Running Time Analysis of Multi-objective Evolutionary Algorithms on a Simple Discrete Optimization Problem

Marco Laumanns[1], Lothar Thiele[1], Eckart Zitzler[1], Emo Welzl[2], and Kalyanmoy Deb[3]

[1] ETH Zürich, Computer Engineering and Networks Laboratory, CH–8092 Zürich
{laumanns,thiele,zitzler}@tik.ee.ethz.ch
http://www.tik.ee.ethz.ch/aroma
[2] ETH Zürich, Institute of Theoretical Computer Science, CH–8092 Zürich
welzl@inf.ethz.ch
[3] Department of Mechanical Engineering, Indian Institute of Technology Kanpur, Kanpur, PIN 208 016, India
deb@iitk.ac.in

Abstract. For the first time, a running time analysis of population-based multi-objective evolutionary algorithms for a discrete optimization problem is given. To this end, we define a simple pseudo-Boolean bi-objective problem (LOTZ: leading ones - trailing zeroes) and investigate time required to find the entire set of Pareto-optimal solutions. It is shown that different multi-objective generalizations of a (1+1) evolutionary algorithm (EA) as well as a simple population-based evolutionary multi-objective optimizer (SEMO) need on average at least $\Theta(n^3)$ steps to optimize this function. We propose the fair evolutionary multi-objective optimizer (FEMO) and prove that this algorithm performs a black box optimization in $\Theta(n^2 \log n)$ function evaluations where n is the number of binary decision variables.

1 Introduction

Evolutionary Algorithms (EAs) are probabilistic search heuristics that mimic principles of natural evolution. They are often used to solve optimization problems, in particular those with multiple objectives, for an overview see e.g. [1]. In multi-objective optimization, the aim is to find or to approximate the set of Pareto-optimal (or non-dominated) solutions.

Existing theoretic work on convergence in evolutionary multi-objective optimization has so far mainly dealt with the limit behavior [9, 10, 12, 11, 4, 5, 14]. Under appropriate conditions for the variation and the selection operators, global convergence to the Pareto set can be guaranteed in the limit.

In addition to that, we are often interested in a quantitative analysis, specifically the expected running time for a given class of problems and the success probability for a given optimization time. For single-objective evolutionary algorithms many such results are contained in [8]. For the optimization of pseudo-Boolean functions an extensive theory has been built up by Wegener et al., see

e.g. [16], Droste, Jansen, and Wegener [2, 3], or for a methodological overview [15].

Results on the running time of evolutionary algorithms in the multi-objective case are rare. Scharnow et al. [13] analyze a (1+1)-EA under multiple, non-conflicting objectives. The purpose of this paper is to present a first analysis of different population-based multi-objective evolutionary algorithms (MOEAs) on a two-objective model problem. In particular, the following results are described in the paper:

- The well known "Leading Ones" problem is generalized to two dimensions. The new problem class is called LOTZ (Leading Ones - Trailing Zeros).
- A simple evolutionary multi-objective optimization algorithm is defined (SEMO - Simple Evolutionary Multi-objective Optimizer). Its expected running time on the above problem is shown to be $\Theta(n^3)$.
- The algorithm is improved by a fair sampling strategy of the population (FEMO - Fair Evolutionary Multi-objective Optimizer). Its running time on the above problem is $\Theta(n^2 \log n)$ with a high probability of $1 - O(1/n)$ and its expected running time is $O(n^2 \log n)$.

The model problem and its characteristics are introduced in section 2. It is a multi-objective extension of the "Leading Ones" problem which has been thoroughly analyzed for example in [8] and [3]. The algorithms are described and analyzed in sections 3 and 4. They are instances of a steady state $(\mu + 1)$-EA with variable population size and differ in the manner how the parents are sampled from the population.

2 The Model Problem

As the example problem for this analysis, we consider the maximization of a 2-dimensional vector valued function, LOTZ, which maps n binary decision variables to 2 objective functions.

Definition 1. *The pseudo-Boolean function* $\text{LOTZ} : \{0,1\}^n \to \mathbb{N}^2$ *is defined as*

$$\text{LOTZ}(x_1, \ldots, x_n) = \left(\sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j) \right)$$

The abbreviation LOTZ stands for "Leading Ones, Trailing Zeroes" and means that we want to simultaneously maximize the number of leading ones and trailing zeroes in a bit-string. The first component, the LEADINGONES function, has been analyzed in detail in [8] and [3].

