{PATH}_{i+1}(1,\cdot)$. The first part of Lemma 2 ensures that for each $p_j$ there is a $p'_j \in P$ such that an elementary mutation of $p'_j$ suffices to create an individual that $r^i$-dominates $p_j$. The second part of Lemma 2 ensures that all following populations $r^{i+1}$-dominate $p_j$.

The probability to choose a certain individual for mutation is $1/|P|$, the probability to perform exactly one elementary mutation is $1/e$, and the probability to append a certain edge $(u,v)$ is at least $1/(2(n-1))$ where the last probability stems from the fact that the mutation operator decides for appending an edge with probability $1/2$ and chooses $(u,v)$ amongst all outgoing edges with probability at least $1/(n-1)$. Hence, the probability to create a path that $r^i$-dominates $p_j$ is at least $1/(2e(n-1)\nu)$ since $|P| \leq \nu$ due to Lemma 3. Due to the coupon collector's problem [12], we have to wait an expected number of

$$2e(n-1)\nu \cdot \sum_{j=1}^{\nu} \frac{1}{j} = 2e(n-1)\nu \cdot H_\nu$$

steps until $f(P) \succeq_{r^{i+1}} f(\text{PATH}_{\leq i+1}(1,\cdot))$ where $H_n$ is the $n$-th harmonic number.

Taking all $n-2$ phases into account, the expected number of generations until the population has evolved into an $\varepsilon$-approximate Pareto set is at most

$$\begin{aligned} & (n-2) \cdot 2e(n-1)\nu H_\nu \\ = \; & \text{O}\left(n^3 \cdot (\log_r(n \cdot w^{\max}))^{k-1} \cdot \log\left(n \cdot (\log_r(n \cdot w^{\max}))^{k-1}\right)\right) \end{aligned}$$

since $H_n = \mathrm{O}(\log(n))$. The upper bound can be simplified according to

$$\begin{aligned}
&\mathrm{O}\left(n^3 \cdot \left(\frac{\log\left(n \cdot w^{\max}\right)}{\log\left(r\right)}\right)^{k-1} \cdot \log\left(n \cdot \left(\frac{\log\left(n \cdot w^{\max}\right)}{\log\left(r\right)}\right)^{k-1}\right)\right) \\
= \ &\mathrm{O}\left(n^3 \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\log\left(1+\varepsilon\right)}\right)^{k-1} \cdot \left(\log\left(n\right) + k \cdot \log\left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\log\left(1+\varepsilon\right)}\right)\right)\right)
\end{aligned}$$

Using $k = \mathrm{O}(1)$ and $\log(n) = \mathrm{O}(\log(n \cdot \log(n \cdot w^{\max})/\log(1+\varepsilon)))$ the last bound can be reduced to

$$\begin{aligned}
&\mathrm{O}\left(n^3 \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\log\left(1+\varepsilon\right)}\right)^{k-1} \cdot \log\left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\log\left(1+\varepsilon\right)}\right)\right) \\
= \ &\mathrm{O}\left(n^3 \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)^{k-1} \cdot \log\left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)\right)
\end{aligned}$$

since $\log(1+\varepsilon) \geq \varepsilon$ if $0 \leq \varepsilon \leq 1$. □

To transform the upper bound on the optimization time into an upper bound on the runtime we have to determine the runtime for the initialization and a loop iteration. If we consider a standard implementation of the above algorithm, the initialization requires $\Theta(m)$ steps and a loop iteration requires an expected number of

$$\Theta(|P| \cdot d) = \mathrm{O}\left(n^2 \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)^{k-1}\right)$$

steps. Hence, the above theorem guarantees that the expected number of steps until the population has evolved into an $\varepsilon$-approximate Pareto set is at most

$$\mathrm{p}(n, 1/\varepsilon) = \mathrm{O}\left(n^5 \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)^{2 \cdot (k-1)} \cdot \log\left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)\right)$$

where $\mathrm{p}(\cdot, \cdot)$ is an appropriate polynomial. If we terminate DEMO($r$) after $4 \cdot \mathrm{p}(n, 1/\varepsilon)$ steps, Markov's inequality ensures that the resulting population is an $\varepsilon$-approximate Pareto set with probability greater than $3/4$. Hence, the algorithm is an FPRAS for the single-source multiobjective shortest path problem.

