Running Time Analysis of Evolutionary Algorithms on a Simplified Multiobjective Knapsack Problem

Marco Laumanns, Lothar Thiele and Eckart Zitzler
Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology Zurich, ETH-Zentrum, CH-8092 Zürich, Switzerland
(E-mail: {laumanns,thiele,zitzler}@tik.ee.ethz.ch)

Abstract. In this paper, the expected running time of two multiobjective evolutionary algorithms, SEMO and FEMO, is analyzed for a simple instance of the multiobjective 0/1 knapsack problem. The considered problem instance has two profit values per item and cannot be solved by one-bit mutations. In the analysis, we make use of two general upper bound techniques, the decision space partition method and the graph search method. The paper demonstrates how these methods, which have previously only been applied to algorithms with one-bit mutations, are equally applicable for mutation operators where each bit is flipped independently with a certain probability.

Keywords: decision space partition, evolutionary algorithms, graph search, knapsack problem, multiobjective optimization, Pareto set, running time analysis

1. Introduction

Evolutionary Algorithms (EAs) are probabilistic search techniques inspired by models of natural evolution (Bäck et al., 1997). EAs have successfully been applied to optimization problems in various disciplines, including problems with multiple objectives (see Deb (2001) for an overview). Although EAs are relatively simple to describe and to implement compared with other, more specialized optimization methods, they represent stochastic processes which are difficult to analyze. Nevertheless, it is desirable to make precise statements about their performance, e.g., how long a certain algorithm takes to find the optimal solution for a specific problem or a problem class.

In the case of a single objective and discrete search spaces, several theoretical results have been achieved regarding the optimization of pseudo-Boolean functions, a recent overview is given by Beyer et al. (2002). In the multiobjective case, however, no running time results have been available until recently. Scharnow et al. (2002) analyzed a variant of the (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. In previous work (Laumanns et al., 2002), we presented the first running time analysis of population-based EAs (SEMO and FEMO) on a pseudo-Boolean problem with

two conflicting objectives (LOTZ — Leading Ones Trailing Zeroes). This study was subsequently extended in (Laumanns et al., 2004) by considering two problems (mLOTZ and mCOCZ) with an arbitrary number of objectives m, and by comparing the running time of different population-based multiobjective EAs with a multistart (1+1)-EA based on the epsilon-constraint method. This study introduced a general upper bound method based on decision space partition and a general graph search method. The results, however, were derived only under the assumption of one-bit mutations.

The use of one-bit mutations is attractive for theoretical investigations as its analysis is often easier compared to other mutation operators. It is, however, only a local search operator flipping one bit at a time and therefore does not guarantee a finite running time for all pseudo-Boolean functions. On the contrary, the independent-bit mutation operator, which flips each bit independently with a certain probability, can be called a global search operator since it can produce any point in the decision space with positive probability. Thus, the independent-bit mutation operator is more commonly used in practise, but its analysis is typically more difficult, in the single-objective as well as in the multiobjective case. Nevertheless, the first analysis of the simple evolutionary multiobjective optimizer SEMO using independent-bit mutations was given by Giel (2003), who proved a tight worst-case bound that holds for all functions $\{0,1\}^n \longrightarrow \mathbb{R}^m$ as well as running time bounds for the LOTZ and for another two-objective problem originally proposed by Schaffer (1985).

The aim of this paper is to show that the general analytical techniques developed in (Laumanns et al., 2004) are also applicable to algorithms with independent-bit mutations, and how this can be achieved. In particular, the graph search method needs to be applied with caution, and we present a new lemma based on spanning trees to cope with the difficulties of independent-bit mutations. To exemplify the approach, we analyze a simple instance of the multiobjective knapsack problem, which will be introduced in the next section. Section 2 also introduces the algorithms to be investigated. The subsequent analysis is split into two parts. The first part covers the time needed to explore non-optimal search points and applies the general upper bound technique based on decision space partition. The second part is concerned with the time needed to explore the Pareto-optimal set and makes use of the general graph search technique.

2. Problem Statement

The optimization problem considered in this study is a special instance of the multiobjective knapsack problem. Given is a set of n items, each of which has m profit and k weight values associated with it. The goal is to select a subset of items such that the sums over each of their k-th weight values do not exceed given bounds and the sums over each of their m-th profit values are maximized.

