On The Effect of Populations in Evolutionary Multi-objective Optimization

Oliver Giel[*]
Fachbereich Informatik, Lehrstuhl 2
Universität Dortmund
44221 Dortmund, Germany
oliver.giel@cs.uni-dortmund.de

Per Kristian Lehre[†]
Dept. of Computer and Information Science
Norwegian University of Science and Technology
7491 Trondheim, Norway
lehre@idi.ntnu.no

ABSTRACT

Multi-objective evolutionary algorithms (MOEAs) have become increasingly popular as multi-objective problem solving techniques. An important open problem is to understand the role of populations in MOEAs. We present a simple bi-objective problem which emphasizes when populations are needed. Rigorous runtime analysis point out an exponential runtime gap between the population-based algorithm *Simple Evolutionary Multi-objective Optimizer* (SEMO) and several single individual-based algorithms on this problem. This means that among the algorithms considered, only the population-based MOEA is successful and all other algorithms fail.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; G.3 [**Probability and Statistics**]: Probabilistic Algorithms

General Terms

Algorithms, Theory

Keywords

Evolutionary algorithms, multi-objective optimization, runtime analysis

[*]Supported by the German Research Foundation (DFG) as a part of the collaborative research center "Computational Intelligence" (SFB 531).

[†]This work was conducted while visiting Universität Dortmund.

1. INTRODUCTION

The understanding of the role of populations in single-objective EAs has been supported by theoretical results [12, 13]. For example, there are problems where reducing the parent population size by one leads to an exponential increase in the runtime [11].

In a population of a single-objective EA, all individuals are comparable. This is typically not the case for MOEAs. Hence, it is not obvious that results for single-objective problems carry over to MOEAs when applied to a truly multi-objective problem. The use of populations in multi-objective EAs is often motivated by the need to find a set of solutions (the Pareto set) rather than a single optimal solution. However, it is unclear whether one can achieve the same goal by restarting a single individual algorithm. Laumanns *et al.* [9] prove that some simple population based MOEAs can slightly outperform a single individual based approach called ε-constrained method. This result alone is not sufficient to understand the role of populations in MOEAs since the runtime gap is rather small, and it gives only little insight into why the population based approach can be superior. In this paper, we present a problem where many single individual-based algorithms, including the ε-constrained method, fail dramatically. Furthermore, the presented problem has a structure that can better explain why the individual based algorithms fail.

This paper is organized as follows. We introduce the MOEAs considered and the necessary notation in the next section. The objective function will be presented in Section 3. In Sections 4–6, we show that all considered single individual algorithms fail on this objective function. Finally, in Section 7, we prove that the SEMO efficiently discovers all Pareto optimal points in the objective space.

2. PRELIMINARIES

2.1 Notation

We assume that the reader is familiar with the concepts of multi-objective optimization (see, e. g., [3]). We consider a binary maximization problem $f\colon \{0,1\}^n \rightarrow \mathbb{R}^m$. For clarity, we say that a vector b from the objective space $\mathbb{R}^m$ *weakly dominates* another vector a, denoted $a \preceq b$, if $a_i \leq b_i$, for all i. We say b *dominates* a, denoted $a \prec b$, if $a \preceq b$ and $a_i < b_i$ for at least one index i. This notation will also be used for solutions x and y in the search space $\{0,1\}^n$. For example, $x \preceq y$ if and only if $f(x) \preceq f(y)$. If $x \preceq y$ or $y \preceq x$ then x and y are *comparable*. The analysis uses

	local	global
weakest	$\text{RLS}_{\text{weakest}}$	(1+1) $\text{EA}_{\text{weakest}}$
weak	RLS_{weak}	(1+1) EA_{weak}
strong	$\text{RLS}_{\text{strong}}$	(1+1) $\text{EA}_{\text{strong}}$
ε-constr.	$\text{RLS}_{\varepsilon\text{-constr.}}$	(1+1) $\text{EA}_{\varepsilon\text{-constr.}}$

Table 1: The single individual algorithms considered according to selection and mutation operator.

standard notation (e. g., O, Ω and Θ) for asymptotic growth of functions (see, e. g., [2]).

2.2 The Multi-objective EAs

All the single individual multi-objective evolutionary algorithms considered in this paper are instantiations of the following scheme.

```
Choose $x$ uniformly from $\{0,1\}^n$.
Repeat
    Apply mutation to $x$ to obtain $x'$.
    If selection favors $x'$ over $x$
        then $x := x'$.
```

The algorithms differ in the choice of the mutation operator and in the choice of the selection operator. Two different mutation operators are considered. The *local mutation operator* flips a randomly chosen bit of x. The *global mutation operator* flips each bit of x independently with probability $1/n$. In the single-objective case, the objective function f establishes a total order in the search space and selection favors x' over x if $f(x') \geq f(x)$. Then, we obtain randomized local search (RLS) and the (1+1) EA with the local and global search operator, respectively.

For multi-objective problems, there are several options when to favor the offspring x' over the parent x. In this work, we consider four different selection operators. The *weakest selection operator* favors x' over x if x' weakly dominates x, or x' and x are incomparable. The *weak selection operator* favors x' over x if x' weakly dominates x. The *strong selection operator* favors x' over x if x' dominates x. Finally, we define the *ε-constraint selection operator* for two criteria problems as follows. (See [9] for the general definition with m criteria.) If $f_1(x) < \varepsilon$, then the operator favors x' over x if $f_1(x') \geq f_1(x)$. If $f_1(x) \geq \varepsilon$, then the operator favors x' over x if $f_1(x') \geq \varepsilon$ and $f_2(x') \geq f_2(x)$. Informally, the idea of the ε-constraint selection is to turn the first objective into a constraint, such that only solutions x with $f_1(x) \geq \varepsilon$ are feasible. So the primary goal is to minimize the constraint violation and then maximize the function f_2.

