Working Paper

Running time analysis of evolutionary algorithms on vector-valued pseudo-Boolean functions

Author(s):
Laumanns, Marco; Thiele, Lothar; Zitzler, Eckart

Publication Date:
2004

Permanent Link:

https://doi.org/10.3929/ethz-a-004723830 →

Running Time Analysis of Evolutionary Algorithms on Vector-Valued Pseudo-Boolean Functions

Marco Laumanns Lothar Thiele Eckart Zitzler

TIK-Report No. 165
Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich
Gloriastrasse 35, ETH-Zentrum, CH–8092 Zürich, Switzerland

Abstract

This paper presents a rigorous running time analysis of evolutionary algorithms on pseudo-Boolean multiobjective optimization problems. We propose and analyze different population-based algorithms, the simple evolutionary multiobjective optimizer SEMO and two improved versions, FEMO and GEMO. The analysis is carried out on two bi-objective model problems, LOTZ (Leading Ones Trailing Zeroes) and COCZ (Count Ones Count Zeroes) as well as on the scalable m-objective versions mLOTZ and mCOCZ. Results on the running time of the different population-based algorithms and for an alternative approach, a multistart (1+1)-EA based on the ϵ-constraint method, are derived The comparison reveals that for many problems, the simple algorithm SEMO is as efficient as the (1+1)-EA. For some problems, the improved variants FEMO and GEMO are provably better. For the analysis we propose and apply two general tools, an upper bound technique based on a decision space partition and a randomized graph search algorithm, which facilitate the analysis considerably.

1 Introduction

Evolutionary Algorithms (EAs) are probabilistic search techniques that are inspired by models of natural evolution. The underlying principles are simple, but nevertheless EAs usually exhibit a complex behavior which is hard to analyze theoretically. Consequently, a lot of empirical knowledge and successful applications have been reported in the literature, but much less rigorous theoretical results about their efficiency are available. Besides empirical investigations, though, theoretical work is important for making more general, but also more precise statements about the performance of EA variants and for better understanding the dynamics of EAs.

A recent overview of the theoretical analysis of EAs is given by Beyer et al. (2). A major part of this theory is the running time analysis, which addresses the question

of how long a certain algorithm takes to find the optimal solution for a specific problem or a class of problems. Such an analysis typically contains the following ingredients:

1. Simple, well-defined algorithms, which are simple instances or stochastic models of EAs,
2. Sample problems (or problem classes), which the algorithms are applied to, and
3. Analytical methods and tools, which are used for the study of the algorithms.

In the case of a single objective and discrete search spaces, several results have been achieved regarding the optimization of pseudo-Boolean functions. Following first results by Mühlenbein (19) and Rudolph (20), a wide range of such problems was covered by Droste et al. (6; 7) who successfully applied and considerably extended analytical methods from the field of randomized algorithms. Modeling the EA as a Markov process, Garnier et al. calculated the distribution of the first hitting time of the optimum for the CountOnes problem (9) and for long-path problems (8). He and Yao derived bounds for the expected running time using drift analysis (13) and exact expressions for the first hitting times of population-based EAs directly from the transition matrix of the associated Markov chains (14).

In the multiobjective case, only few theoretical results are available. Most of the corresponding studies were concerned with the limit behavior, i.e. the question whether the search algorithm converges, if the number of iterations goes to infinity (21; 22; 23; 24; 11; 12; 26; 16). Only recently, Scharnow et al. (25) provided a running time analysis of a (1+1)-EA on the shortest path problem and showed that a multiobjective formulation of the problem can reduce the time to find the single optimum considerably. The first running time analysis of population-based EAs on a multiobjective problem with conflicting objectives was given in (17) for a simple bi-objective model problem.

This paper contains running time results for different multiobjective EAs and for different problem scenarios. In particular, we

- Introduce two pseudo-Boolean model problems, which are scalable in the number of decision variables and number of objectives,
- Define simple individual-based and population-based multiobjective EAs,
- Propose methods, how population-based EAs can be analyzed in a multiobjective framework, and
- Present complexity results in terms of bounds of the expected running time of the different algorithms.

Besides these fundamental contributions, a further motivation for this analysis is to investigate whether the use of a population is beneficial in solving multiobjective problems: is a population-based EA searching concurrently for all optimal solutions in a single run more efficient, or represent multiple, separate runs of a (1+1)-EA searching for different optimal solutions a better strategy?

The paper is organized as follows. In section 2, we construct two multiobjective example problems. In section 3, we introduce and analyze SEMO; SEMO is the first example of a simple, parameterless population-based EA for multiobjective optimization problems. The analysis of SEMO reveals a similar performance on the bi-objective problems as a (1+1)-EA using multi-starts. In this context, we present a theorem that can help bounding the running time for a general class of elitist population-based EAs on multiobjective problems. In section 4, we propose two algorithmic improvements, a fair sampling strategy that accelerates the exploration of the optimal set, and a greedy selection mechanism that leads to a faster progress towards the optimal set. With these two improvements, implemented in the algorithms FEMO and GEMO, we are able to prove that population-based algorithms can actually have a lower running time than the (1+1)-EA. In section 5, the obtained results are generalized to higher dimensional objective spaces: We present and apply a theorem on a general graph search procedure that models the behavior of the fair sampling algorithms on the set of optimal solutions. Section 6 discusses the changes and difficulties when independent-bit mutations are used instead of one-bit mutations as considered before in the analysis of the different algorithms Finally, Section 7 summarizes the results obtained in this paper and discusses how these results can lead to a more efficient usage of multiobjective EAs for other problem domains.

2 Two Example Problems

The optimization problems considered in this paper are binary decision problems with n variables and $m \geq 2$ objectives. All objective functions are to be maximized. As there is no single search point that maximizes all components simultaneously, the goal is to find a set of so-called Pareto-optimal solutions as defined as follows.

Definition 1 (Pareto optimality)
Let $f : X \to F$ where $X \subseteq \{0,1\}^n$ is called decision space *and $F \subseteq \mathbb{R}^m$* objective space. *The elements of X and called* decision vectors *and the elements of F* objective vectors. *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* Pareto front.

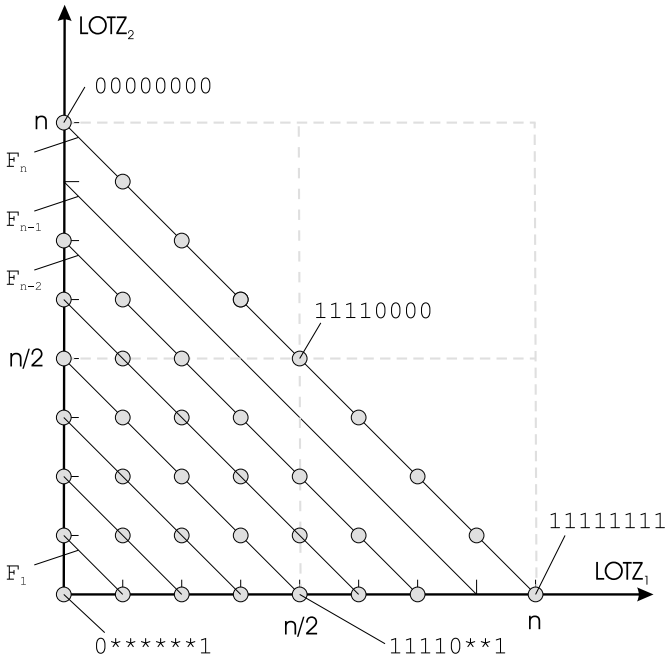

Figure 1: Objective space of the LOTZ problem with $n = 8$

In this paper, the running time of an algorithm equals the number of necessary evaluations of the objective function. As the algorithms defined in the following do not have an explicit stopping rule, we are interested in the running time until all elements of the Pareto front have been identified and are contained in the internal memory of the algorithm, together with one corresponding Pareto-optimal decision vector each.