This type of knapsack problem has been used extensively as a benchmark problem for evolutionary multiobjective optimization (see, e.g., Zitzler and Thiele, 1999, or Jaszkiewicz, 2002). A representation as a pseudo-Boolean optimization problem is typically used, where the n binary decision variables denote whether an item is selected or not. For empirical studies, the parameters of the problem, the weight and profit values, are usually drawn at random from a given probability distribution.

Here, we restrict ourselves to the following simple instance with $m = 2$ objectives and $k = 1$ constraint. The weights are constant, and the profit values are exponentially decreasing in the first objective and exponentially increasing in the second objective.

Definition 1. (MOKP) Let $X = \{0,1\}^n$ be the decision space and $n \in \mathbb{N}$. The Multiobjective Knapsack Problem (MOKP) is defined as the problem to

$$\begin{aligned} \text{maximize} \quad & f(x) = (f_1(x), f_2(x)) \\ \text{subject to} \quad & g(x) \leq n/2, \\ \text{where} \quad & f_1(x) = \sum_{i=1}^{n} 2^{n-i} x_i, \\ & f_2(x) = \sum_{i=1}^{n} 2^{i-1} x_i, \\ & g(x) = \sum_{i=1}^{n} x_i, \\ & x = (x_1, \ldots, x_n) \in X. \end{aligned}$$

Our goal is to identify the set of Pareto-optimal decision vectors of MOKP according to the following definition.

Definition 2. (Pareto optimality) Let $f : X \to F$, where f is called *objective function*, X *decision space*, and $F \subseteq \mathbb{R}^m$ *objective space*. The elements of X are called *decision vectors* and the elements of F *objective vectors*. Let $X' \subseteq X$ denote the set of *feasible decision vectors*, i.e.,

decision vectors fulfilling the constraints. A feasible decision vector $x^* \in X'$ is *Pareto optimal* if there is no other $x \in X'$ that *dominates* x^*. x dominates x^*, denoted as $x \succ_f x^*$, 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*.

The MOKP function is a bijection, so none of the $|X| = \Theta(2^n)$ decision vectors maps to the same objective vector. The Pareto set contains $|X^*| = n^2/4 + 1$ elements and has the following form.

Proposition 1. A decision vector $x \in X$ is Pareto-optimal in the MOKP problem, if and only if it has the form

$$(\underbrace{1, 1, \ldots, 1}_{a}, \underbrace{0, 0, \ldots, 0}_{\frac{n}{2}}, \underbrace{1, 1, \ldots, 1}_{\frac{n}{2}-a}) \quad \text{(Type 1)}$$

or

$$(\underbrace{1, 1, \ldots, 1}_{b}, \underbrace{0, 0, \ldots, 0}_{c}, 1, \underbrace{0, 0, \ldots, 0}_{\frac{n}{2}-c}, \underbrace{1, 1, \ldots, 1}_{\frac{n}{2}-b-1}), \quad \text{(Type 2)}$$

where $a \in \{0, 1, \ldots, \frac{n}{2}\}$, $b \in \{0, 1, \ldots, \frac{n}{2} - 1\}$, and $c \in \{1, \ldots, \frac{n}{2} - 1\}$.

Proof. First let us note that only vectors with $n/2$ variables set to one can be Pareto-optimal. A solution with less than $n/2$ ones can be improved in both objectives by exchanging any arbitrary zero to one. Setting more than $n/2$ variables to one, however, results in an infeasible solution. Now we show that all decision vectors with $n/2$ ones that are not of the claimed form are dominated by some other vector. For those vectors there exist some $p, q \in \{2, \ldots, n\}$ with $p < q$ such that $x_p = x_q = 1$ and $x_{p-1} = x_{q+1} = 0$. Inverting all these variables will increase the first objective by $2^{n-p} - 2^{n-q-1} > 0$ and the second objective by $2^{q-1} - 2^{p-2} > 0$, thus creating a dominating vector. Finally, we show that a given decision vector x of the claimed form cannot be dominated by any other decision vector. The first objective can only be improved if a pair of variables $x_j = 0$ and $x_i = 1$ with $j < i$ can be found and inverted, i.e., a one-bit is moved to the left in the bit-string. To compensate for the decrease in the second objective, at least another one-bit has to be moved to the right. We are not allowed to change any of the one-bits with index smaller than j, since this would sweep off the gain in the first objective. All remaining one-bits, however, occupy already the rightmost positions in the bit-string. Due to symmetry, the same argumentation holds if we want to increase the second objective without decreasing the first objective. □