By the different choices of the mutation and the selection operators, we obtain eight different single individual MOEAs as summarized in Table 1. To obtain different Pareto optimal solutions, a single objective algorithm is run repeatedly, either sequentially (re-starts) or in parallel (multi-starts), and the hope is that not all runs will end up with the same solution. In case of the ε-constraint selection operator, it is necessary to vary the parameter ε between runs.

We compare the single individual algorithms with the population based algorithm SEMO which was introduced in [9]. The idea of SEMO is to keep a population of incomparable individuals. In each step, an offspring is produced and added to the population if it is not dominated by some individual in the population. Afterwards it may be necessary to remove individuals from the population that are weakly dominated by the new individual.

```
$P := \{x\}$, where $x$ is uniformly chosen from $\{0,1\}^n$.
Repeat
    Choose $x$ uniformly from $P$.
    Apply mutation to $x$ to obtain $x'$.
    If $x'$ is not dominated by any individual in $P$
        then add $x'$ to $P$, and remove all individuals
        weakly dominated by $x'$ from $P$.
```

In this paper, *local* SEMO refers to SEMO with the local mutation operator and analogously for *global* SEMO. The local SEMO and the global SEMO can be considered as multi-objective counterparts to (single-objective) RLS and the (1+1) EA, respectively. Both variants of SEMO have been the subject of theoretical runtime analysis ([6, 9, 10]).

3. THE OBJECTIVE FUNCTION

The idea is to devise an objective function that partitions the search space into k paths, each leading to some Pareto optimal solution. The search points on a path form plateaus such that it is difficult to proceed along a path, i. e., to reach the next plateau dominating the previous plateau. But there is always a short distance to another path. The idea is that the single-objective algorithms will spend most of the time jumping between paths, instead of improving on a single path, or they will spend much time to overcome a plateau. The hope is that SEMO will quickly produce a population of k individuals, one individual for each path, and these individuals will advance in parallel to their respective optima.

Let $n = k \cdot m$ and x a bit string of length n. We say that bit string x is divided into k *blocks*, where each block has length $m \geq 2$.

Definition 1. (Block Value, Active Block) Given a search point $x \in \{0,1\}^{k \cdot m}$ and an integer i, $0 \leq i \leq k-1$. Then the ith *block value* of x, denoted $|x|_i$, is defined as

$$|x|_i := \sum_{j=m \cdot i}^{m \cdot (i+1)-1} x_j.$$

The *active block* of a search point x is the left-most block with lowest block value. The number

$$j := \min \operatorname*{argmin}_{0 \leq i \leq k-1} \{|x|_i\}$$

denotes the *active block index*.

Table 2 gives examples of block values and active blocks, where $k = 4$ and $m = 3$. The following multi-objective function is essentially OneMax ([4]) defined on the active block, weighted differently with respect to each objective as to create different Pareto optimal solutions.

Definition 2. (Objective Function) For all search points x the objective function $f\colon \{0,1\}^n \to \mathbb{N} \times \mathbb{N}$ is defined by

$$f(x) := \big(f_1(x), f_2(x)\big),$$

where

$$f_1(x) := 2^{j \cdot m} \cdot (|x|_j + 1) \quad \text{and} \quad f_2(x) := 2^{(k-j-1) \cdot m} \cdot (|x|_j + 1)$$

and j is the active block index of x. The aim is to maximize f.

x	$\lvert x\rvert_0$	$\lvert x\rvert_1$	$\lvert x\rvert_2$	$\lvert x\rvert_3$	j	$f(x)$
$\underline{000}$ 000 000 000	0	0	0	0	0	(1, 512)
111 011 $\underline{010}$ 001	3	2	1	1	2	(128, 16)
$\underline{010}$ 111 011 001	1	3	2	1	0	(2, 1024)
$\underline{111}$ 111 111 111	3	3	3	3	0	(4, 2048)

Table 2: Examples of block values, active blocks (underlined), and objective function values.

The following proposition states that search points with the same active block index are comparable whereas search points with distinct block indices are incomparable.

Proposition 1. *Let $x, y \in \{0,1\}^n$ be two search points with active block index i and j, respectively.*

1. *If and only if $i = j$, x and y are comparable.*
2. *Moreover, if $i = j$, $x \preceq y$ is equivalent to $|x|_i \leq |y|_j$.*

Proof. Assume $i = j$. Then the objective function values of x and y depend only on $|x|_i$ and $|y|_j$, respectively. Since f_1 and f_2 are strictly increasing functions, x weakly dominates y or vice versa. Moreover, $|x|_i \leq |y|_j$ is equivalent to $f_1(x) \leq f_1(y) \;\wedge\; f_2(x) \leq f_2(y)$ and the latter is equivalent to $x \preceq y$.

It remains to show that if x and y are comparable then $i = j$. W.l.o.g. assume $x \preceq y$. Then, by f_1, the assumption implies $2^{i\cdot m}(|x|_i + 1) \leq 2^{j\cdot m}(|y|_j + 1)$. This is equivalent to $|x|_i \leq 2^{(j-i)\cdot m}(|y|_j + 1) - 1$. For $i > j$, the right-hand side of the last inequality is at most $2^{-m}(m+1) - 1 < 0$ as $m \geq 2$. Hence, $i > j$ is impossible. For $i < j$, considering f_2 leads to the same contradiction. $\square$

We can now describe the Pareto front and the Pareto set.

Proposition 2. *Let $n = m \cdot k$. For all integers i, $1 \leq i \leq k-1$, define the sets*

$$\begin{aligned} X_0^* &:= \{1^n\}, \quad \text{and} \\ X_i^* &:= \{x \in \{0,1\}^n \mid |x|_i = m-1, \\ &\qquad \forall j, 0 \leq j < i, \quad |x|_j = m, \text{ and} \\ &\qquad \forall j, i < j < k, \quad |x|_j \geq m-1\}. \end{aligned}$$

Furthermore, for all integers i, $1 \leq i \leq k-1$, define the points

$$\begin{aligned} F_0^* &:= \left(m+1, 2^{(k-1)\cdot m} \cdot (m+1)\right), \quad \text{and} \\ F_i^* &:= \left(2^{i\cdot m} \cdot m, 2^{(k-i-1)\cdot m} \cdot m\right). \end{aligned}$$

Then the Pareto set X^* and the Pareto front F^* of the bi-objective function defined in Definition 2 with parameters m and k are given by

$$X^* = \bigcup_{i=0}^{k-1} X_i^* \quad \text{and} \quad F^* = \bigcup_{i=0}^{k-1} \{F_i^*\}.$$

Furthermore, the preimage $f^{-1}(F_i^*)$ is X_i^*.