As there is no single search point that maximizes both components simultaneously, we want to find the whole set of non-dominated points based on the concept of Pareto optimality, here defined for an m-objective maximization problem with binary decision variables.

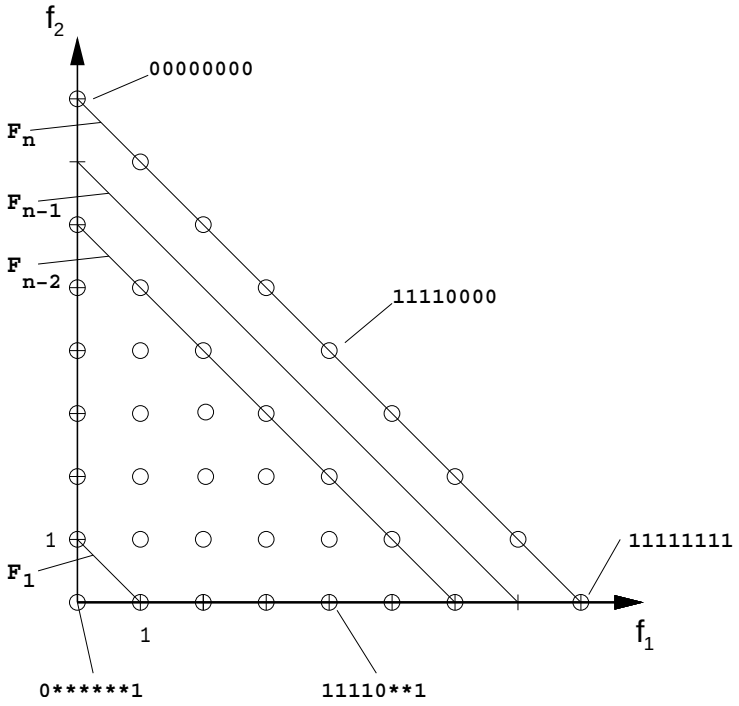

Fig. 1. Objective space of the LOTZ function with $n = 8$

Definition 2 (Pareto optimality, Pareto set). *Let $f : X \to F$ where $X \subseteq \{0,1\}^n$ and $F \subseteq \mathbb{R}^m$. A decision vector $x^* \in X$ is* Pareto optimal *if there is no other $x \in X$ that* dominates *x^*. x dominates x^*, denoted as $x \succ x^*$ if $f_i(x) \geq f_i(x^*)$ for all $i = 1, \ldots, m$ and $f_i(x) > f_i(x^*)$ for at least one index i. The set of all Pareto optimal decision vectors X^* is called* Pareto set*. $F^* = f(X^*)$ is the set of all Pareto optimal objective vectors and denoted as the* Pareto front*.*

The objective space of this problem can be partitioned into $n+1$ sets $F_i, i = 0, \ldots, n$ (see Fig. 1). The index i corresponds to the sum of both objective values, i.e. $(f_1, f_2) \in F_i$ if $i = f_1 + f_2$. Obviously, F_n represents the Pareto front F^*. The sub-domains X_i are defined as the sets containing all decision vectors which are mapped to elements of F_i. They are of the form $1^a 0 *^{(n-i-2)} 10^b$ with $a + b = i$ for $i < n$, and $1^a 0^b$ with $a + b = n$ for X_n.

The cardinality of the Pareto set $X^* = X_n$ is $|X_n| = n + 1$ and we also have $n + 1$ Pareto optimal objective vectors as $|F_n| = n + 1$. The next set F_{n-1} is empty. For the remaining sets with $i = 0, \ldots, n-2$ we have $|F_i| = i + 1$ and $|X_i| = |F_i| \cdot 2^{n-2-i}$. As a consequence, the decision space X contains 2^n different elements, which are mapped to $|F_n| + \sum_{i=0}^{n-2} |F_i| = 1/2 \cdot n^2 + 1/2 \cdot n + 1 = O(n^2)$ different objective vectors.

2.1 How Difficult Is It to Find the Whole Pareto Set?

How long does it take to optimize the LOTZ function? Droste et al. [3] have proved that the expected running time of a (1+1)-EA on LEADINGONES is $\Theta(n^2)$. Using the same algorithm with an appropriate generalization of the acceptance criterion (either accepting only dominating offspring or by using a weighted sum as a scalar surrogate objective) will certainly lead to finding *one* element of the Pareto set in the same amount of time.