This MOKP problem has a further interesting feature: only the $n/2+1$ Pareto-optimal decision vectors of the first type are so-called 'supported' solutions. Only for a supported solution there exists a weight combination such that the solution is the maximum of $\sum_{i=1}^{m} w_i f_i(x)$, the single-objective surrogate problem given by a weighted sum of the objectives.

Proposition 2. Let $\Phi(x) \stackrel{\text{def}}{=} wf_1(x)+(1-w)f_2(x)$. $w \in [0,1]$. For each Pareto-optimal decision vector x'' of Type 2 and each $w \in [0,1]$, there exists some Pareto-optimal vector x' of Type 1 such that $\Phi(x') > \Phi(x'')$.

Proof. Let w.l.o.g. x'' be a Pareto-optimal vector of Type 2, where the bit with index $b+c+1$ is set to one. Let x', x''' be defined as the two vectors identical with x'' at $n-2$ positions, where only this one-bit is moved to position $b+1$ and $n/2+b+1$, respectively. Clearly, these x', x''' belong to Type 1. We show that $(\Phi(x') + \Phi(x'''))/2 > \Phi(x'')$ from which follows that at least one of the two values $\Phi(x')$ and $\Phi(x''')$ must be larger than $\Phi(x'')$.

$$
\begin{aligned}
(\Phi(x') + \Phi(x'''))/2 &> \Phi(x'') \\
\Leftrightarrow w(2^{n-b-2} + 2^{n-n/2-b-2}) + & \\
(1-w)(2^{b-1} + 2^{n/2+b-1}) &> w2^{n-b-c-1} + (1-w)2^{b+c} \\
\Leftrightarrow w(2^{-1} + 2^{-n/2-1}) + & \\
(1-w)(2^{-1} + 2^{n/2-1}) &> w2^{-c} + (1-w)2^{c}
\end{aligned}
$$

The last inequality is fulfilled for $1 \leq c \leq n/2-1$, which holds for all possible decision vectors of Type 2 according to the definition in Proposition 1. □

Thus, we cannot expect to identify all Pareto-optimal solutions by using multiple runs of a single-objective optimizer using a weighted sum aggregation of the objectives with different weight combinations.

Our goal is therefore to investigate, how long different multiobjective EAs take to find the whole Pareto set X^* of MOKP. The algorithms we consider are the SEMO (Simple Evolutionary Multiobjective Optimizer) and the FEMO (Fair Evolutionary Multiobjective Optimizer). In contrast to the original version of Laumanns et al. (2002), we use the independent-bit mutation, which results, according to the definition of Giel (2003), in the global SEMO and global FEMO. Both algorithms were so far only applied to unconstrained problems, so the question arises of how to deal with the constraint here. Our idea is to adapt and generalize the order relation of Definition 2, which so far only involves the objectives and ignores the constraints. This approach makes it

Algorithm 1 Global SEMO

Choose an initial individual x uniformly from X
$P \leftarrow \{x\}$
loop
  Select one element x out of P uniformly.
  Create x' by flipping each bit of x with probability $1/n$
  if $\nexists y \in P$ such that $(y \succ x' \vee f(y) = f(x'))$ **then**
    $P \leftarrow (P \setminus \{z \in P | x' \succ z\}) \cup \{x'\}$
  end if
end loop

Algorithm 2 Global FEMO

Choose an initial individual x uniformly from X
$w(x) \leftarrow 0$ {Initialize offspring count}
$P \leftarrow \{x\}$
loop
  Select one element x out of $\{y \in P \,|\, w(y) \leq w(z)\ \forall z \in P\}$ uniformly.
  $w(x) \leftarrow w(x) + 1$ {Increment offspring count}
  Create x' by flipping each bit of x with probability $1/n$
  if $\nexists y \in P$ such that $(y \succ x' \vee f(y) = f(x'))$ **then**
    $P \leftarrow (P \setminus \{z \in P | x' \succ z\}) \cup \{x'\}$
    $w(x') \leftarrow 0$ {Initialize offspring count}
  end if
end loop

unnecessary to incorporate a dedicated constraint handling technique in the definition of the algorithm. The algorithms can therefore remain unchanged.