The first example problem for this analysis is the LOTZ problem. 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 (20) and (7).

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

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

The objective space of LOTZ 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 $1^a0\,*^{(n-i-2)}\,10^b$ with $a+b=i$ for $i<n$, and 1^a0^b with $a+b=n$ for X_n. The asterisk (*) is used as a wildcard symbol and indicates that the corresponding bits can be chosen arbitrarily as zero or one.

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.

The LOTZ problem has a particular feature: All non-Pareto-optimal decision vectors only have one-bit Hamming neighbors that are either better or worse, but never incomparable to it. This fact facilitates the analysis of the population-based algorithms, which certainly cannot be expected from other multiobjective optimization problems. Therefore, we additionally present another simple multiobjective problem where this condition doesn't hold. This problem is a multiobjective extension of the CountOnes problem and is defined below. The problem consists of two parts: a cooperative part (the first half of the bit-string) and a conflicting part (the second half of the bit-string). In the cooperative part, the objective is to maximize the number of ones in both functions. In the conflicting part, the first objective is to maximize the number of ones and the second objective is to maximize the number of zeroes. The single-objective CountOnes problem has been extensively studied in the literature, see e.g. (19; 20; 6; 9).

Definition 3
The pseudo-Boolean function $COCZ : \{0,1\}^n \to \mathbb{N}^2$ *is defined as follows:*

$$COCZ(x_1, \ldots, x_n) = \left(\sum_{i=1}^{n} x_i, \quad \sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{n} (1 - x_i) \right)$$

where $n = 2 \cdot k$ *and* $k \in \mathbb{N}$.

Also for the COCZ, a partition of the objective space (see Fig. 2) provides an easy understanding of the problem. We distinguish $n/2+1$ sets $F_i, i = 0, \ldots, n/2$, where the index i corresponds to the number of ones in the first half of the bit-string. All F_i contain $n/2+1$ elements who distinguish themselves by the number of ones in the second half of the bit-string. $F_{n/2}$ represents the Pareto front F^*. The cardinality of the Pareto set $X^* = X_n$ is $|X_n| = 2^{n/2}$. However, more Pareto-optimal decision vectors map to objective vectors in the middle of the Pareto front than to its borders: while $\binom{n/2}{n/4}$ decision vectors map to the objective vector $(3n/4, 3n/4)$, the objective vectors $(n, n/2)$ and $(n/2, n)$ only have one corresponding element in decision space each.

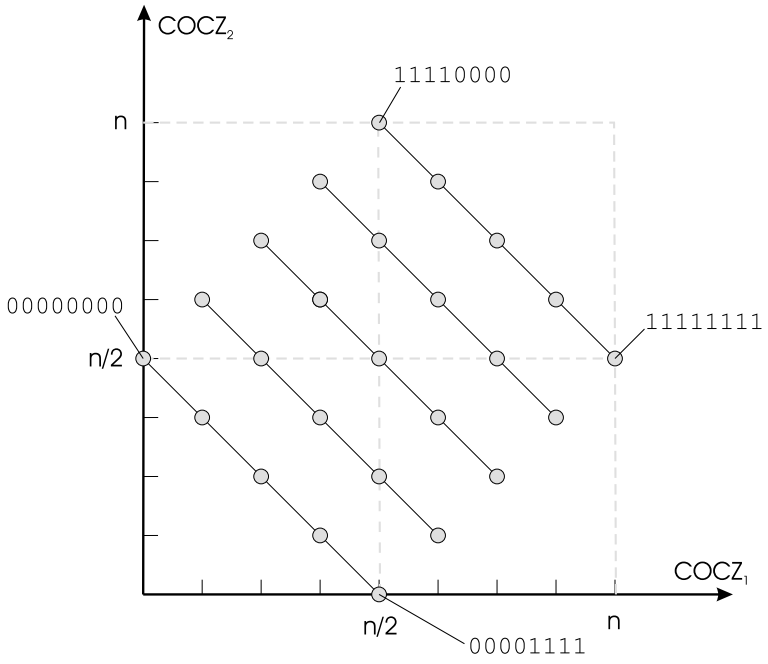

Figure 2: Objective space of the COCZ problem with $n = 8$

3 A Simple Evolutionary Multiobjective Optimizer

The field of evolutionary multiobjective optimization is characterized by a vast variety of EA variants with specialized operators of increasing complexity(5; 4). However, simple algorithms, which can serve as baseline algorithms for comparisons or theoretical analysis, are missing. The necessity for a baseline algorithm was already pointed out by (15) and motivated the invention of the Pareto Archived Evolution Strategy (PAES). The PAES makes use of a complex archiving and selection logic and is therefore difficult to analyze theoretically. We therefore propose and analyze a simple baseline algorithm, which can be seen a multiobjective generalization of a $(1 + 1)$-EA.

3.1 The SEMO

The Simple Evolutionary Multiobjective Optimizer (SEMO) represents the simplest instance of a population-based EA for multiobjective optimization. The SEMO contains a population of variable size that stores all individuals that are not dominated by any other individuals found so far. In each iteration, one parent individual x is drawn from this population uniformly at random and mutated. The child x' is added to the population, if it is not dominated by any population member and if its objective vector is not already contained in the population. All individuals that are dominated by the child are in turn deleted from the population.

Algorithm 1 Simple Evolutionary Multiobjective Optimizer (SEMO)

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

3.2 Analysis of SEMO on LOTZ

We start our analysis with the SEMO algorithm applied to the LOTZ problem. A run of the SEMO on LOTZ can be divided 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
The expected running time of Alg. 1 until the first Pareto-optimal point of LOTZ is found is $O(n^2)$.

Proof. 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 above 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
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 different sub-phases. Sub-phase i lasts from the time when i 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 $T = \sum_{i=1}^{n} T_i$, $1/8n^3 + 1/8n^2 \leq E(T) \leq 1/2n^3 + 1/2n^2$.

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

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

□

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

Theorem 1
The expected running time of SEMO on LOTZ is $\Theta(n^3)$.