The proof has been omitted due to space limitations. It follows from Proposition 2 that the Pareto front has cardinality k.

A popular method to solve a multi-objective problem $g = (g_1, \ldots, g_m)$ is to solve the single-objective problem $g' := \sum_i w_i \cdot g_i$ instead, where the scalar objective function g' is a weighted sum of the original vector valued function g. The hope is to find different Pareto optima for different parameter settings $w_i > 0$. However, it is well-known that such linear aggregation functions fail for the non-convex parts of the Pareto front of g. Pareto optimal vectors which are not located on the convex hull of the solutions in the objective space cannot be detected for any setting of the weights. For our bi-objective function f presented in Definition 2, maximizing $w_1 f_1 + w_2 f_2$ is equivalent to maximizing $w f_1 + (1-w) f_2$ where $w \in [0,1]$. As the entire Pareto front of f is non-convex, any choice of w allows only to detect the solutions that maximize either f_1 or f_2. Hence, all methods that require a convex Pareto front are not applicable to f.

4. WEAK AND STRONG SELECTION

We show that there is a large fraction of the Pareto front such that, with an overwhelming probability, the algorithms RLS$_{\text{weak}}$, (1+1) EA$_{\text{weak}}$, RLS$_{\text{strong}}$, and (1+1) EA$_{\text{strong}}$ have to be started $e^{\Omega(n)}$ times before finding any Pareto optimal point from this fraction. The next proposition follows directly from Proposition 1.

Proposition 3. *All search points selected by either the weak selection operator or the strong selection operator have the same active block index as the initial search point.*

The idea behind the following theorem is that when the block length m is a constant, then the active block index of the initial search point will be low with high probability.

Theorem 1. *Let $n = m \cdot k$, where m is a constant, and A is any of the algorithms RLS$_{weak}$, (1+1) EA$_{weak}$, RLS$_{strong}$, and (1+1) EA$_{strong}$. Then, for all constants α, $0 < \alpha < 1$, there is a subset F_α^* of the Pareto front F^* with cardinality $|F_\alpha^*| \geq \alpha|F^*| - 1$ such that the probability that algorithm A needs less than $e^{c\cdot n}$ runs to find any Pareto optimal point in F_α^* is bounded by $e^{-\Omega(n)}$, $c > 0$ a sufficiently small constant.*

Proof. Define $F_\alpha^* := \cup_{i \geq (1-\alpha)k}^{k-1} \{F_i^*\}$, where F_i^* is as in Proposition 2. The Pareto front contains k points, and F_α^* contains αk elements, so the cardinality of F_α^* is greater than $\alpha|F^*| - 1$. All search points in the pre-image of F_α^* have active block indices at least $(1-\alpha)k$. By Proposition 3, in order to find a search point with an active block index that high, the initial search point must have active block index at least $(1-\alpha)k$. To get such a high active block index, it is necessary that all of the first $\lfloor (1-\alpha)k \rfloor$ blocks have block value unequal to 0. An upper bound on the probability that the active block index is higher than $(1-\alpha)k$ is therefore $(1 - (1/2)^m)^{(1-\alpha)\cdot k} = e^{-\Omega(n)}$ as m is a constant and, therefore, $k = \Theta(n)$. Furthermore, the probability that the event occurs within $e^{c\cdot n}$ runs is no more than $e^{c\cdot n} \cdot e^{-\Omega(n)} = e^{-\Omega(n)}$, for $c > 0$ a sufficiently small constant. $\square$

Both weak and strong selection turn out to be inadequate. The active block index of the initial search point will almost always be low, but these selection operators do not allow changing the active block index in a run. The weakest selection operator alleviates this problem by allowing to change the active block index; however, we show in the next section that this is not sufficient.

5. WEAKEST SELECTION

We prove that $\text{RLS}_{\text{weakest}}$ and the (1+1) $\text{EA}_{\text{weakest}}$ need with overwhelming probability an exponential time to find any Pareto optimal solution. The idea is the following. All Pareto optimal solutions have at most k 0-bits. We show that if the number of 0-bits is close to k, it is unlikely to lose another 0-bit in the next accepted step and it is much more likely to gain new 0-bits. Thus, there is a strong tendency to increase the number of 0-bits. To show that the probability of increasing the number 0-bits is high, we first show that there are many 1-bits that can be turned into zeroes.

Proposition 4. *Let x be any search point with $\ell \geq m$ 1-bits. There are $\ell - m$ 1-bits in x such that flipping any of these 1-bits produces a search point x' which is not dominated by x.*

Proof. There are at most m 1-bits in the active block of x. Assume that any of the remaining $\ell - m$ 1-bits flip. If the active block index changes, x and x' are incomparable (Proposition 1). Otherwise, the active block value remains unchanged implying that x' cannot be dominated by x. □

5.1 RLS

The analysis of how the number of 0-bits evolves over time will be based on a simple Markov process also known as the *gambler's ruin problem*. A gambler owns an initial capital of a dollars and plays against an adversary whose initial capital is b dollars. The gambler wins and loses a round with probability p and $1-p$, respectively. If he loses a round, he pays a dollar to the adversary and otherwise receives a dollar from the adversary. They continue the game until one player is ruined and the winner is the player who then owns the combined capital $a + b$. For a proof of the next theorem see, e.g., [5] or [1].