Definition 3. Let the MOKP according to Definition 1 be given and let

$$g'(x) \stackrel{\text{def}}{=} \begin{cases} n/2 - g(x) & \text{if } g(x) > n/2 \\ 0 & \text{else.} \end{cases}$$

The order relation $\succ \subseteq X \times X$ is defined as

$$x \succ y :\Leftrightarrow [g'(x) > g'(y)] \vee [g(x) = 0 \wedge x \succ_f y].$$

Since the algorithms do not have an explicit stopping rule, we are interested in their running time until all elements of the Pareto set have been identified and are contained in the internal memory P of the algorithm.

Definition 4. (Running Time) Let an algorithm $A_{f,g}$ be given which calculates the pair $(f(x), g(x))$ of objective and constraint values for each decision vector x given as input. Let $P^{(t)}, t \in \mathbb{N}$, denote the population of a multiobjective optimization algorithm applied to problem (f, g) after t calls to $A_{f,g}$. Then the random variable T denoting the running time of A on (f, g) is defined as the minimum t such that $X^* \subseteq P^{(t)}$.

3. Analysis

The analysis is based on bounding the number of times a decision alternative will undergo mutation, and hence contribute to the total running time to the algorithm. The analysis will be split into two parts. The reason for this is that we have to handle Pareto-optimal and non-Pareto optimal decision vectors differently: all non-Pareto optimal points can be improved by mutation and have therefore a limited lifetime after which a better point is found and the original one is removed from the population. Pareto-optimal solutions, in contrast, stay in the population forever, and therefore need to be accounted for with a different technique. In the first part, we investigate the number of objective function evaluations caused by mutating solutions that are not Pareto-optimal. The general upper bound technique based on decision space partition helps us to derive a common upper bound for SEMO and FEMO together. In the second part, we calculate the remaining time by considering mutations of Pareto-optimal solutions only. Here, we apply the general graph search method to derive different bounds for SEMO and FEMO.

3.1. Part I: Towards the Pareto Set

The idea behind the general upper bound lemma below is as follows. We require the selection operator never to accept any dominated solution and always to accept any non-dominated one (and never discard it unless it becomes dominated by another solution). Such a simple elitist selection strategy is, for instance, employed in our algorithms SEMO and FEMO. Every suboptimal solution can be improved with positive probability by mutation. This allows us to individually bound their maximum 'lifetime', i.e., the expected number of mutation trials of every solution until a better (dominating) individual is produced. The summation over all these individual lifetimes gives an upper bound for the expected number of mutations, and hence objective function evaluations, necessary for all suboptimal solutions. The number of terms

in this sum, which is given by the size of the decision space, can be reduced substantially by a suitable grouping of decision vectors.

Lemma 1. (Decision Space Partition Lemma, Laumanns et al.*, 2004)* Let the dominated part of the decision space, $X \setminus X^*$ be partitioned into k sets $X_1, \ldots, X_k$ with $\bigcup_{1 \le i \le k} X_i = X \setminus X^*$ and $X_i \cap X_j = \emptyset$ for all i, j. Let the dominance relation on sets be defined as

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

The sets $d(X_i) := \{X_j : X_j \succ X_i\}$ contain all sets X_j that dominate set X_i.

Let an algorithm be given that iteratively modifies a population P by a sequence of mutation and selection operations with the following properties.

(1) For each X_i, the probability that the mutation operator applied to any $x \in X_i$ produces an individual x' in a dominating decision space subset from $d(X_i)$ is bounded below by $p(X_i) > 0$, i.e., $0 < p(X_i) \le \min_{x \in X_i}\{\text{Prob}\{x' \in d(X_i) | x \in X_i\}\}$.

(2) A newly generated decision vector will enter the population P only if it is not dominated by any other element of P.

(3) 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 mutation operator is applied to non-Pareto optimal decision vectors is bounded above by $\sum_{i=1}^{k} p(X_i)^{-1}$.