3.3 Analysis of SEMO on COCZ and a General Upper Bound Technique

The analysis of the SEMO on the LOTZ problem has been facilitated by the observation that the population does not contain more that one individual until the Pareto set is reached. In this respect, the COCZ is a more realistic problem, because the population will certainly start growing before the Pareto set is reached. Unfortunately, this population growth is hard to analyze in detail and therefore hard to bound. In the worst case, the population might extend over the whole objective space and moves forward with a broad front of solutions. We can, however, derive the following general upper bound for this worst-case scenario, which holds not only for SEMO, but for a more general class of population-based approaches under the assumption of an elitist selection strategy. The idea behind this lemma is that it is sufficient for each decision vector to be mutated *once* into a dominating decision vector. The elitist selection strategy then guarantees that this decision vector will discarded from the population for ever. The result is independent of the sampling (or mating selection) strategy used and depends only on the properties of the variation and replacement selection operator, which can be formalized as:

Lemma 3 (General upper bound I)
Let an algorithm be given that iteratively modifies a population P *by a sequence of variation and selection operations with the properties*

V1 *For each* $y \in F \setminus F^*$, *the probability that the variation operator applied to any* $x \in X$ *with* $f(x) = y$ *produces a dominating decision vector* x' *with* $x' \succ x$ *is bounded below by* $p(y) > 0$

S1 *A new generated decision vector will enter the population P only if it is not dominated by any other element of P.*

S2 *A decision vector is deleted from the population P if and only if a dominating decision vector is included into the population.*

Then the expected number of times the variation operator is applied to non-Pareto optimal decision vectors is bounded above by $\sum_{y \in F \setminus F^*} p(y)^{-1}$.

Proof. Consider any objective vector $y' \in F \setminus F^*$ and the corresponding set of decision vectors $X' := f^{-1}(y) = \{x \in X | f(x) = y\}$. An element of X' can only undergo variation, if it is present in the population P. Let $T(X')$ denote the total number of times that elements from X' undergo variation until for the first time, an x'' is generated with $f(x'') \succ y'$. Clearly, x'' will enter P and cause all elements of $x' \in X'$ being deleted due to S2. With S1, this will further mean that no element of X' will ever be accepted again in the population, hence no elements of X' will be subject to variation anymore. As the random variables $T(X')$ are independent it follows with V1 that $E(T(X')) \leq p(y)^{-1}$. The summation over all $y' \in F \setminus F^*$ leads to the claimed expression. □

In many cases, the summation over all elements in $F \setminus F^*$ is impractical because its cardinality is too large. We can, however, also work with larger groups of decision vectors such that a smaller number of groups has to be accounted for. The following lemma is an extension of the previous one for arbitrary partitions of the decision space. It is similar to the fitness level technique (27) of single-objective problems and can be seen as a generalization of this technique for partially ordered objective spaces.

Lemma 4 (General upper bound II)
Let the dominated part of the decision space, $X \setminus X^*$ be partitioned into k sets $X_1, \ldots, X_k$ with $\bigcup_{1 \leq i \leq k} X_i = X \setminus X^*$ and $X_i \cap X_j = \emptyset$ for all i, j. Let the dominance relation on sets be defined as

$$X_i \succ X_j \Leftrightarrow \forall (a, b) \in X_i \times X_j : a \succ b.$$

$d(X_i) := \{X_j : X_j \succ X_i\}$ contain all sets X_j that dominate set X_i. If the algorithm fulfills the same properties as in Lemma 3 and $p(X_i)$ is a lower bound for the probability that a variation applied to an individual $x \in X_i$ produces an individual x' in a dominating decision space subset, i.e. $0 < p(X_i) \leq \min_{x \in X_i} \{Prob(x' \in d(X_i) | x \in X_i\}$ then the expected number of times the variation operator is applied to non-Pareto optimal decision vectors is bounded above by $\sum_{i=1}^{k} p(X_i)^{-1}$.

The proof can be done as for the previous lemma. With the help of this lemma it is now possible to prove upper bounds for different problems, given that an appropriate decision space partition and expressions for the improvement probabilities $p(\cdot)$ can be derived. We demonstrate its use by deriving an upper bound for the expected running time of the SEMO algorithm on the COCZ problem.

Theorem 2
The expected running time of SEMO applied to COCZ is bounded by $O(n^2 \log n)$.

Proof. We divide the total number of objective function evaluation into those that are required for mutants of non-Pareto-optimal parents and those for mutants of Pareto-optimal parents. Let the parents that are not Pareto-optimal be divided into group $X_{i,j} := \{x \in X | f(x) = (n/2 - i + j, n - i - j)\}$, $i, j \in \{0, \ldots, n/2\}$, where the i refers to the Hamming distance from the Pareto set and j to the number of ones in the bit-sting. It is obvious that the $X_{i,j}$ constitute a search space partition according to Lemma 4 and that the SEMO also fulfills the elitist selection conditions P1 and P2. As a result, the total number of mutations of non-Pareto-optimal search points can be bounded by $\sum_{j=0}^{n/2} \sum_{i=1}^{n/2} n/i = O(n^2 \log n)$.

Let k be the number of ones of the first Pareto-optimal point found. Now consider a modified SEMO that starts with this point and never accepts any non-Pareto-optimal points (their contribution to the running time has already been bounded) and never accepts any points with more than k ones. The expected time of this algorithm until the remaining Pareto-optimal points have been found is $\sum_{i=1}^{k} n(k - i)/i = \Theta(nk \log k)$. For an upper bound, consider the negative assumption that SEMO has to traverse the whole chain starting from $k = n/2$ twice, to move to both ends of the Pareto front. In every case, the claimed bound of $O(n^2 \log n)$ holds. □

3.4 Comparing SEMO to a (1+1)-EA using Multistarts

An alternative approach to find a set of Pareto-optimal solutions is the so-called scalarizing approach (18). The different objective functions are aggregated to form a single-objective surrogate problem, on which a single-objective optimizer can be applied. This scalarization involves parameters to be set that balance the relative importance of the different objectives. The working principle of this approach is to use multiple runs of the single-objective optimizer with different parameter settings such that in each run a different Pareto-optimal solution is found. It is an open problem whether this approach is more efficient compared to a population-based algorithm that is able to search for the whole Pareto front in a single run.

In this study, we want address this question by a theoretical analysis of the different approaches. As a representative for the single-objective multistart class we use the ϵ-constraint method (10). The ϵ-constraint method works by choosing one objective function as the only objective and the remaining objective functions as constraints. By a systematic variation of the constraint bounds, different elements of the Pareto front can be obtained (3, p. 285) by solving the constrained single-objective problems

$$\max \quad f_1 \tag{2}$$

$$\text{s.t.} \quad f_i \geq \epsilon_i \quad \forall 2 \leq i \leq m, i \in \mathbb{N}. \tag{3}$$

The (1+1)-EA therefore needs a set of different ϵ_i values to be specified. For the following analysis we optimistically assume a best-case scenario, where

- The coordinates of all Pareto-optimal objective vectors $f^{*1}, f^{*2}, \ldots, f^{*|F^*|}$ are known.
- The single-objective (1+1)-EA, Alg. 2, is run for each of those $f^{*k}, 1 \leq k \leq |F^*|$, using the constraint bounds $\epsilon_i = f_i^{*k}$.
- Each of the $|F^*|$ runs of the (1+1)-EA is independently executed until the optimum is found.

Using information about the optimum might appear unfair, but we are mainly interested in lower bounds for the multistart (1+1)-EA. It is obvious that a realistic variant cannot be better than this one as it has to make at least one run for each Pareto-optimal objective vector to be found, plus a number of run that are redundant because of an non-optimal choice of the ϵ_i values.