Theorem 2. *For $p \neq 1/2$, the probability that the gambler wins is $(1 - t^a)/(1 - t^{a+b})$, where $t := (1 - p)/p$. Consequently, for $p > 1/2$, this probability is at least $1 - t^a$.*

Theorem 3. *For $k \geq 4$ and $m \geq 5$, the expected time for $\text{RLS}_{\text{weakest}}$ to find any Pareto optimal solution is $e^{\Omega(n)}$. Moreover, there are positive constants c and c' such that the probability that $\text{RLS}_{\text{weakest}}$ finds any Pareto optimal solution in $e^{c' \cdot n}$ runs, each of $e^{c \cdot n}$ steps, is $e^{-\Omega(n)}$.*

Proof. By Chernoff bounds, the initial search point has less than $n/4$ 0-bits with an exponentially small probability of $e^{-\Omega(n)}$. We only consider the case where the first search point has at least $n/4$ 0-bits and wait for the first point in time where the number of 0-bits is at most $n/4$. In the following, we consider only situations with at most $n/4$ 0-bits. Then the number of 1-bits is at least $3n/4 > m$ and we can apply Proposition 4. Each mutation step of RLS either increases or decreases the number of 0-bits by 1, but not all steps are accepted. The probability that the next step is accepted and the number of 0-bits increases is at least $(3n/4 - m)/n = 3/4 - 1/k \geq 2/4$. The probability that the next step is accepted and the number of 0-bits decreases is at most $(n/4)/n \leq 1/4$. Hence, accepted steps increase and decrease the number of 0-bits with a probability of at least $p := 2/3$ and at most $1 - p = 1/3$, respectively. We consider the number of 0-bits as the capital of the gambler in Theorem 2. Initially, it is $\lfloor n/4 \rfloor$ and the capital of his opponent is 1. Then the probability that the number of 0-bits increases to $\lfloor n/4 \rfloor + 1$ before it decreases to 0 is at least $1 - t^{\lfloor n/4 \rfloor + 1}$ where $t := 1/2$. We are interested in the probability to produce a Pareto optimal point. A Pareto optimal point has less than k 0-bits. Hence, we consider the gambler ruined as soon as his capital reaches k dollars. This is equivalent to reducing his initial capital by k. Since $m \geq 5$, we have $k \leq n/5$ and obtain an upper bound of $t^{\lfloor n/4 \rfloor - k + 1} = e^{-\Omega(n)}$ for the probability to reach a Pareto optimal point before a point with at least $n/4$ 0-bits again. Hence, we can apply this argument repeatedly such that for a sufficiently small constant c, $e^{c \cdot n}$ repetitions of the game are successful with a probability of only $e^{-\Omega(n)}$. Taking into account the probability that the initial step is not as desired leads to the result that a run of $e^{c \cdot n}$ steps is successful with a probability of only $e^{-\Omega(n)}$ and leads to the claimed expected runtime.

We now consider independent runs of $\text{RLS}_{\text{weakest}}$, i.e., sequential runs (restarts) or parallel runs (multi-starts). If each of $e^{c' \cdot n}$ runs includes up to $e^{c \cdot n}$ steps, the probability that any of these runs is successful is at most $e^{c' \cdot n} \cdot e^{-\Omega(n)} = e^{-\Omega(n)}$ if $c' > 0$ is sufficiently small. Hence, independent runs of $\text{RLS}_{\text{weakest}}$ do not help to increase the success probability substantially. □

5.2 (1+1) EA

The global mutation operator of the (1+1) $\text{EA}_{\text{weakest}}$ may flip many bits in one step and increase or decrease the number of 0-bits by large values. Although the probability of a large change in a single step is rather low, such a step is not unlikely to happen in a run including exponentially many steps. Therefore, we have to take large changes into account. The following drift theorem provides a general technique for proving exponential lower bounds on the first hitting-time in Markov processes. It serves as a counterpart to Theorem 2. We apply a result due to [8] that goes back to [7]. Analyzing the proof in [8], it follows immediately that it includes a stronger result than stated, namely a result on the success probability to reach a state with certain properties and not only the expected waiting time. We state this result in Theorem 4.

Theorem 4. *(Drift Theorem) Let $X_0, X_1, X_2, \ldots$ be the random variables describing a Markov process over the state space S and $g\colon S \to \mathbb{R}_0^+$ a function that assigns to each state a non-negative real number. Pick two real numbers $a(n)$ and $b(n)$ which depend on a parameter n such that $0 \leq a(n) < b(n)$ holds. Let the random variable T denote the earliest point in time $t \geq 0$ that satisfies $g(X_t) \leq a(n)$.*

If there are constants $\lambda > 0$ and $D \geq 1$ and a polynomial $p(n) > 0$ such that the four conditions

$$\begin{aligned} g(X_0) &\geq b(n), \\ b(n) - a(n) &= \Omega(n), \\ \forall t \geq 0 : E\big(e^{-\lambda(g(X_{t+1}) - g(X_t))} \mid X_t,\ a(n) < g(X_t) < b(n)\big) &\\ \leq 1 - 1/p(n) &\\ \forall t \geq 0 : E\big(e^{-\lambda(g(X_{t+1}) - b(n))} \mid X_t,\ b(n) \leq g(X_t)\big) \leq D& \end{aligned}$$

hold then for all time bounds $B \geq 0$

$$\text{Prob}(T \leq B) \leq e^{\lambda(a(n) - b(n))} \cdot B \cdot D \cdot p(n).$$

Since $\lambda(a(n) - b(n)) = -\Omega(n)$ and $p(n)$ is a polynomial, the last bound is exponentially small for $B := e^{c\cdot n}$ if $c > 0$ is sufficiently small.

THEOREM 5. *For $4 \le k \le \big((1/10) - \varepsilon\big)n$, $\varepsilon < 1/10$ a positive constant, the expected time for the (1+1) EA$_{weakest}$ to find any Pareto optimal solution is $e^{\Omega(n)}$. Moreover, there are positive constants c and c' such that the probability that the (1+1) EA$_{weakest}$ finds any Pareto optimal solution in $e^{c'\cdot n}$ runs, each of $e^{c\cdot n}$ steps, is $e^{-\Omega(n)}$.*

PROOF. Let the random variable $X_t \in \{0,1\}^n$ denote the search point of the (1+1) EA$_{\text{weakest}}$ at time $t \ge 0$ when applied to f. To apply the above drift theorem, $g(X_t)$ equals the number of 0-bits of X_t. We choose $b(n) := n/10$ and $a(n) := k = n/m$. By Chernoff bounds, the initial search point X_0 has more than $b(n)$ 0-bits with an overwhelming probability of $1 - e^{-\Omega(n)}$. Thus, we only consider the case where $b(n) < g(X_0)$ such that the first condition is satisfied. As $k \le \big((1/10) - \varepsilon\big)n$, also the second condition is met.