To apply this general lemma, the following steps have to be completed.

1. We have to define a suitable decision space partition.
2. We have to find and prove suitable lower bounds on the mutation success probabilities for each subset in the partition.
3. We have to prove that the selection operator of the algorithm under concern fulfills properties (2) and (3).
4. We have to sum the individual mutation success probabilities.

Next, we will exemplify this procedure for both the SEMO and FEMO applied to the MOKP problem.

Step 1: The aim in finding a suitable partition is to group as many decision vectors as possible into each subset without, however, hampering the search for suitable mutation success probabilities in the next step. This might in general be very difficult or even impossible for certain problems. Here, we choose to classify the feasible decision vectors according to two aspects, the number of consecutive components set to one at the beginning and at the end:

$$\begin{aligned} X'_{i,j} := \{x \in X' \mid\ & x_j = 0, x_{n/2+i+j-1} = 0, \\ & x_s = 1 \,\forall\, (s < j \vee s > n/2 + i + j - 1)\} \\ \text{where}\quad & i \in \{2, \ldots, n/2\}, j \in \{1, \ldots, n/2 - i - 1\}. \end{aligned}$$

The infeasible solutions are simply classified according to the number of ones they contain:

$$\begin{aligned} X''_r := \{x \in X \setminus X' \mid\ & \sum_{i=1}^{n} x_i = r\} \\ \text{where}\quad & r \in \{n/2 + 1, \ldots, n\}. \end{aligned}$$

Step 2: Consider an arbitrary $x \in X''_r$. To produce an $x' \in d(X''_r)$, it is sufficient to flip one variables from one to zero and keep all variables set to zero unchanged. Therefore, $p(X''_r) \geq 1/en$. For an arbitrary $x \in X'_{i,j}$. To produce an $x' \in d(X'_{i,j})$, it is sufficient to flip both variables x_j and $x_{n/2+i+j-1}$ from zero to one, which happens with probability $1/n^2$. In the worst case, the constraint will be exceeded by 2, and we have to flip 2 out of i variables from one to zero, which happens with probability $i(i-1)/2n^2$. The remaining $n-4$ variables have to stay unchanged, which happens with probability at least $1/e$. Overall, a lower bound of $p(X'_{i,j}) := i(i-1)/2en^4$ holds.

Step 3: Both SEMO and FEMO fulfill the properties by construction: for SEMO see lines 6 and 7, for FEMO see lines 8 and 9.

Step 4:

$$\begin{aligned} & \sum_{i=2}^{n/2} \sum_{j=1}^{n/2-i-1} p(X'_{i,j})^{-1} + \sum_{r=n/2+1}^{n} p(X''_r)^{-1} \\ & = \sum_{i=2}^{n/2} (\frac{n}{2} - i - 1) \frac{en^4}{i(i-1)} + \sum_{r=n/2+1}^{n} en \\ & \leq 2en^5 \sum_{i=1}^{n/2-1} \frac{1}{i^2} + en^2 = O(n^5) \end{aligned}$$

Now, we can apply the decision space partition lemma to arrive at the following theorem bounding the expected number of function evaluations of both SEMO and FEMO in suboptimal regions.

Theorem 1. For SEMO and FEMO applied to the MOKP problem, the expected number of objective function evaluations caused by mutants of suboptimal solutions, i.e., solutions from $X \setminus X^*$, is bounded above by $O(n^5)$.

3.2. Part II: On the Pareto Set

The lemma we applied in the previous section is based on the assumption of a maximum lifetime of a suboptimal solution in the population. Once a Pareto-optimal solution enters the population it will stay there forever because no further improvement is possible for this solution. Hence, we cannot resort to the decision space partition technique when accounting for mutations of Pareto-optimal solutions. Instead of asking how much time it takes to improve a solution, i.e., to move the population closer to the Pareto set, we will investigate now how quickly the population can expand along the Pareto set. This is the motivation of the graph search method described in the following.

We are given a random starting node v_1 and a randomized operator $\mathrm{jump} : \mathrm{V} \to \mathrm{V}$, which returns for each node v a neighbor v' of v with probability given by the edge weight $w(v, v')$. The purpose of Algorithm 3 is to determine V and E using a minimal number of calls to jump. Note that the edge weights are parameters of the problem, while the node weights are used by the algorithm as internal variables.