To find all different Pareto optimal points with such a (1+1) EA we can consider the multi-start option, i.e. to run to the EA several times, and collect all non-dominated solutions in an archive. For the acceptance criterion based on the dominance relation, the random variable describing the number of ones in the final solution of each single run follows a binomial distribution with $p = 0.5$. Hence the probability of finding the "outer" points of the Pareto set decreases exponentially. This would mean that the running time of this strategy until all Pareto optimal points are found is exponentially large in n.

Another possibility would be to use the multi-start option together with a weighted sum of the objective values. However, an appropriate choice of the weights is very difficult. In our case, equal weights would lead to the same situation as before, with a very low probability to reach the outer points. Any other selection of weights will let the sequence of search points converge to one of the outer points of the Pareto set. The remaining points must be found "on the way", but the probability of such events is not easy to calculate. Even if we could supply $n + 1$ different weights corresponding to each of the $n + 1$ Pareto optimal points, this strategy would still need $(n + 1) \cdot \Theta(n^2) = \Theta(n^3)$ steps.

Contrary to the multi-start option, one could relax the acceptance criterion such that only dominated offspring are rejected. This allows to accept other Pareto-optimal solutions once an element of the Pareto set has been found and one could continue this process until all Pareto-optimal solutions have been visited at least once. This concept has been implemented in the PAES algorithm [6]. It finally leads to a random walk on or around the Pareto set, and the running time of the algorithm could be calculated based on the cover time of the random walk. A random walk on the path given by the $n + 1$ Pareto-optimal solutions has a cover time of $\Theta(n^2)$. As each move of this random walk needs a successful mutation, whose probability is bounded above by $2/n$, the whole process again needs $\Theta(n^3)$ steps.

A last possibility would be to use a simple strategy known from classical multi-objective function optimization. In this case, we optimize only one objective, e.g. the number of leading ones, and constrain the other objective to be strictly larger than its value obtained in the previous optimization run. Therefore, we find all $n + 1$ Pareto vectors in $n + 1$ runs of a single-objective EA with an additional constraint. At the best, this strategy again needs $\Theta(n^3)$ steps.

The above discussion indicates that a (1+1) strategy may not be the best approach for solving multi-objective optimization problems. Moreover, most of the current multi-objective optimization algorithms use the concept of an archive that maintains a set of Pareto optimal vectors of all decision vectors visited so

Algorithm 1 Simple Evolutionary Multi-objective Optimizer (SEMO)

1: Choose an initial individual x uniformly from $X = \{0,1\}^n$
2: $P \leftarrow \{x\}$
3: **loop**
4: Select one element x out of P uniformly.
5: Create offspring x' by flipping a randomly chosen bit.
6: $P \leftarrow P \setminus \{z \in P | x' \succ z\}$
7: **if** $\nexists z \in P$ such that $(z \succ x' \vee f(z) = f(x'))$ **then**
8: $P \leftarrow P \cup \{x'\}$
9: **end if**
10: **end loop**

far. This indicates that the concept of a population is vital in multi-objective evolutionary optimization. In the next sections, we propose and analyze two simple *population-based* steady state EAs.

3 A Simple Evolutionary Multi-objective Optimizer (SEMO)

At first, we analyze a simple population-based multi-objective EA. This algorithm contains a population of variable size that stores all non-dominated individuals. From this population a parent is drawn according to some probability distribution and mutated by flipping a randomly chosen bit. For Algorithm 1 we consider a uniform distribution for selecting the parent.

An appropriate archiving strategy [7] is assumed to prevent the population from growing exponentially. For this study it suffices to ensure that a solution is only accepted if it has different objective values (line 7).

3.1 Running Time Analysis of SEMO Applied to LOTZ

The running time of an algorithm equals the number of necessary evaluations of the objective function. For the analysis we divide the run of the SEMO into two distinct phases: the first phase lasts until the first Pareto-optimal individual has entered the population, and the second phase ends when the whole Pareto set has been found.

Lemma 1 (Expected running time for phase 1). *The expected running time of Alg. 1 until the first Pareto-optimal point is found is* $O(n^2)$.