To check the third condition we have to bound

$$E\big(e^{-\lambda(g(X_{t+1})-g(X_t))} \mid X_t,\ k < g(X_t) < n/10\big)$$

from above. Let $p_j(X_t)$ denote the probability that the g-value increases by j in the next step when the current search point is X_t and $k < g(X_t) < n/10$. Then the above expectation is bounded from above by

$$\sum_{j:=-g(X_t)}^{n-g(X_t)} e^{-\lambda\cdot j}\cdot p_j(X_t). \qquad (*)$$

For $j > 0$, we only increase the value of the sum $(*)$ if we replace $p_j(X_t)$ with some lower bound p_j independent of X_t and increase $p_0(X_t)$ by $\Delta_j := p_j(X_t) - p_j$. For all $j \ge 2$, we choose the trivial lower bounds $p_j := 0$. The probability $p_1(X_t)$ is lower bounded by the probability of the event that exactly one 1-bit of at least $n - b(n) - m \ge 9n/10 - m$ 1-bits flip (Proposition 4). Hence, for $k \ge 4$,

$$\frac{9n/10 - m}{n}\big(1 - 1/n\big)^{n-1} \ \ge\ \frac{9}{10e} - \frac{1}{ke} \ \ge\ \frac{2}{10}$$

and we can choose $p_1 := 2/10$.

For $j < 0$, the value of the sum $(*)$ only increases if we replace $p_j(X_t)$ by some upper bound p_j and decrease $p_0(X_t)$ by $\Delta_j := p_j - p_j(X_t)$. The probability of decreasing the g-value by j in the next step is upper bounded by the probability of the event that at least j 0-bits are turned into 1-bits. Hence,

$$\binom{g(X_t)}{j}\cdot\frac{1}{n^j} \ \le\ \frac{(n/10)^j}{j!}\cdot\frac{1}{n^j} \ =\ \frac{1}{10^j j!}$$

and $p_j := 1/(10^j j!)$ is a correct upper bound. We now consider the Markov process where all probabilities $p_j(X_t)$ are replaced with our corresponding bounds. If we pessimistically assume that the g-value, the number of 0-bits can decrease by any $j > 0$ (i.e., also for $j > g(X_t)$) we only overestimate the probability to decrease the g-value. We obtain a process where the transition probabilities are independent of the $g(X)$-value. The new process with

$$p_j := 0, \quad \text{and} \quad p_{-j} := 1/(10^j j!) \quad \text{for all } j \ge 2,$$
$$p_{-1} := 1/10, \quad p_1 := 2/10, \quad \text{and} \quad p_0 := 1 - \sum_{j\ne 0} p_j,$$

reaches a g-value of at most $a(n)$ only "faster" than the original process describing the (1+1) EA$_{\text{weakest}}$ applied to f.

It now suffices to bound the sum

$$\begin{aligned}&e^{-1\cdot\lambda}p_1 + e^{-0\cdot\lambda}p_0 + e^{1\cdot\lambda}p_{-1} + \sum_{j\ge 2} e^{j\cdot\lambda}p_{-j}\\ &= \Big(1 - p_1 - p_{-1} - \sum_{j\ge 2} p_{-j}\Big) + e^{-\lambda}p_1 + e^{\lambda}p_{-1} + \sum_{j\ge 2} e^{j\cdot\lambda}p_{-j}\\ &= 1 - \big((1 - e^{-\lambda})p_1 + (1 - e^{\lambda})p_{-1}\big) + \sum_{j\ge 2}\big(e^{j\cdot\lambda} - 1\big)p_{-j}\end{aligned} \qquad (**)$$

for an appropriate choice of $\lambda > 0$. We choose $\lambda := (1/2)\ln 2$ and show that the sum $(**)$ is at most $1 - \alpha + \beta$ for positive constants α and β, where $\alpha > \beta$. For our choice of λ, we obtain

$$(1 - e^{-\lambda})p_1 + (1 - e^{\lambda})p_{-1} \ =:\ \alpha \ >\ \frac{17}{1000}$$

and

$$\begin{aligned}\sum_{j\ge 2}\big(e^{j\cdot\lambda} - 1\big)p_{-j} \ &\le\ \sum_{j\ge 2}\frac{2^{j/2}}{10^j j!} \ =\ \sum_{j\ge 2}\frac{(\sqrt{2}/10)^j}{j!}\\ &=\ \exp(\sqrt{2}/10) - 1 - \sqrt{2}/10 \ =:\ \beta \ <\ \frac{11}{1000}.\end{aligned}$$

Hence, the sum $(**)$ is at most $1 - \delta$ for a positive constant δ and we can choose a constant polynomial $p(n) := 1/\delta$.