In previous work (Laumanns et al., 2004), we developed a theorem to bound the expected number of calls to jump in Algorithm 3. This theorem was then applied for algorithms using the selection strategy of FEMO, and under the assumption of one-bit mutations. In this case, it is straightforward to associate the node set V with X^* and the edge set E with all possible mutations, i.e., pairs of solutions that have a hamming distance of one. All edges weights are therefore equal to $1/n$. With the additional property that no mutation of a Pareto-optimal solution could result in a suboptimal solution, the theorem could be applied by setting $p = 1/n$ and determining the cardinality of V and E.

In the case of independent-bit mutations, though, this theorem cannot be applied in its current form, because every decision vector is reachable by mutation with positive probability from any decision vector. Thus, we have to consider (i) that Pareto-optimal solutions can always produce suboptimal solutions by mutation, and (ii) $G = (V, E) =$

Algorithm 3 Randomized Graph Search

```
1: w(v_1) ← 0
2: V ← {v_1}; E ← {}
3: loop
4:    Select a node v' out of {v' ∈ V | w(v') ≤ w(v) ∀v ∈ V} uniformly.
5:    w(v') ← w(v') + 1
6:    v'' ← jump(v')
7:    if v'' ∉ V then
8:       w(v'') ← 0
9:       V ← V ∪ {v''}; E ← E ∪ {(v', v'')}
10:   end if
11: end loop
```

$(X^*, X^* \times X^*)$ is completely connected, with a lower bound on the edge weights as small as 2^{-n}. The first issue can be solved by arguing that even if this suboptimal solution is included into the population and subsequently selected to undergo mutation, such mutation trials are also already accounted for by the decision space partition lemma and can be ignored here. To solve the second issue, we have will derive a stronger version of this theorem, denoted below as the General Graph Search Lemma. The idea here is to work with edge sets that represent spanning trees. This guarantees that all elements are reachable successively by mutation, and minimizes the number of necessary edges to be traversed by mutation. All mutations that correspond to edges not belonging to the spanning tree can be ignored. Out of all possible spanning trees, we look for the one with the largest possible lower bound on the edge weights. In the proof we apply the technique of the investigation of typical runs proposed by Wegener (2000).

Lemma 2. (General Graph Search Lemma) If p is a lower bound of the edge weights of a spanning tree of $G = (V, E)$ then Algorithm 3 has found all nodes and edges of G after $(c+1)\frac{|V|}{p} \ln |V|$ calls to jump with probability at least $1 - |V|^{-c}$. The expected number of calls to jump is bounded by $O(\frac{|V|}{p} \log |V|)$.

Proof. Let a typical run of the algorithm defined by the property that there is no node $v \in V$, for which jump has been called more than $\lambda = \frac{c+1}{p} \ln |V|$ times at the moment where the last missing node has been found. This implies that jump has not been called more than $(c+1)\frac{|V|}{p} \ln |V|$ times. To prove the first claim of the lemma, we have to bound the failure probability, i.e., the probability that a run is not typical, by $|V|^{-c}$. As soon as the initial node v_1 is fixed, this node is

defined as the root of the considered spanning tree. Let $a(v_i)$ be the direct predecessor of node v_i in this spanning tree. For each $v_i, i \in \{2\ldots,|V|\}$, we define the individual failure probabilities $q(v_i)$ as the probability of the event that none of the calls of jump$(a(v_i))$ returns v_i. With p being a lower bound on the edge weights, $q(v_i) \leq (1-p)^\lambda \leq |V|^{-(c+1)}$. The total failure probability can at most be the sum of the individual failure probabilities over all nodes to be found. With $\sum_{i=2}^{|} V| \leq |V|^{-c}$, the first part of the proof is completed. For the expected number of calls, let the random variable J denote the number of calls to jump, divided by $\frac{|V|}{p}\ln|V|$. With

$$\begin{aligned} E(J) &\leq 1\cdot\text{Prob}\{0\leq J<1\}+2\cdot\text{Prob}\{1\leq J<2\}+\ldots \\ &\leq 3+\sum_{c=3}^{\infty} c\cdot\text{Prob}\{J\geq c-1\} \\ &\leq 3+\sum_{c=1}^{\infty}(c+2)|V|^{-c} \\ &\leq 3+\frac{|V|}{(|V|-1)^2}+\frac{2}{|V|-1}=O(1), \end{aligned}$$

the bound on the expected number of calls to jump follows. □

For our MOKP, it is possible to construct a spanning tree of X^* with a lower edge weight bound of $p = 1/en^2$ (see Figure 1). Therefore, and since $|V| = |X^*| = |E| - 1$, the following theorem holds.