Algorithm 2 Single-objective (1+1)-EA

```
1: Choose an initial individual x uniformly from X
2: loop
3:    Create offspring x' by mutation of x.
4:    g(x) ← Σ_{f_i ≤ ε_i, 2 ≤ i ≤ m} (ε_i − f_i(x))        {constraint violation by x}
5:    g(x') ← Σ_{f_i ≤ ε_i, 2 ≤ i ≤ m} (ε_i − f_i(x'))      {constraint violation by x'}
6:    if g(x') < g(x) ∨ (g(x') = g(x) = 0 ∧ f_1(x') > f_1(x)) then
7:       x ← x'
8:    end if
9: end loop
```

The (1+1)-EMO proceeds by first minimizing the constraint violation given by the second objective and the different thresholds c. After the constraint is satisfied, the algorithm turns to optimize the first objective while keeping the second objective value above the constraint boundary. We give an exact expression for the expected running time of the (1+1)-EMO on the LOTZ and the COCZ problem, the proof is given in the appendix.

Theorem 3
The expected running time of the multistart (1+1)-EA is $\frac{1}{2}(n^3 + n^2)$ *on the LOTZ problem and* $\Theta(n^2 \log n)$ *on the COCZ problem.*

It follows that running time of the simple population-based optimizer SEMO for both the LOTZ and the COCZ problem is of the same order as the multistart (1+1)-EA in its base-case scenario. In the next section, we investigate whether a population-based approach can be provably better than the (1+1)-EA.

4 Two Improved Evolutionary Multiobjective Optimizers

Obviously, SEMO is not advantageous to the (1+1)-EA using an ϵ-constraint method regarding the LOTZ and the COCZ problem. In this section, we propose two modifications of the SEMO algorithm, a fair sampling strategy and a greedy selection mechanism. With the fair sampling strategy, the FEMO (Fair Evolutionary Multiobjective Optimizer) can solve the LOTZ problem quicker. Using in addition a greedy selection mechanism, the GEMO (Greedy Evolutionary Multiobjective Optimizer) is also quicker on the COCZ problem.

4.1 The FEMO and the Fair Sampling Strategy

The main weakness of the SEMO on the LOTZ problem lies in the fact that a large number of mutations are allocated to parents 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, leading to a running time of $\Theta(n^2)$. 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 *fair* sampling strategy implemented by FEMO guarantees that at the end all individuals receive about the *same* number of samples.

The FEMO (see Alg. 3) implements a fair selection 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).

Algorithm 3 Fair Evolutionary Multiobjective Optimizer (FEMO)

```
1:  Choose an initial individual x uniformly from X
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 mutation of x.
8:     if ∄z ∈ P such that (z ≻ x' ∨ f(z) = f(x')) then
9:        P ← (P \ {z ∈ P | x' ≻ z}) ∪ {x'}
10:       w(x') ← 0                                   {Initialize offspring count}
11:    end if
12: end loop
```

For the analysis of the FEMO on LOTZ we note that the first phase is identical to

the SEMO described before. With the following theorem we prove that the total running time of FEMO on LOTZ is bounded by $\Theta(n^2 \log n)$.

Theorem 4
With probability at least $1 - O(1/n)$*, the running time Algorithm 3 needs from the discovery of the first two Pareto-optimal objective vectors of the LOTZ problem 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,* $E(T) = O(1/p \cdot n \log n)$*.*

The proof of this theorem is given in the appendix. The idea is to bound the number times each element that enters P in the second phase will be mutated. Once the first Pareto-optimal points is found, there is exactly one possible parent for each of the remaining n points. We are interested in the number of mutations that must be allocated to *each* of these n parents in order to have at least one successful mutation that leads to the desired child. Lemma 7 and 8 provide upper and lower bounds on the probability that a certain number of mutations per parent are sufficient. In Theorem 4 these probabilities are used to bound the running time of the FEMO algorithm.

As discussed earlier, the COCZ problem leads to a possible growth of the population before the Pareto set is found, for the SEMO as well as for the FEMO. This makes it a difficult challenge to derive tight upper and lower bounds for the FEMO on COCZ. Nevertheless, Lemma 4 and the same argumentation as with the SEMO leads to a $O(n^2 \log n)$ upper bound. For the lower bound it is sufficient to note that in order to find the last Pareto-optimal objective vector there is only one bit that must flip, the chance of which is $1/n$. Due to the fair sampling strategy, FEMO will therefore allocate $\Omega(n)$ mutation trials for all $n/2$ other Pareto-optimal members of the population. In summary:

Theorem 5
The expected running time of FEMO applied to COCZ is bounded above by $O(n^2 \log n)$ *and below by* $\Omega(n^2)$

In the next section we propose a further improvement of the algorithm, which avoids the population growth as much as possible.

4.2 The GEMO and the Greedy Selection Mechanism

The Greedy Evolutionary Multiobjective Optimizer (GEMO) is an extension of the FEMO in order to achieve maximum progress towards the Pareto front. The main idea is to allocate all search effort to offspring of the most recently successful mutant, which is implemented as follows.

As long as only mutually non-dominating individuals are found, the algorithm acts like FEMO, in order to spread out the population, and hence the search effort, fairly

and equally. However, when further progress towards the Pareto front is achieved (realized by the fact that a new individual is found that dominates elements of the current population), all other remaining population members are disabled. This means that they cannot produce any offspring for the time being and is implemented by setting their weight to infinity (line 18). When GEMO finally reaches the Pareto front and no further progress is possible, it will again behave like FEMO. Here, it is necessary to re-enable all re-discovered individuals (line 11), otherwise these individuals would constitute barriers in the objective space that are difficult or, depending on the mutation distribution, impossible to cross.

Algorithm 4 Greedy Evolutionary Multiobjective Optimizer (GEMO)

```
 1: Choose an initial individual x uniformly from X
 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 mutation of x.
 8:    if ∄z ∈ P such that z ≻ x' then
 9:       if ∃z ∈ P such that f(z) = f(x') then
10:          if w(z) = ∞ then
11:             w(z) ← 0                              {Reset offspring count}
12:          end if
13:       else
14:          D ← {z ∈ P|x' ≻ z}                      {Determine individuals dominated by x'}
15:          if D ≠ ∅ then
16:             P ← P \ D                            {Delete dominated individuals}
17:             for all y ∈ P do
18:                w(y) ← ∞                          {Disable remaining individuals}
19:             end for
20:          end if
21:          P ← P ∪ {x'}                            {Add x' to population}
22:          w(x) ← 0                                {Initialize offspring count}
23:       end if
24:    end if
25: end loop
```

Due to the special characteristics of the LOTZ problem, the GEMO behaves identically to the FEMO here. On the COCZ problem, the new features of the GEMO algorithm allow us to prove a tight bound of the expected running time.

Theorem 6
The expected running time of GEMO applied to COCZ is bounded by $\Theta(n^2)$.