It remains to check the last condition of Theorem 4. We bound

$$\begin{aligned}&E\big(e^{-\lambda(g(X_{t+1})-n/10))} \mid X_t, n/10 \le g(X_t)\big)\\ &\le\ E\big(e^{-\lambda(g(X_{t+1})-g(X_t))} \mid X_t, n/10 \le g(X_t)\big)\end{aligned}$$

from above and proceed analogously to the case of the third condition. We bound $p_j(X_t)$ by the trivial lower bound $p_j := 0$, for all positive j. For j negative, the probability $p_j(X_t)$ is upper bounded by the probability of the event that at least j 1-bits flip. The corresponding probability is at most

$$\binom{n}{j}\cdot\frac{1}{n^j} \ \le\ \frac{1}{j!} \ \le\ \Big(\frac{e}{j}\Big)^j.$$

We consider the process where

$$p_j \ :=\ 0, \quad \text{and} \quad p_{-j} \ :=\ \Big(\frac{e}{j}\Big)^j, \quad \text{for all } j \ge 1, \text{ and}$$
$$p_0 \ :=\ 1 - \sum_{j\ge 1} p_j.$$

Now it suffices to estimate the sum

$$\begin{aligned}e^{-0\cdot\lambda}p_0 + \sum_{j\ge 1} e^{j\cdot\lambda}p_{-j} \ &\le\ 1 + \sum_{j\ge 1} 2^{j/2}\Big(\frac{e}{j}\Big)^j\\ &\le\ 1 + \sum_{j\ge 1}\Big(\frac{\sqrt{2}e}{j}\Big)^j \ = O(1) + \sum_{j\ge 12}\Big(\frac{1}{2}\Big)^j \ =\ O(1).\end{aligned}$$

Hence the last sum is bounded by some positive constant D.

By Theorem 4, the probability that a state with less than k 0-bits is reached in $B := e^{c\cdot n}$ steps is $e^{-\Omega(n)}$ if c is sufficiently small. Taking into account the probability that the initial search point has at least $n/10$ 0-bits leads to the success probability of $e^{-\Omega(n)}$ in a run of up to $e^{c\cdot n}$ steps. This result implies the claimed expected runtime and, by the same arguments as presented at the end of the proof of Theorem 3, a success probability of $e^{-\Omega(n)}$ for $e^{c' \cdot n}$ independent runs. □

6. ε-CONSTRAINT SELECTION

Proposition 5. *Let $n = m \cdot k$ and let A be any of the algorithms RLS$_{\varepsilon\text{-constr.}}$ and (1+1) EA$_{\varepsilon\text{-constr.}}$. For a constant β, $0 < \beta < 1$, define the stochastic process $X_1, \ldots, X_s$, where $X_i \in \{0,1\}^{(1-\beta)mk}$ is the $(1-\beta)mk$-bit long suffix of the search point in step i of algorithm A working on the objective function defined in Definition 2. Then, given that the search points in the first s steps of A have active block indices less than βk and active block values 0, each of the vectors X_i is uniformly distributed over $\{0,1\}^{(1-\beta)mk}$.*

Proof. The proof is by induction over the number of iterations i. The initial search point is sampled uniformly at random, so the statement trivially holds for the base case $i = 1$. Assume that the vectors $X_1, \ldots, X_i$ are uniformly distributed over $\{0,1\}^{(1-\beta)mk}$. Because the active block index of X_i is less than βk and the active block value is 0, acceptance of a new search point does not depend on the $(1-\beta)mk$-bit long suffix of the search point. The global and the local mutation operator applied to a uniformly distributed suffix, produces a uniformly distributed suffix. (See e.g., [4]). □

Theorem 6. *Let $n = m \cdot k$, where $m \geq 2$ is a constant and α, $0 < \alpha < 1$, is an arbitrary constant. Then there exist constants $c, c' > 0$, and a subset F^*_α of the Pareto front F^* with cardinality $|F^*_\alpha| \geq \alpha \cdot |F^*|$ such that the probability that $e^{c' \cdot n}$ runs, each of $e^{c \cdot n}$ steps, of RLS$_{\varepsilon\text{-constr.}}$ or (1+1) EA$_{\varepsilon\text{-constr.}}$ find any Pareto optimal point of F^*_α is $e^{-\Omega(n)}$. The parameter setting ε is allowed to change between runs.*

Proof. Define $F^*_\alpha := \cup_{i=0}^{\alpha \cdot k-1}\{F^*_i\}$, where F^*_i is as in Proposition 2. We call a run *bad* when the initial search point has active block index higher than αk, or has active block value higher than 0. The probability of the first case is upper bounded by the probability of the event that all the first αk blocks have block values different from 0, and the second case is upper bounded by the probability of the event that all blocks have block values different from 0. Because the second event implies the first event, the probability of a bad run is no more than $(1-(1/2)^m)^{\alpha k} = e^{-\Omega(n)}$.

Assume now that the initial search point has active block index less than αk and active block value 0. No Pareto optimal search point has active block value 0 when $m \geq 2$. We lower bound the optimization time by analyzing the time until the search point for the first time has active block value at least 1. We say that the algorithm is in the *constraint-minimization* state when the search point x has function value $f_1(x) < \varepsilon$, and in the *maximization* state when the search point x has function value $f_1(x) \geq \varepsilon$. In the maximization state, a search point x will be replaced by a search point x' if and only if $f_2(x') \geq f_2(x)$ and $f_1(x') \geq \varepsilon$. Consequently, the algorithm will never leave the maximization state once entered, and the active block index can only decrease in this state. (See Definition 2.) The maximal active block index during a run will, therefore, never be higher than the active block index in the first step after the algorithm has entered the maximization state. We will show that with overwhelming probability, the highest active block index will never be higher than $(k/2)(1+\alpha)$.

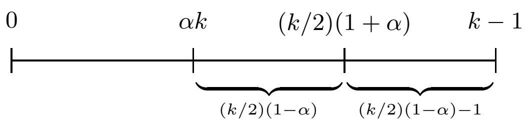

Figure 1: Active block index.

We divide the search point into three parts as shown in Figure 1. We first divide the string into an αk blocks long prefix and a $(1-\alpha)k$ blocks long suffix. Then, we divide the suffix into two almost equally long parts, each approximately $(k/2)(1-\alpha)$ blocks long. The last part now begins at block index $\alpha k + (k-\alpha k)/2 = (k/2)(1+\alpha)$.