Theorem 2. For FEMO applied to the MOKP problem, the expected number of objective function evaluations caused by mutants of Pareto-optimal solutions is bounded above by $O(n^4 \log n)$.

Proof. Consider the spanning tree of X^*, where $E' := \{(x_a, x_b)$ such that

$$x_a = (\underbrace{1,1,\ldots,1}_{a},\underbrace{0,0,\ldots,0}_{\frac{n}{2}},\underbrace{1,1,\ldots,1}_{\frac{n}{2}-a}), a \in \{1,\ldots,n/2\},$$

$$x_b = (\underbrace{1,1,\ldots,1}_{a-1},\underbrace{0,0,\ldots,0}_{b},1,\underbrace{0,0,\ldots,0}_{\frac{n}{2}-b},\underbrace{1,1,\ldots,1}_{\frac{n}{2}-a}), b \in \{1,\ldots,n/2\}\}.$$

Every two nodes connected by an edge have a hamming distance of two, so the probability that the mutation operator applied to either of them produces the other one is at least $1/en^2$. Now assume that Algorithm 3 is applied to find (X^*, E') and initialized with v_1 as the first Pareto-optimal solution found by FEMO. With $p = 1/en^2$ and

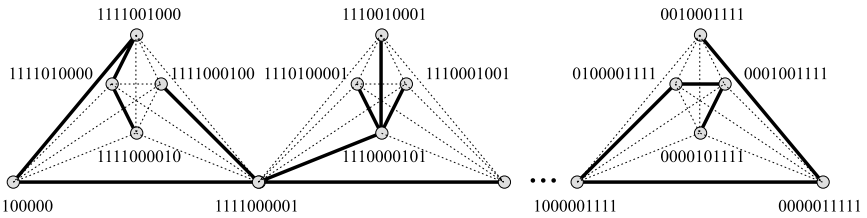

Figure 1. Graph whose node set represents the Pareto set of MOKP with $n = 10$. The dotted edges connect all points with hamming distance two. The edges marked in black constitute a possible spanning tree.

Theorem 2, the expected number of iterations J of this algorithm is $O(n^4 \log n)$. In FEMO, though, it might occur that the population P contains both Pareto-optimal and non-optimal solutions at the same time. Let the random variable T' be defined as the number of function evaluations caused by mutants of Pareto-optimal solutions. If a non-optimal solution is selected in line 5 of FEMO, the respective objective function evaluation is not an event which T' accounts for. The new solution produced in this iteration can nevertheless be Pareto-optimal. The possibility of such events can only lower the failure probabilities of Lemma 2. Therefore, $\text{Prob}\{\text{T}' \leq \text{t}\} \leq \text{Prob}\{\text{J} \leq \text{t}\}$ for all t, and hence $E(T') \leq E(J) = O(n^4 \log n)$. □

Algorithm 3 represents the selection and sampling scheme of FEMO. In order to model the behavior of SEMO, we simply have to replace line 4 by

4: Select a node v out of V uniformly.

Lemma 3. For Algorithm 3, modified as above, the expected number of calls to jump until all nodes and edges of $G = (V, E)$ are found is bounded by $p^{-1}|V|^2$, where p is a lower bound of the edge weights of a spanning tree of G.

Proof. As long as not all nodes of V have been found, the spanning tree contains at least one node v with a so far unmarked edge (v, v') joining v with a yet undiscovered node v'. The probability to select this v in line 4 is at least $1/|V|$, and the probability that the immediate call to jump returns v' is at least p. Thus, the expected waiting time until the number of found nodes increases by one is at most $|V|/p$. As at most $|V| - 1$ such increasing steps are necessary, the claim follows. □

With this lemma, the following theorem holds:

Theorem 3. For SEMO applied to the MOKP problem, the expected number of objective function evaluations caused by mutants of Pareto-optimal solutions is bounded above by $O(n^6)$.