Proof. Note that during this first phase the population will consist of one individual only, as a mutation changing the objective values yields either a dominating or a dominated individual. Hence, if an offspring is accepted, it will replace the parent from which it was produced. We consider the partition of the search space into distinct subsets X_i as defined in section 2 and note that from any

subset X_i only points in $X_j, j > i$ are accepted. As there is always a one-bit mutation leading to the next subset, the probability of improvement is at least $1/n$. As there are at most $n-1$ such steps necessary (X_{n-1} is empty) the expected time is at most n^2. □

Lemma 2 (Expected running time for phase 2). *After the first Pareto-optimal point is found, the expected running time of Alg. 1 until all Pareto-optimal points are found is $\Theta(n^3)$ and the probability that the running time is less than $n^3/c(n)$ is less than $(8e/c(n))^{n/2}$.*

Proof. We partition this phase into $n-1$ different sub-phases. Sub-phase i lasts from the time when $i-1$ Pareto-optimal solutions have been found to the time when the next solution is found. T_i is a random variable denoting the duration of sub-phase i and the random variable T is the sum of these times. As we always have a contiguous subset of the Pareto set, only the individuals corresponding to the outer points of this subset can create a new Pareto-optimal point. The probability $p_s(i)$ to sample such a candidate in phase i is at least $1/i$ and at most $2/i$. A subsequent mutation has a success probability of at least $1/n$ and at most $2/n$. Hence, $ni/4 \leq E(T_i) \leq ni$. As $T = \sum_{i=1}^{n-1} T_i$, $1/8n^3 - 1/8n^2 \leq E(T) \leq 1/2n^3 - 1/2n^2$.
To derive a lower bound of the running time which holds with a high probability we consider the run after $n/2$ Pareto-optimal solutions have already been found. In this case the probability to find a new Pareto-optimal solution is at most $4/n^2$. If we allow $n^3/c(n)$ trials, the expected number of successes S is at most $4n/c(n)$. With Chernoff's inequality, the probability that we reach the required $n/2+1$ successes to find the remaining solutions can be bounded as

$$P(S > n/2) \leq \left(\frac{e^{\frac{1}{8}c(n)-1}}{(\frac{1}{8}c(n))^{\frac{1}{8}c(n)}} \right)^{4n/c(n)} \leq \left(\frac{8e}{c(n)} \right)^{\frac{1}{2}n}$$

□

From the concatenation of the two phases the following Corollary can be derived.

Corollary 1 (Expected running time Alg. 1). *The expected running time of Alg. 1 until all Pareto-optimal points are found is $\Theta(n^3)$.*

For this problem, the simple population-based SEMO is at least as good as any potential multi-objective adaptation of the (1+1)-EA discussed in the previous section. But is there room for further improvement? As it takes about n^2 steps to find *one* Pareto-optimal point, there is no hope to find the whole Pareto set in less time. But the time to generate all Pareto-optimal points can be reduced substantially.

4 The Fair Evolutionary Multi-objective Optimizer (FEMO)

The main weakness of the SEMO for the optimization problem under consideration lies in the fact that a large number of mutations are allocated to parents

Algorithm 2 Fair Evolutionary Multi-objective Optimizer (FEMO)

```
1: Choose an initial individual x uniformly from X = {0,1}^n
2: w(x) ← 0                                          {Initialize offspring count}
3: P ← {x}
4: loop
5:    Select one element x out of {y ∈ P|w(y) ≤ w(z) ∀z ∈ P} uniformly.
6:    w(x) ← w(x) + 1                                {Increment offspring count}
7:    Create offspring x' by flipping a randomly chosen bit.
8:    P ← P \ {z ∈ P|x' ≻ z}
9:    if ∄z ∈ P such that (z ≻ x' ∨ f(z) = f(x')) then
10:      P ← P ∪ {x'}
11:      w(x) ← 0                                    {Initialize offspring count}
12:   end if
13: end loop
```

whose neighborhood has already been explored sufficiently. On the other hand, an optimal sampling algorithm would use always the most promising parent at the border of the current population. Of course, this information is not available in a black box optimization scenario.

The uniform sampling leads to a situation, where the Pareto-optimal individuals have been sampled unevenly depending on when each individual entered the population. The following *fair* sampling strategy guarantees that the end all individuals receive about the *same* number of samples.

Algorithm 2 implements this strategy by counting the number of offspring each individual produces (line 6). The sampling procedure deterministically chooses the individual which has produced the least number of offspring so far, ties are broken randomly (line 5).