Assume first that $\varepsilon < 2^{\alpha mk}$. As long as the algorithm is in the constraint minimization state, the active block index is less than αk. (See Definition 2.) Therefore, by Proposition 5, the $(1-\alpha)k$ blocks long suffix will be uniformly distributed. Furthermore, in the step when the algorithm enters the maximization state, the blocks in the interval from αk to $(k/2)(1+\alpha)$ will also be uniformly distributed. Hence, the probability that the first search point in the maximization state has active block index higher than $(k/2)(1+\alpha)$ is upper bounded by the probability that all the blocks in the interval from αk to $(k/2)(1+\alpha)$ have block values different from 0. By the uniform distribution, the probability of this event is no more than $(1-(1/2)^m)^{(k/2)(1-\alpha)} = e^{-\Omega(n)}$. Therefore, with overwhelming probability, the active block index during the entire run will be no more than $(k/2)(1+\alpha)$.

We now consider the second case where $\varepsilon \geq 2^{\alpha km}$. In this case, all elements in F^*_α violate the ε-constraint. If the active block index becomes higher than αk, none of the search points in F^*_α can be found. We optimistically assume that the active block index during the entire run will never be higher than αk when $\varepsilon \geq 2^{\alpha km}$.

Hence, for both cases, we can now assume that the active block index is less than $(k/2)(1+\alpha)$ during the entire run. So by Proposition 5, the suffix X_i corresponding to the last $(k/2)(1-\alpha)-1$ blocks of the search point will be uniformly distributed over the set $\{0,1\}^{m((k/2)(1-\alpha)-1)}$. We say that the vector X_i is *good* if all blocks in X_i have block values different from 0. Because X_i is uniformly distributed, Prob("X_i is good") $= (1-(1/2)^m)^{(k/2)(1-\alpha)-1} = e^{-\Omega(n)}$. To reach active block value 1 within s steps, at least one of the variables $X_1, \ldots, X_s$ must be good. By union bound, the probability of at least one good variable X_i during a run of length $s := e^{cn}$ is no more than Prob($\cup_{i=1}^{e^{cn}}$"X_i is good") $\leq e^{cn} \cdot e^{-\Omega(n)} = e^{-\Omega(n)}$ for a sufficiently small constant c.

Furthermore, $e^{c' \cdot n}$ runs, each of length $e^{c \cdot n}$, will be successful with probability $e^{-\Omega(n)}$ for a sufficiently small constant $c' > 0$. □

7. SEMO

We prove that within polynomial time, the SEMO population covers the entire Pareto front on the problem defined in Definition 2. The idea is the following. The problem consists of k independent paths, one path for each block, with Pareto optimal solutions at the end of each path. However, to progress to a higher level on a path, a large plateau must be overcome. We show that the individuals in the SEMO

P	Individual 0	Individual 1	Individual 2	t	v
1	$\underline{00}\ 00\ 00$	$01\ \underline{00}\ 00$	$\boxed{01\ 01\ \underline{00}}$	2	0
2	$\boxed{\underline{01}\ 01\ 01}$	$01\ \underline{00}\ 00$	$01\ 01\ \underline{00}$	0	1
3	$\underline{01}\ 01\ 01$	$\boxed{11\ \underline{01}\ 01}$	$01\ 01\ \underline{00}$	1	1
4	$\underline{01}\ 01\ 01$	$11\ \underline{01}\ 01$	$\boxed{11\ 11\ \underline{01}}$	2	1
5	$\boxed{\underline{11}\ 11\ 11}$	$11\ \underline{01}\ 01$	$11\ 11\ \underline{01}$	0	2

Table 3: Examples of the active path concept.

population will be distributed over these paths, with at most one individual per path. SEMO will, thereby, optimize the paths in parallel, such that no gain along any path is lost. The individuals in the population of SEMO are pairwise incomparable. Hence, the following proposition is a consequence of Proposition 1.

Proposition 6. *For each block index j, $0 \leq j \leq k-1$, a population in SEMO has at most one element with active block index j.*

We introduce a concept called the *active path* of the population to analyze the parallel improvements along each path. Informally, the *active path number* corresponds to the active block index on which SEMO has advanced the most, and the *active path value* designates how far on this path SEMO has advanced.

Definition 3. (Active Path) Let $x_1, x_2, \ldots, x_r$ be the individuals in a SEMO population, and $j_1, j_2, \ldots, j_r$ their respective active block indices. Then the *active path value v* of a population is the maximal active block value in the population, i. e.,

$$v := \max_{1 \leq i \leq r} \{|x_i|_{j_i}\},$$

and the *active path number t* of the population is the highest active block index among the individuals having active block value v, i. e.,

$$t := \max_{1 \leq i \leq r} \{j_i \mid |x_i|_{j_i} = v\}.$$

(Note that, by Proposition 6, there is only one individual in the population with active block index t.)

Table 3 gives five examples of active path number and active path value of a population. Each row describes a population, and each population has three individuals. The last two columns in the table give the active path number t and the active path value v of the corresponding population. Additionally, the active path representative in each population is framed.

When the active path value of a population is m, the population must contain the Pareto optimal solution 1^n. After the Pareto optimal solution 1^n has been found, SEMO will quickly discover the rest of the Pareto front. Our approach to analyze SEMO, therefore, focuses on the time it takes to increase the active path value to m.

Proposition 7. *The active path value never decreases. If the active path number decreases then the active path value increases.*

Proof. Suppose that the active path value decreases, and the old active path was represented by individual x. Then individual x cannot be member of the new population. Hence, the new population must contain a new element x' such that $x \preceq x'$. Proposition 1 implies that $|x|_j \leq |x'|_j$, which contradicts that the active path value decreases.

For the second claim, let the old active path number i be represented by the search point x. If the active path number decreases then the new active path must be represented by the search point x', having active block index j, $0 \leq j < i$. By Proposition 1, the search point x will remain in the new population. The only way the new search point x' can be the new active path representative is when x' has higher active block value than x. This means that the active path value must increase. □

Proposition 8. *The expected time to increase the active path value is bounded above by the expected time to change the active path number k times.*

Proof. Since there are k different blocks, the maximal number of times the active path number can increase without being decreased is $k-1$. Hence, by Proposition 7, after k active path number changes, the active path value must have increased at least once. □

Theorem 7. *The expected time until the SEMO population covers the Pareto front is $O(nk^2 \log m)$.*

Proof. Our analysis will be based on 1-bit-mutations only. Since the probability that a specified bit flips is at least $1/n$ and $1/(en)$ for the local and the global mutation operator, respectively, the waiting time for a specific 1-bit-mutation is only larger for the global SEMO. Consequently, it suffices to derive upper bounds on the runtime of the global SEMO.