The proof, which is given in the appendix, is again split into the two phases. The

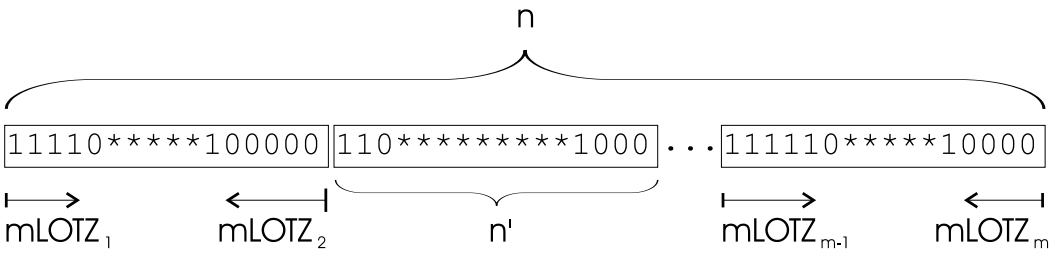

Figure 3: Schematic view of the mLOTZ problem

idea is that the new greedy selection of the last improved individual reduces the time needed on the way towards the Pareto front considerably.

5 Higher-dimensional Objective Spaces

In this section, we generalize the two bi-objective problems from the previous sections, LOTZ and COCZ, to arbitrary objective space dimensions. Bounds for the expected running time for the different algorithms are derived for each problem, where the problem size is again determined by the number of decision variables n, while the number of objectives m is considered as a constant. To facilitate the analysis of the fair sampling strategy of the GEMO, we derive a general result on a graph searching process, which serves as a model how GEMO behaves on the Pareto front.

5.1 The Multiobjective Leading Ones (mLOTZ) Problem

The LOTZ problem can be generalized to an arbitrary even number of objectives m by concatenating $m/2$ bi-objective LOTZ problems of $2n/m$ bits each.

Definition 4
The pseudo-Boolean function $mLOTZ : \{0,1\}^n \to \mathbb{N}^m$ *is defined as follows:*

$$mLOTZ(x_1, \ldots, x_n) = (f_1, f_2, \ldots, f_m)$$

with

$$f_k = \begin{cases} \sum_{i=1}^{n'} \prod_{j=1}^{i} x_{j+n'(k-1)/2} & \text{if } k \text{ is odd} \\ \sum_{i=1}^{n'} \prod_{j=i}^{n'} (1 - x_{j+n'(k-2)/2}) & \text{else.} \end{cases}$$

where $m = 2 \cdot m', m' \in \mathbb{N}$ *and* $n = m' \cdot n', n' \in \mathbb{N}$.

The construction principle of the mLOTZ problem is depicted in Fig. 3. To solve mLOTZ, we are searching for a representation of the Pareto front with $|F^*| = (2n/m + 1)^{m/2}$ elements.

Theorem 7
The expected running time of the (1+1)-EMO applied to mLOTZ is bounded by $\Theta(n^{m/2}n^2)$.

Proof. For each of the $|F^*|$ Pareto-optimal points, we have to find a unique bit-string. For each mutation, there are at least one and at most m bits that can flip for a success. In addition, a bit is set correctly with probability 1/2 even without mutation, so half of the steps are for free. Hence, the expected running time for each constrained sub-problem is $\Theta(n^2)$, and with the number of $|F^*| = (2n/m+1)^{m/2}$ sub-problems to be solved the claim follows. □

To derive an upper bound for the SEMO, we can again make use of the general upper bound of Lemma 4.

Theorem 8
The expected running time of the SEMO and the FEMO applied to mLOTZ is bounded by $O(n^{m+1})$.

Proof. We have $O((n/m)^m)$ different objective vectors. Each of the dominated vectors receives on average $O(n)$ mutations until it is improved and deleted for ever. The discovery of the last Pareto-optimal vector takes $O(n/m)^{m/2} \cdot n$ time, which is an upper bound the discovery of all Pareto-optimal vectors. □

For the analysis of the GEMO we again note that the running time is mainly determined by the exploration of the Pareto front. In the general case, the Pareto front can be modeled as a graph, where the nodes correspond to the different Pareto-optimal objective vectors and the directed weighted edges correspond to mutation probabilities. Instead of analyzing GEMO directly, we first define and analyze a more general randomized graph search algorithm, which is similar to a process described and analyzed in (1). The purpose of this approach is two-fold. First, it gives a more intuitive view of how the different population-based algorithms behave on the Pareto front. Second, it provides a general tool, like the upper bound technique of Lemma 4, which facilitates and shortens the analysis of different real algorithms.

The algorithm assumes that we are given a random starting node v and a random operator $\text{jump} : \text{V} \mapsto \text{V}$, which returns for each node v a neighbor v' of v with probability $w(v, v')$. The purpose of the following algorithm is to determine V and E using a minimal number of calls to jump.

It will be shown that the above algorithm explores a graph G using $O(\frac{|V|}{p} \log |E|)$ calls to the function jump with high probability, where p is a lower bound on the edge weights. The first lemma bounds the weight of a node during a run of the algorithm.

Algorithm 5 Randomized Graph Search

1: $w(v) \leftarrow 0$
2: $V \leftarrow \{v_1\}$; $E \leftarrow \{\}$
3: **loop**
4: Select a node v out of $\{v' \in V | w(v) \leq w(v')\ \forall v' \in V\}$ uniformly.
5: $w(v) \leftarrow w(v) + 1$
6: $v' \leftarrow \text{jump}(v)$
7: **if** $v' \notin A$ **then**
8: $w(v) \leftarrow 0$
9: $V \leftarrow V \cup \{v'\}$; $E \leftarrow E \cup \{(v, v')\}$
10: **end if**
11: **end loop**

Lemma 5
With probability at most $\Delta \cdot e^{-\lambda}$, the weight of a node with Δ neighbors exceeds $\frac{\lambda}{p}$ before all neighbors of the node are found.

Proof. Starting from $w(v) = 0$, the weight increases by 1 after each trial. Let us suppose that the neighbors of v are $\{v_1, ..., v_\Delta\}$. The weight exceeds $\frac{\lambda}{p}$ if it exceeds this value before neighbor v_1 is found *or* before neighbor v_2 is found *or* ... Using the Boole-Bonferroni inequality, we can bound the probability P we are looking for, according to

$$P \leq \sum_{k=1}^{k=\Delta} P_k = \Delta \cdot P_1$$

where P_k denotes the probability that neighbor v_k has not been found yet and $w(v)$ exceeds $\frac{\lambda}{p}$. Now we have

$$P_1 = (1-p)^{\frac{\lambda}{p}} \leq e^{-\lambda} \qquad P \leq \Delta \cdot e^{-\lambda}$$

□

Lemma 6
With probability at most $|E| \cdot e^{-\lambda}$, the weight of some node exceeds $\frac{\lambda}{p}$ before all nodes and edges are found.

Proof. There are N nodes that must find all their neighbors. The events that the weight of a node exceeds $\frac{\lambda}{p}$ are independent for all nodes. Using the number of neighbors Δ_i of node v_i, we find the probability

$$\sum_{i=1}^{i=|V|} \Delta_i \cdot e^{-\lambda} = |E| \cdot e^{-\lambda}$$

□

Now, we can bound the total running time of the exploration algorithm.