4.1 Running Time Analysis of FEMO Applied to LOTZ

For the analysis of Algorithm 2 we focus only on the second phase as the first phase is identical to the simple Algorithm 1 described before.

Once the first two Pareto-optimal points are found, there is exactly one possible parent for each of the remaining $n-1$ points. We are interested in the number of mutations that must be allocated to *each* of these $n-1$ parents in order to have at least one successful mutation each that leads to the desired child. Lemma 3 and 4 provide upper and lower bounds on the probability that a certain number of mutations per parent are sufficient. In Theorem 1 these probabilities are used to bound the running time of the FEMO algorithm.

Lemma 3 (Minimal success probability). *Let p be the success probability for each single mutation and $c > 0$ an arbitrary constant. With probability at least $1 - n^{1-c}$ all $n-1$ remaining offspring have been constructed in at most $c \cdot 1/p \cdot \log n$ mutation trials for each corresponding parent.*

Proof. For each individual, the probability of having at least $c \cdot 1/p \cdot \log n$ non-successful mutation is bounded above by

$$(1-p)^{c\cdot 1/p\cdot \log n} = (1-\frac{1}{1/p})^{c/p\log n} = (1-\frac{1}{1/p})^{1/p^{c\log n}} \leq \left(\frac{1}{e}\right)^{c\log n} = \frac{1}{n^c}$$

There are $n-1$ individuals that must be produced with the given number of trials. These events are independent, so the probability that at least one individual needs more than $c/p \cdot \log n$ trials is bounded above by $\frac{n-1}{n^c} \leq n^{1-c}$. □

Lemma 4 (Maximal success probability). *Let $k \in \{1,\ldots,n\}$, $a = k/n$, and $c > 0$ be an arbitrary constant. The probability that $k = a \cdot n$ individuals are produced in $c \cdot 1/p \cdot \log n$ mutation steps each is not larger than $(e^a)^{-n^{(1-c-c/n)}}$.*

Proof. The probability that a parent has created a certain offspring within the first $t = c \cdot 1/p \cdot \log n$ trials is $1-(1-p)^t$. The probability that this happens independently for a selection of k such pairs can thus be bounded as

$$(1-(1-p)^t)^k \leq (1-\frac{1}{n^{c\frac{n+1}{n}}})^{an} \leq e^{-\frac{an}{n^{c(n+1)/n}}} = (e^a)^{-n^{(1-c-c/n)}}$$

□

Now we can translate the number of mutations that are needed into the running time of Algorithm 2.

Theorem 1 (Running time bounds). *With probability at least $1-O(1/n)$ the number of objective function evaluations T Algorithm 2 needs from the discovery of the first two Pareto-optimal points until the whole Pareto set has been found lies in the interval $[1/4 \cdot 1/p \cdot n\log n, 2 \cdot 1/p \cdot n\log n]$. Hence, $Prob\{T = \Theta(1/p \cdot n\log n)\} = 1-O(1/n)$. Furthermore, $E(T) = O(1/p \cdot n\log n)$.*

Proof. Let the Pareto-optimal points be indexed according to the order in which they have entered the set P. Let $k \in \{0,\ldots,n\}$ be the index of the individual that required the largest number of mutations to be produced. We apply Lemma 3 with $c = 2$ and notice that this individual k did not need more than $2/p \cdot \log n$ trials with probability $1-O(1/n)$.
What remains to be shown for the upper bound is that no node will be *sampled* more than t times during the algorithm. This can be guaranteed since there is always a candidate $x \in P$ with $w(x) \leq t$ (the element that has most recently been added to P). Hence, any element whose weight has reached t will never be sampled again. As there are n such elements, each of which is sampled at most t times, the total number of samples (steps) the algorithm takes does not exceed $T = n \cdot t = 2 \cdot 1/p \cdot n\log n$.

For the lower bound we apply Lemma 4 with $c = 1/2$ and $k = n/2$. With a probability of $1-\sqrt{e}^{-n^{(0.5-0.5/n)}}$ there is an individual in the second half which needs at least $1/2 \cdot 1/p \cdot \log n$ trials. Hence, all individuals in the first half have

been sampled at least $1/2 \cdot 1/p \cdot \log n - 1$ times each. Of course, all individuals in the second half must be sampled at least once. The summation over all nodes gives a total number of samples of at least $1/4 \cdot 1/p \cdot n \log n$ with probability $1 - O(1/n)$.