We divide the optimization process into two consecutive phases. The first phase begins when the algorithm starts and ends when the population for the first time contains the Pareto optimal solution 1^n. Thereafter, the second phase starts and it lasts until the entire Pareto front is covered.

In the first phase, the active path value must be increased at most m times because an active path value of m implies that the population includes the individual 1^n. By Proposition 8, at most k active path number changes suffice to increase the active path value once. We call a step *successful* if

1. the active path number increases, or

2. the active path value increases.

We claim that a step is successful if it first chooses the individual x representing the active path (i. e., its active block index equals the active path number) and then flips one or more of the 0-bits in the active block to obtain x'. Two cases must be considered, either x and x' have the same active block index, or they do not.

In the case that x and x' have the same active block index, then x' clearly dominates x because x' has more 1-bits in its active block. Hence, x' replaces x in the new population and the active path value increases.

Now, assume that x and x' have different active block indices i and j, respectively. We first show that x' will be accepted. If the new search point x' is not accepted, then there must exist a search point y in the population which strictly dominates x'. Proposition 1 implies that y has active block index j, and that $|x'|_j < |y|_j$. Furthermore, because i is the active block index in x, and x' by assumption differs from x in block i only, we have $|x|_i \leq |x|_j = |x'|_j$. However, the last inequality implies that $|x|_i < |y|_j$, which contradicts that x was the active path representative in the old population. The search point x' will, therefore, be accepted.

If $i > j$ then $|x|_i < |x|_j = |x'|_j$ because i is the active block index of x, and x and x' do not differ in block j. The search point x' will be the new active path representative and the active path value increases. Analogously, $i < j$ implies $|x|_i \leq |x|_j = |x'|_j$. Hence, x' has at least as high active block value as x, and x' has higher active block index than x. The search point x' will be the new active path representative and the active path number will increase.

Now we estimate the probability of a successful step. By Proposition 6, the probability of choosing the individual representing the active path is at least $1/k$. Given that the active block value is v, the probability of flipping at least one of the $m - v$ 0-bits in the active block of x (and no other bits) is at least

$$\frac{m-v}{n}\left(1-\frac{1}{n}\right)^{n-(m-v)} \geq \frac{m-v}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{(m-v)}{en}.$$

The probability of a successful step, therefore, is at least $(m - v)/(ekn)$. Using Proposition 8, the expected duration of the first phase is bounded from above by

$$\sum_{v=0}^{m-1} k \cdot \frac{ekn}{(m-v)} = ek^2 n \sum_{v=1}^{m} \frac{1}{v} = O\big(k^2 n \log m\big).$$

In the second phase, the population contains the Pareto optimal solution 1^n, and it will never be removed from the population. Given that there are i remaining points in the Pareto front to be discovered, the probability of selecting $x = 1^n$ and mutating solely one 1-bit in one of the corresponding i blocks is at least

$$\frac{1}{k} \cdot \frac{im}{n} \cdot \left(1-\frac{1}{n}\right)^{n-1} \geq \frac{im}{ekn} = \frac{i}{ek^2}.$$

The expected time to find the at most $k - 1$ remaining points in the Pareto front is, therefore, no more than

$$\sum_{i=1}^{k-1} \frac{ek^2}{i} \leq ek^2 \sum_{i=1}^{k} \frac{1}{i} = O(k^2 \log k).$$

Hence, the expected time until SEMO covers the entire Pareto front is $O(k^2 n \log m)$. $\square$

8. CONCLUSION

This paper introduces a simple bi-objective function to contrast two types of multi-objective EAs: population-based and single individual-based algorithms. The problem features a large number of incomparable search points and large plateaus. The runtime of the population-based algorithm SEMO is compared with the runtime of nine single individual-based approaches (eight variants from Table 1 plus the linear aggregation approach in Section 3).

Among the algorithms studied, only the population-based algorithm SEMO finds the Pareto front in expected polynomial time. All single individual algorithms fail on this problem because they either too easily accept incomparable search points, or because they cannot overcome the large plateaus in the search space. SEMO is efficient on the problem because the individuals in the population collectively lead to better solutions, i. e., each individual follows a path leading to one Pareto optimal solution. The result demonstrates the importance of populations for certain types of multi-objective problems.

Our result improves an earlier result in [9] where it is shown that some simple population-based MOEAs slightly outperform the ε-constrained method. However, the result yields only a small polynomial runtime gap. Here, we provide an exponential gap, proving that even multi-start variants of a number of single individual-based approaches fail with overwhelming probability. In contrast, SEMO discovers all Pareto optimal solutions efficiently.

9. REFERENCES

[1] K. L. Chung. *Elementary Probability Theory with Stochastic Processes*. Springer, 1974.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw Hill, New York, NY, 2nd edition, 2001.

[3] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

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

[5] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.

[6] O. Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Proc. of the 2003 Congress on Evolutionary Computation (CEC '03)*, volume 3, pages 1918–1925. IEEE, 2003.

[7] B. Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14:502–525, 1982.

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

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

[10] F. Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. In *Proc. of the 8th Conference on Parallel Problem Solving from Nature (PPSN '04)*, volume 3242 of *LNCS*, pages 80–89. Springer, 2004.

[11] T. Storch. On the choice of the population size. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO '04)*, volume 3102 of *LNCS*, pages 748–760. Springer, 2004.

[12] C. Witt. Population size vs. runtime of a simple EA. In *Proc. of the 2003 Congress on Evolutionary Computation (CEC '03)*, volume 3, pages 1996–2003. IEEE Press, 2003.

[13] C. Witt. An analysis of the $(\mu + 1)$ EA on simple pseudo-boolean functions. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO '04)*, volume 3102 of *LNCS*, pages 761–773. Springer, 2004.