Theorem 9
With probability at least $1 - \frac{1}{|E|^c}$, *the above algorithm explores all nodes and edges of* G *after* $(c+1)\frac{|V|}{p}\log|E|$ *calls to the function* jump.

Proof. With $\lambda = (c+1)\log|E|$ and Lemma 6 we find that with probability $1 - \frac{1}{|E|^c}$, the maximal weight of the nodes in V does not exceed $\frac{(c+1)}{p}\log|E|$ at the time when the whole graph has been explored. Therefore, the function jump has been called at most $|V|\frac{(c+1)}{p}\log|E|$ times. □

Theorem 10
The running time of the GEMO applied to mLOTZ is bounded by $O(n^{m/2} \cdot \frac{m}{2}n\log n) = \Theta(n^{m/2}n\log n)$ *with probability at least* $1 - O(n^{-m})$.

Proof. Let us again consider two phases depending on whether a Pareto-optimal point has been found so far. In the first phase, there are two types of mutations that lead to accepting new points: successful mutations and indifferent mutations. Successful mutations cause the active population to shrink to one element x and increase the objective sum $s = \sum_{i=1}^{m} f_i(x)$ by one. Indifferent mutations can cause the population to grow, but keep s constant. Therefore, at most n successful mutations are needed in the first phase. Since the probability of a successful mutation is always at least $1/n$, the expected running time of the first phase is $O(n^2)$. At the end of the first phase, P contains exactly one element of the Pareto set, and we can describe the behavior in phase 2 by the the model of Alg. 5. In this case, $|V| = (2n/m+1)^{m/2}$, $|E| \leq m|V|$ and $p = 1/n$. Application of Theorem 9 leads to the claimed bound. □

5.2 The Multiobjective Count Ones Problem (mCOCZ) Problem

The idea of the mCOCZ is again a concatenation of multiple bi-objective COCZ problems. However, the bits shall be re-arranged so that the first $n/2$ bits represent the cooperative part, where the number of ones contributes equally to all objectives. The remaining parts represent the mutual tradeoffs between pairs of objectives.

Definition 5
The pseudo-Boolean function $mCOCZ : \{0,1\}^n \to \mathbb{N}^m$ *is defined as follows, where* $n' = n/m$ *:*

$$mCOCZ(x_1, \ldots, x_n) = (f_1, f_2, \ldots, f_m)$$

with

$$f_j = \sum_{i=1}^{n/2} x_i + \begin{cases} \sum_{i=1}^{n'} x_{i+n/2+(j-1)n'/2} & \text{if } j \text{ is odd} \\ \sum_{i=1}^{n'} (1 - x_{i+n/2+(j-2)n'/2}) & \text{else.} \end{cases}$$

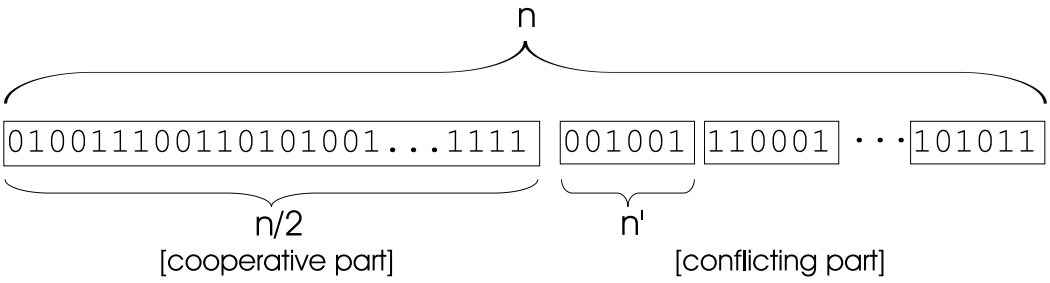

Figure 4: Schematic view of the mCOCZ problem

where $m = 2 \cdot m', m' \in \mathbb{N}$ *and* $n = m \cdot n', n' \in \mathbb{N}$.

This gives rise to a Pareto front with $|F^*| = (n/m + 1)^{m/2}$ different elements. The construction principle of the mCOCZ problem is depicted in Fig. 4.

Theorem 11
For the mCOCZ problem, the following bounds for the expected running times hold:

- $\Theta(n^{m/2} n \log n)$ *for the (1+1)-EMO*
- $O(n^{m+1})$ *for the SEMO and the FEMO*

The running time of the GEMO applied to mCOCZ is bounded by $\Theta(n^{m/2} n \log n)$ *with probability at least* $1 - O(n^{-m})$.

The proof of Theorem 11 is given in the appendix.

6 From One-bit to Independent-bit Mutations

So far, all algorithms in this paper have been considered with one-bit mutations only as this simplifies the analysis. The disadvantage is that the one-bit mutation operator is less relevant in practise, because it is a local operator and will fail on problems, where jumps of more than one bit are required to proceed the search.

In this section, we discuss the use of the independent-bit mutation operator, i.e. in a mutation step each bit is flipped independently with probability $1/n$. We claim that upper bounds derived by the general technique of Lemma 4 also hold in the case of independent-bit mutations. Our argumentation relies mainly on the success probabilities. We can simply disregard all mutations where not exactly one bit flips. The expected waiting time from one one-bit mutation to the next is $(1 - 1/n)^{-(n-1)} = O(e)$, which doesn't change the order of the total expected running

time. Lemma 4 then assures us that we can work with the success probabilities alone and completely ignore those mutations, where not exactly one bit flips.

The situation is different when we try to verify more tighter bounds, like only for the FEMO on the LOTZ problem. Here, the independent-bit mutations can produce incomparable individuals and hence cause the population to grow before it reaches the Pareto set. This problem is somehow similar to the one encountered in the analysis of FEMO on COCZ with the one-bit mutations, and it is a so far unsolved challenge to prove tight bounds for both cases.

The effect of mutations that flip more than one bit has to be taken into account also for the GEMO algorithm. Here, we are faced with the following type of negative event that might enlarge the running time: Offspring can be accepted into the population with a larger Hamming distance from the Pareto set than their parent. If this offspring then happens to be the first in the population to produce a dominating child, then the minimum Hamming distance of the active part of the population can increase. It is therefore necessary to bound the negative effect of this event.

Altogether it can be stated that bounds for the case of independent-bit mutations can be derived for all scenarios considered in this paper, but that it remains open whether these bounds are tight.

7 Summary and Discussion

In this paper, running time results were derived for different types of evolutionary algorithms on different pseudo-Boolean multiobjective optimization problems. To this end, we proposed and analyzed various population-based multiobjective EAs:

- The simple population-based algorithm SEMO, and
- Two improved population-based algorithms, FEMO and GEMO

and compared their running time against a multi-start (1+1)-EA based on the ϵ-constraint method. To facilitate the analysis of population-based EA, we derived two analytical tools,

- A general upper bound technique based on a partition of the decision space and
- A general randomized graph search algorithm.

While the former can be used to bound the search effort in to non-Pareto optimal regions, the latter helps to analyze the time spent for the exploration of the Pareto front itself.