The next theorem shows that the optimization time bound given in the above theorem can be improved. The above proof follows a classical proof technique, which relies on upper bounds on the expected optimization time needed to evolve an $r^i$-approximate Pareto set of $f\colon \mathrm{PATH}_{\leq i}(1, \cdot) \to (\mathbb{R}^+)^d$ into an $r^{i+1}$-approximate Pareto set of $f\colon \mathrm{PATH}_{\leq i+1}(1, \cdot) \to (\mathbb{R}^+)^d$. The

used proof technique is well structured, but does not account for the construction of good paths with more than $i+1$ edges as long as not all paths with at most $i+1$ edges are $r^{i+1}$-dominated. Since the mentioned effect often occurs, the derived bound is not sharp. The proof of the following theorem relies on an proof technique developed in [1], which allows to derive an improved upper bound on the optimization time.

**Theorem 2.** *Let* $0 < \varepsilon \leq 1$ *and* $c > (1+\varepsilon)^{(n-2)/(n-1)}$. *Then* DEMO($r$) *with* $r = (1+\varepsilon)^{1/(n-1)}$ *optimizing* $f$ *achieves an* $\varepsilon$*-approximate Pareto set within*

$$t := 2 \cdot (n-2) \cdot (2e(n-1)\nu) = \mathrm{O}\left(n^3 \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)^{k-1}\right)$$

*generations with probability* $1 - \exp(-\Omega(n))$ *where*

$$\nu := (n-1) \cdot (\lfloor \log_r((n-1) \cdot w^{\max}) \rfloor + 1)^{k-1} + 1.$$

*Proof.* Consider a simple path $p = (v_1, \ldots, v_\ell)$, $1 \leq \ell \leq n$, with $v_1 = 1$. We call a mutation step *$i$-th improvement of $p$*, $1 \leq i \leq \ell - 2$, if an individual that $r^i$-dominates $(v_1, \ldots, v_{i+2})$ is created for the first time. Due to the proof of Theorem 1, the probability for an improvement is at least $p := 1/(2e(n-1)\nu)$. Consider a phase of length $t$. Define the random variables $X_i$, $1 \leq i \leq t$, with $\mathrm{Prob}(X_i = 1) = p$ and $\mathrm{Prob}(X_i = 0) = 1 - p$, and $X = \sum_{i=1}^{t} X_i$. It holds that

$$\mathrm{E}(X) = t \cdot p = 2(n-2)(2e(n-1)\nu) \cdot \frac{1}{2e(n-1)\nu} = 2(n-2)$$

due to the linearity of the expectation operator. Using these random variables the probability that the $(\ell-2)$-th improvement has not occured within $t$ mutation steps is at most

$$\begin{aligned}
\mathrm{Prob}(X < \ell - 2) &\leq \mathrm{Prob}(X < n-2) \\
&= \mathrm{Prob}(X < \frac{n-2}{\mathrm{E}(X)} \cdot \mathrm{E}(X)) \\
&= \mathrm{Prob}(X < (1-\delta) \cdot \mathrm{E}(X))
\end{aligned}$$

where $\delta := 1/2$ since

$$\frac{n-2}{\mathrm{E}(X)} = \frac{n-2}{2(n-2)} = \frac{1}{2}.$$

Due to a Chernoff bound [12],

$$\mathrm{Prob}(X < (1-\delta) \cdot \mathrm{E}(X)) \leq \exp(-\frac{\mathrm{E}(X) \cdot \delta^2}{2}) = \exp(-\frac{n-2}{4}).$$

There are at most $\nu$ nondominated box indices of $b_r(f(\mathrm{PATH}_{\leq n-1}(1,\cdot)))$. We fix from each of the corresponding boxes a path $p_j = (v_{j,1}, \ldots, v_{j,\ell_j})$. Due to Boole's inequality, the probability that the $(\ell_j - 2)$-th improvement of an $p_j$ has not occured within $t$ mutation steps is at most

$$\nu \cdot \exp(-\frac{n-2}{4}) = \exp(-\Omega(n)),$$

Hence, DEMO($r$) achieves an $\varepsilon$-approximate Pareto set within $t$ generations with the converse probability $1 - \exp(-\Omega(n))$. □

# 7 Proof Technique

We point out the underlying proof technique of Theorem 2. We hope that the proof technique turns out to be useful for the analysis of DEMO($r$) on other combinatorial optimization problems. An inspection of the proof of Theorem 2 leads to the following result.