Using the probability bound from Lemma 3 the expected running time can be bounded as

$$\begin{aligned}(1/p \cdot n \log n)^{-1} E(T) &\leq 1 \cdot P\{0 \leq T < 1\} + 2 \cdot P\{1 \leq T < 2\} + \ldots \\ &\leq 2 + \sum_{c=3}^{\infty} c \cdot P\{T \geq k-1\} \\ &\leq 2 + \sum_{c=1}^{\infty} (c+2) n^{-c} \\ &\leq 2 + \frac{n}{(n-1)^2} + \frac{2}{n-1} \quad .\end{aligned}$$

Hence, $E(T) = O(1/p \cdot n \log n)$. □

As before, the time to find the first one (or two) elements of the Pareto set can be neglected and the total running time is mainly determined by Theorem 1. For our case the mutation success probability is $p = 1/n$, which leads with a high probability to a running time of $\Theta(n^2 \log n)$. This is a considerable improvement in comparison to any multi-start strategy of a single objective EA and to the SEMO algorithm.

5 Concluding Remarks

In this paper we have given first analytical results on the running time of evolutionary algorithms for multi-objective optimization problems. We have defined a bi-objective model problem and discussed different concepts of how to find the whole set of Pareto-optimal solutions for this problem.

For a simple steady-state evolutionary multi-objective optimizer (SEMO) a running time of $\Theta(n^3)$ was proven, which is at least as good as any strategy based on using a scalar surrogate objective function or multi-objective adaptations of the (1+1)-EA.

We proposed FEMO, a new evolutionary multi-objective optimizer involving an archive or population and a fair sampling strategy. This algorithm improves the running time substantially as it is able to find the whole Pareto set in $\Theta(n^2 \log n)$ steps.

The FEMO algorithm uses information collected during the run and stored in a population. For the first time it could be shown analytically that this concept of a population-based EA leads to a provable advantage on a multi-objective optimization problem compared to standard approaches based on scalarizing functions.

In addition to its good running time behavior, the algorithm is simple and problem- and representation-independent so that it might be easily and successfully applied to practical problems as well.

6 Acknowledgments

The research has been funded by the Swiss National Science Foundation (SNF) under the ArOMA project 2100-057156.99/1.

References

1. K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.
2. S. Droste, T. Jansen, and I. Wegener. A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with Boolean inputs. *Evolutionary Computation*, 6(2):185–196, 1998.
3. 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.
4. T. Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal Of Operational Research*, 117(3):553–564, 1999.
5. T. Hanne. Global multiobjective optimization with evolutionary algorithms: Selection mechanisms and mutation control. In *Evolutionary Multi-Criterion Optimization (EMO 2001), Proc.*, LNCS 1993, pages 197–212, Berlin, 2001. Springer.
6. J. D. Knowles and D. W. Corne. Approximating the non-dominated front using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
7. M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3), 2002.
8. G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač, Hamburg, 1997.
9. G. Rudolph. Evolutionary search for minimal elements in partially ordered sets. In *Evolutionary Programming VII – Proc. Seventh Annual Conf. on Evolutionary Programming (EP-98)*, San Diego CA, 1998. The MIT Press, Cambridge MA.
10. G. Rudolph. On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *IEEE Int'l Conf. on Evolutionary Computation (ICEC'98)*, pages 511–516, Piscataway, 1998. IEEE Press.
11. G. Rudolph. Evolutionary Search under Partially Ordered Fitness Sets. In *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)*, pages 818–822. ICSC Academic Press: Millet/Sliedrecht, 2001.
12. G. Rudolph and A. Agapie. Convergence properties of some multi-objective evolutionary algorithms. In *Congress on Evolutionary Computation (CEC 2000)*, volume 2, pages 1010–1016, Piscataway, NJ, 2000. IEEE Press.
13. J. Scharnow, K. Tinnefeld, and I. Wegener. Fitness landscapes based on sorting and shortest paths problems. This volume.
14. D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Graduate School of Engineering of the Air Force Institute of Technology, Air University, June 1999.
15. I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. Technical Report CI-99/00, SFB 531, Universität Dortmund, 2000.
16. I. Wegener. Theoretical aspects of evolutionary algorithms. In *ICALP 2001*, volume 2076 of *LNCS*, pages 64–78. Springer-Verlag, 2001.