	(1+1)-EMO	SEMO	FEMO	GEMO
LOTZ	$\Theta(n^3)$	$\Theta(n^3)$	$\Theta(n^2 \log n)$	$\Theta(n^2 \log n)$
COCZ	$\Theta(n^2 \log n)$	$O(n^2 \log n)$	$O(n^2 \log n)$	$\Theta(n^2)$
mLOTZ	$\Theta(n^{m/2} n^2)$	$O(n^{m+1})$	$O(n^{m+1})$	$O(n^{m/2} n \log n)$
mCOCZ	$\Theta(n^{m/2} n \log n)$	$O(n^{m+1})$	$O(n^{m+1})$	$O(n^{m/2} n \log n)$

Table 1: Bounds on the expected running time derived in this paper.

We additionally defined two model problems, mLOTZ and mCOCZ, which are composed of unimodal pseudo-Boolean functions and which are scalable in the number of objectives m.

The bounds on the expected running times are summarized in Table 1. It was shown that on the bi-objective problems, the SEMO needs a running time of the same order as the (1+1)-EA, while it loses its competitiveness with increasing number of objectives. This is mainly due to SEMO's poor ability to expand the population on the Pareto set. The GEMO on the other hand has the lowest running time on all problems, except for the mCOCZ for $m > 2$, where an upper bound of the same order as for the (1+1)-EA could be given, but which is not necessarily a tight bound.

A major difficulty in the analysis of the population-based algorithms is the handling of the population size. In order to be able to store the whole Pareto front, a large enough population size must be allowed, implying that in the worst case all possible objective vectors will be visited. If is it not possible to bound the size of the population on its way towards the Pareto set, the running time advantage compared with the (1+1)-EA vanishes. The strategy of the GEMO to focus the search effort on the most recently successful offspring counteracts the tendency of the population to increase, as long as successful mutations are possible. The GEMO can therefore be viewed as an algorithm that implicitly notices when the Pareto front is reached and automatically adapts its behavior. The rationale behind the GEMO strategy is that we first want to find the Pareto set on the straightest way possible and only thereafter to spread out the population as quick as possible. This way, the GEMO represents the opposite of the (1+1)-EA, which searches in all different directions from the start, while the behavior of the FEMO and the SEMO is somehow in between these two extremes.

There certainly exist multiobjective problems where it is not easy to traverse the Pareto set, e.g. if its elements are at a large Hamming distance from each other. In such a case it might be necessary to spread out the population early in the search in order to make all Pareto-optimal points reachable. Clearly, the GEMO strategy would be wrong here, because it was designed to prevent exactly this. Nevertheless, it can be argued that such a scenario is anyway the domain of scalarizing approaches like the multi-start (1+1)-EA presented in this paper, and that any approach with a population would be less efficient.

8 Conclusion

We have shown for two types of problems that a population-based multiobjective optimizer has a provable lower running time than traditional scalarizing methods using multistarts of single-objective optimizers. So far it has often been claimed, but never shown, that the use of a population is beneficial in the context of multiobjective optimization, even without the use of recombination. The results presented in this paper are the first theoretical evidence for this claim. Instead of the explicit co-operation using traditional recombination operators, implicit cooperation via measuring of success in the population is used here.

The FEMO algorithm was invented as an optimal strategy for searching homogeneous graphs. If the Pareto set of a multiobjective optimization problem has such a structure, as in the examples presented here, the FEMO will traverse it efficiently. The FEMO, however, loses its efficiency in cases where the population spreads out quickly before the Pareto set is reached. This problem is circumvented by GEMO algorithm, where a 'greedy' selection strategy maintains maximum progress towards the Pareto set.

One of the many challenging problems in this area is to derive tight bounds for the cases where the population of the multiobjective EA spreads out before the Pareto set is reached.

Appendix

Proof of Theorem 3. As the Pareto front of the LOTZ problem contains $n+1$ different elements, the algorithm has to solve $n+1$ sub-problems with different constraint values. Let $T_i, 0 \leq i \leq n$, denote the running time to solve the i-th sub-problem whose optimum is given by the point with $n-i$ leading ones and i trailing zeroes. Each sub-problem can be further divided into n consecutive sub-phases, the first i of which correspond to finding the trailing zeroes starting from the back and the following $n-i$ by finding the leading ones starting from the front. At the beginning of each sub-phase, the bit under concern is with equal probability 0 or 1. If the bit is already set correctly, this sub-phase has length 0, otherwise it follows a geometric distribution with parameter $(n-1)/n$. Hence, all sub-phases are independent and identically distributed with expectation $n/2$. Therefore,

$$E(T) = \sum_{i=0}^{n} E(T_i) = (n+1)\frac{n^2}{2} = \frac{1}{2}(n^3 + n^2)$$

The Pareto front contains $n/2+1$ different elements, therefore the algorithm has to solve $n/2+1$ sub-problems with different constraint values. In each of the runs, a bit-sting has to be produced where the first $n/2$ bits, the cooperative section, are ones. We call this the end of phase 1. Its expected time is $\Theta(n \log n)$ as it is equivalent to solving the CountOnes problem with $n/2$ bits (6) and taking into

account that on average every second mutation will be realized in the first $n/2$ bits. For the upper bound, consider the worst case, where at the end of the first phase the first $n/2$ bits are set to one and the rest can be any string. Now there are two alternatives. If the constraint is not satisfied, then some of the ones must be mutated to zero. If the constraint is already satisfied, then it might be possible to turn some of the zeroes into ones. In both cases, there are at most $n/2$ such candidate bits, which can flip at a random order, and the expected waiting time for this event is again bounded by $O(n \log n)$. Thus, the expected time for solving one sub-problem is $\Theta(n \log n)$ and with $n/2+1$ such problems to solve gives a total expected running time of $\Theta(n^2 \log n)$. □

Lemma 7
Let the population P of FEMO applied to LOTZ contain exactly one Pareto-optimal solution and let $c > 0$ be an arbitrary constant. With probability at least $1 - n^{1-c}$, it takes at most $c \cdot n \log n$ mutation trials per solution to generate all remaining n Pareto-optimal solutions.

Proof. For each individual, the probability that its parent did not generate it within its first $c \cdot n \log n$ mutations is bounded by

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

There are n 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 $c \cdot n \log n$ trials is bounded above by n^{1-c}. □

Lemma 8
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 $c \cdot n \log n$ 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 $t = c \cdot n \log n$ mutations is $1 - (1-1/n)^t$. The probability that this happens independently for a selection of k such parent-offspring combinations can thus be bounded as

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

□

Proof of Theorem 4. 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 7 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 8 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 7 the expected running time can be bounded with $T' = pT/(n \log n)$ and

$$\begin{aligned} E(T') &\leq 1 \cdot P\{0 \leq T' < 1\} + 2 \cdot P\{1 \leq T < 2\} + \ldots \\ &\leq 2 + \sum_{c=3}^{\infty} c \cdot P\{T' \geq c - 1\} \\ &\leq 2 + \sum_{c=1}^{\infty} (c+2) n^{-c} \\ &\leq 2 + \frac{n}{(n-1)^2} + \frac{2}{n-1} \end{aligned}$$

as $E(T) = O(1/p \cdot n \log n)$. □

Proof of Theorem 6. Consider again the two successive phases, where the first phase ends when the first Pareto-optimal point is found. The algorithm starts with a random point. A mutation can cause a step towards the Pareto front (successful step) if in the first half of the bit-sting a zero is flipped, or an incomparable step if any bit in the second half is flipped. If a one in the first half flips, nothing happens, because the mutant is dominated by the parent. If an incomparable mutant is not already in the population, it will be accepted (line 21) while its parent remains, causing the population to grow. A successful mutation leading to a dominating child, however, will delete its parent and all other dominated individuals (line 16). The remaining individuals in the population are temporarily disabled from offspring production (line 18). Now consider the Markov chain with $n/2+1$ states, where the state is given by the number of zeroes in the first half of all population members with $w < \infty$. The only possible transitions are from state i to $i-1$, which happen with probability i/n. The expected time to absorption into state 0, and hence the time to reach the Pareto set, is bounded by $\Theta(n \log n)$.