Together with the results of Part I, the expected total running time of SEMO and FEMO can be bounded as:

Corollary 1. To find the Pareto set of MOKP, the expected running time of SEMO can be bounded by $O(n^6)$ and the expected running time of FEMO by $O(n^5)$.

In order to judge the efficiency of the algorithms based on these results, we could hypothetically distribute the total running time evenly among the $\Theta(n^2)$ Pareto-optimal solutions found: FEMO 'uses' about $O(n^3)$ time and SEMO about $O(n^4)$ time on average per Pareto-optimal solution. As there are no results of other algorithms available, these numbers are difficult to interpret. A single-objective (1+1)-EA, for instance, needs $\Theta(n^2)$ time to find the optimum of any weighted sum of the two objectives of MOKP. However, as discussed earlier, only the $n/2-1$ supported Pareto-optimal points can be found in this way, and only if the right weight combination would be known beforehand, while the remaining $\Omega(n^2)$ Pareto-optimal points are not guaranteed to be found by this approach.

4. Conclusion

We have demonstrated the use of two general upper bound methods to analyze the running time of population-based multiobjective evolutionary algorithms with simple elitist selection rules and independent-bit mutations. A simple two-objective instance of the knapsack problem was defined, where the minimum hamming distance between Pareto-optimal solutions is two so that algorithms with one-bit mutations are not able to solve this problem. For two specific algorithms, SEMO and FEMO, the methods allowed us to derive bounds on the expected running time of $O(n^6)$ and $O(n^5)$, respectively.

These techniques, the decision space partition and the graph search method, are applicable to different algorithms and a broad range of problems. It might of course be possible to derive better bounds for certain cases. This, however, has to be done on a case-by-case analysis for each problem and each algorithm separately. The power of the general methods lies in the fact that they can avoid this tedious work.

One way to show that a bound obtained by the general techniques are tight is to prove a matching lower bound of the same growth rate. Such lower bound techniques do not exist so far, and it is a challenge to develop them.

Acknowledgements

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

References

Bäck, T., D. B. Fogel, and Z. Michalewicz (eds.): 1997, *Handbook of Evolutionary Computation*. Bristol, UK: IOP Publishing and Oxford Univertity Press.

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

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

Giel, O.: 2003, 'Expected Runtimes of a Simple Multi-objective Evolutionary Algorithm'. In: *Congress on Evolutionary Computation (CEC 2003)*. Piscataway, NJ.

Jaszkiewicz, A.: 2002, 'On the performance of multiple objective genetic local search on the 0/1 knapsack problem. A comparative experiment'. *IEEE Transactions on Evolutionary Computation* **6**(4), 402–412.

Laumanns, M., L. Thiele, and E. Zitzler: 2004, 'Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions'. *IEEE Transactions on Evolutionary Computation*. Accepted for publication.

Laumanns, M., L. Thiele, E. Zitzler, E. Welzl, and K. Deb: 2002, 'Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem'. In: *Parallel Problem Solving From Nature (PPSN VII)*. pp. 44–53, Springer.

Schaffer, J. D.: 1985, 'Multiple Objective Optimization with Vector Evaluated Genetic Algorithms'. In: J. J. Grefenstette (ed.): *Proceedings of an International Conference on Genetic Algorithms and Their Applications*. pp. 93–100.

Scharnow, J., K. Tinnefeld, and I. Wegener: 2002, 'Fitness Landscapes Based on Sorting and Shortest Paths Problems'. In: J. J. M. Guervas et al. (eds.): *Parallel Problem Solving From Nature (PPSN VII)*. pp. 54–63, Springer.

Wegener, I.: 2000, 'Methods for the analysis of evolutionary algorithms on pseudo-boolean functions'. In: R. Sarker, X. Yao, and M. Mohammadian (eds.): *Evolutionary Optimization*. Kluwer, pp. 349–369.

Zitzler, E. and L. Thiele: 1999, 'Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach'. *IEEE Transactions on Evolutionary Computation* **3**(4), 257–271.