**Theorem 3.** *Let $g\colon S \to (\mathbb{R}^+)^d$ and $S_i \subseteq S$, $1 \leq i \leq n$, with $S_{i-1} \subseteq S_i$, $1 < i \leq n$, and $S_n = S$. Consider* DEMO($r$) *and the following conditions.*

1. *The initial population is a Pareto set of $g\colon S_1 \to (\mathbb{R}^+)^d$.*
2. *The population size is at most $\nu$.*
3. *For each population that is an $r^i$-approximate Pareto set of $g\colon S_i \to (\mathbb{R}^+)^d$ and for each $s \in S_{i+1}$, the population contains an individual that would be evolved by the mutation operator with probability $p > 0$ into an individual that $r^i$-dominates $s$.*

*If all conditions hold, then* DEMO($r$) *optimizing $g$ achieves an $r^n$-approximate Pareto set of $g\colon S \to (\mathbb{R}^+)^d$ within*

$$\frac{2 \cdot (n-1) \cdot \nu}{p}$$

*generations with probability $1 - \exp(-\Omega(n))$.*

The proof technique stated in Theorem 3 can be seen as a generalization of the method of fitness-based partitions [6] designed for the analysis of EAs on singleobjective optimization problems.

# 8 Discussion

We have shown that a natural EA operating on a natural model of a practical relevant NP-hard problem is an FPRAS. Comparing the upper bound on

the optimization time of the EA (see Theorem 2) with the upper bound

$$\mathrm{O}\left(n \cdot m \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)^{k-1}\right)$$

on the runtime of the FPTAS provided in [20], which is the most time-efficient FPTAS available, shows that both bounds coincide up to a factor of $\Theta(n^2/m)$. For dense graphs the difference reduces to $\Theta(1)$. It appears to be plausible that this difference is close to the true difference of the optimization time and the runtime. While the EA relies on a randomized variation operator to explore the search space, the FPTAS explores the search space in a systematic manner. Hence, although the EA follows a randomized approach, it achieves the optimization goal with similar success. However, a standard implementation of the proposed EA needs an expected number of

$$\mathrm{O}\left(n^2 \cdot \left(\frac{n \cdot \log\left(n \cdot w^{\max}\right)}{\varepsilon}\right)^{k-1}\right)$$

steps for the simulation of a single generation, which deteriorates the actual runtime.

It would be interesting to set further limits to the optimization time of the EA by showing a nontrivial lower bound. At the end the upper and lower bound should match. Furthermore it would be interesting to investigate if other general randomized search heuristics beat the runtime of the proposed EA. Another possible topic for future research is the analysis of the average-case behavior of the EA w. r. t. reasonable distributions of the input instances.

# References


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

[2] B. Doerr, E. Happ, and C. Klein. Crossover can provably be useful in evolutionary computation. In *Proc. of GECCO 2008*, pages 539–546. ACM Press, 2008.

[3] B. Doerr, N. Hebbinghaus, and F. Neumann. Speeding up evolutionary algorithms through asymmetric mutation operators. *Evolutionary Computation*, 15(4):401–410, 2007.

[4] B. Doerr and D. Johannsen. Adjacency list matchings — an ideal genotype for cycle covers. In *Proc. of GECCO 2007*, pages 1203–1210. ACM Press, 2007.

[5] B. Doerr, C. Klein, and T. Storch. Faster evolutionary algorithms by superior graph representations. In *Proc. of FOCI 2007*, pages 245–250. IEEE Press, 2007.

[6] 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.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[8] O. Giel and I. Wegener. Evolutionary algorithms and the maximum matching problem. In *Proc. of STACS 2003*, volume 2607 of *LNCS*, pages 415–426. Springer, 2003.

[9] P. Hansen. Bicriterion path problems. In *Proc. of MCDM 1979*, volume 117 of *LNEMS*, pages 109–127. Springer, 1979.

[10] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[11] M. Laumanns, L. Thiele, and E. Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182, 2004.

[12] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

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

[14] F. Neumann. Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. *Computers and Operations Research*, 35(9):2750–2759, 2008.

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

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

[17] P. S. Oliveto, J. He, and X. Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007.

[18] C. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proc. of FOCS 2000*, pages 86–92. IEEE Press, 2000.

[19] 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.

[20] G. Tsaggouris and C. Zaroliagis. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. In *Proc. of ISAAC 2006*, volume 4288 of *LNCS*, pages 389–398. Springer, 2006.

[21] A. Warburton. Approximation of Pareto optima in multiple-objective shortest path problems. *Operations Research*, 35(1):70–79, 1987.

[22] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS 2005*, volume 3404 of *LNCS*, pages 44–56. Springer, 2005.