For the second phase we proceed similarly to the analysis of FEMO on LOTZ and claim the following: If we allocate $c \cdot n$ mutation trials to each element of the Pareto front, each one will have produced its neighbor with a probability depending on c. The failure probability, i.e. the probability that we are not ready after all these cn^2 mutations in total, can thus be bounded above by

$$\sum_{i=1}^{n} \left(1 - \frac{i}{n}\right)^{cn} \leq \sum_{i=1}^{n} \left(\frac{1}{e^c}\right)^{i} \leq \frac{1}{e^c - 1} \leq \left(\frac{1}{2}\right)^{c-1}.$$

To bound the expected total number of mutations T we use the substitution $T' = T/n^2$, and

$$\begin{aligned} &E(T') \leq \textstyle\sum_{c=1}^{\infty} c \cdot P\{c-1 \leq T' \leq c\} \leq \sum_{c=1}^{\infty} c \cdot P\{T' \geq c-1\} \\ &\leq \textstyle\sum_{c=1}^{\infty} c \cdot \left(\frac{1}{2}\right)^{c-2} = 4 \sum_{c=1}^{\infty} c \cdot \left(\frac{1}{2}\right)^{c} = 8 \end{aligned}$$

leads to an upper bound of $E(T) = 8n^2$ for the second phase. □

Proof of Theorem 11. The (1+1)-EMO has to solve $|F^*| = (n/m+1)^{m/2}$ constrained sub-problems. For each sub-problem we have to at least solve a CountOnes problem in the first $n/2$ bits, which takes $\Theta(n \log n)$ time. This already proves the lower bound. For the upper bound we pessimistically assume that we then optimize each of the $m/2$ trade-off sections separately. As all bits within the same section are interchangeable, this is equivalent with solving $m/2$ CountOnes problems of n/m bits, which again takes on average $\Theta(n \log n)$ time and proves the upper bound.

For SEMO and FEMO, we again apply Lemma 4 with a search space partition where each objective vector constitutes its own subset. In each subset, let k denote the Hamming distance from the Pareto set, i.e. the number of zeros among the first $n/2$ bits. For each k there are $|F^*|$ subsets in the partition whose elements have a Hamming distance k to the Pareto set and whose probability of improvement p_k equals k/n (flipping one of the zeroes in the first half of the bit-string). With Lemma 4 we can bound the number of mutations allocated to non-Pareto optimal points by $O(n^{m/2} n \log n)$. On the Pareto front, we want to bound the expected time to find the k-th element under the conditions that $k-1$ elements are already found. Such a bound is given $O(kn)$, and the summation over all k from 1 to $|F^*|$ leads to a total expected running time of $O(n^{m+1})$.

For GEMO, let us again consider two phases depending on whether a Pareto-optimal point has been found so far. In the first phase, there are two types of mutations that lead to accepting new points: successful mutations and indifferent mutations. Successful mutations cause the active population to shrink to one element x and increase the objective sum $s = \sum_{i=1}^{m} f_i(x)$ by one. Indifferent mutations can cause the population to grow, but keep s constant. Therefore, at most n successful mutations are needed in the first phase. Since the probability of a successful mutation is always at least $1/n$, the expected running time of the first

phase is $O(n^2)$. At the end of the first phase, P contains exactly one element of the Pareto set, and we can describe the behavior in phase 2 by the the model of Alg. 5. In this case, $|V| = (n/m+1)^{m/2}$, $|E| \leq m|V|$ and $p \geq 1/n$. Application of Theorem 9 leads to the claimed bound. □

Acknowledgments

The research has been funded by the Swiss National Science Foundation (SNF) under the ArOMA project 2100-057156.99/1. We are grateful to Emo Welzl, Ingo Wegener and Oliver Giel for numerous discussions on the subject.

References

[1] N. Alon. A random process for searching a graph (comment). Personal communication, 2002.

[2] H.-G. Beyer, H.-P. Schwefel, and I. Wegener. How to analyse evolutionary algorithms. *Theoretical Computer Science*, 287:101 – 130, 2002.

[3] V. Chankong and Y.Y. Haimes. *Multiobjective Decision Making Theory and Methodology*. Elsevier, 1983.

[4] C. A. Coello Coello, D. A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, New York, 2002.

[5] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester, UK, 2001.

[6] 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.

[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] J. Garnier and L. Kallel. Statistical distribution of the convergence time of evolutionary algorithms for long-path problems. *IEEE Transactions on Evolutionary Computation*, 4(1):16 – 30, 2000.

[9] J. Garnier, L. Kallel, and M. Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7(2):173–203, 1999.

[10] Y.Y. Haimes, L.S. Lasdon, and D.A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:296 – 297, 1971.

[11] T. Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal Of Operational Research*, 117(3):553–564, 1999.

[12] 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.

[13] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:57 – 85, 2001.

[14] J. He and X. Yao. From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):495 – 511, 2002.

[15] J. D. Knowles and D. W. Corne. Approximating the non-dominated front using the Pareto Archived Evolution Strategy. Technical Report RUCS/1999/TR/005/A, Department of Computer Science, University of Reading, UK, 1999.

[16] M. Laumanns, L. Thiele, E. Zitzler, and K. Deb. Archiving with guaranteed convergence and diversity in multi-objective optimization. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 439–447, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[17] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Parallel Problem Solving From Nature — PPSN VII*, 2002.

[18] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer, Boston, 1999.

[19] H. Mühlenbein. How genetic algorithms really work: I. mutation and hillclimbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 15–25, Amsterdam, 1992. Elsevier Science.

[20] G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač, Hamburg, 1997.

[21] G. Rudolph. Evolutionary search for minimal elements in partially ordered finite sets. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Evolutionary Programming VII, Proceedings of the 7th Annual Conference on Evolutionary Programming*, pages 345–353, Berlin, 1998. Springer.

[22] 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.

[23] 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.

[24] 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.

[25] J. Scharnow, K. Tinnefeld, and I. Wegener. Fitness landscapes based on sorting and shortest paths problems. In *Parallel Problem Solving From Nature — PPSN VII*, 2002.

[26] 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.

[27] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In R. Sarker, X. Yao, and M. Mohammadian, editors, *Evolutionary Optimization*, pages 349–369. Kluwer, 2000.