On the Impact of Utility Functions in Interactive Evolutionary Multi-Objective Optimization

Frank Neumann and Anh Quang Nguyen

Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia

Abstract. Interactive evolutionary algorithms for multi-objective optimization have gained an increasing interest in recent years. As multi-objective optimization usually deals with the optimization of conflicting objectives, a decision maker is involved in the optimization process when encountering incomparable solutions. We study the impact of a decision maker from a theoretical perspective and analyze the runtime of evolutionary algorithms until they have produced for the first time a Pareto optimal solution with the highest preference of the decision maker. Considering the linear decision maker, we show that many multi-objective optimization problems are not harder than their single-objective counterpart. Interestingly, this does not hold for a decision maker using the Chebeyshev utility function. Furthermore, we point out situations where evolutionary algorithms involving a linear decision maker have difficulties in producing an optimal solution even if the underlying single-objective problems are easy to be solved by simple evolutionary algorithms.

1 Introduction

Evolutionary algorithms (EAs) are frequently used for tackling multi-objective optimization problems [5, 4]. Multi-Objective problems usually allow for an exponential number of trade-offs with respect to the given objective functions. In the usual setting, solutions representing the different trade-offs according to given objective functions are presented to the decision maker and he then has to decide on one of these solutions for implementation.

In order to let an EA focus on regions in the objective space that are preferable to a decision maker, one can add the possibility of interacting with the algorithm. In particular, the decision maker can make the decision which solution to prefer in the case that two solutions are incomparable with respect to the classical Pareto dominance relation which drives most evolutionary multi-objective algorithms.

Interactive evolutionary multi-objective optimization has gained increasing attention during the last years [10, 6]. The goal is to involve the decision maker into the optimization process and gain knowledge about his preferences in order to focus on the regions that he prefers during the optimization run. It should

be mentioned that the preferences of the decision maker are usually not known in advance as he does not know the different possibilities of solutions and their corresponding objective vectors before starting the run of the algorithm.

The runtime analysis of interactive evolutionary multi-objective optimization has been started recently by Brockhoff et al. [3]. The authors considered the algorithms iRLS and (1+1) iEA which are interactive versions of randomized local search and the (1+1) EA [8]. The algorithms iRLS and (1+1) iEA work on the Pareto dominance relation and use the knowledge of a decision maker to decide between incomparable search points. The influence of a linear decision maker using the weighted sum and a decision maker working with the Chebyshev utility function has been analyzed for two well known example problems called LOTZ and COCZ [3].

In this paper, we investigate the setting of Brockhoff et al. [3]. Our aim is to give a general characterization of problems where the use of a linear decision maker makes a multi-objective optimization problem as easy as the optimization of its single-objective functions. Here, we assume that the multi-objective problem consists of single-objective problems of the same type, e.g. a minimum spanning tree problem or a shortest path problem. We show that the linear decision maker turns such problems from a structural point of view into single-objective problems. This implies that we can translate known runtime results of RLS and (1+1) EA to their interactive versions in the multi-objective setting. For a decision maker using the Chebyshev utility function, we show that there are instances of the multi-objective setting of the knapsack problem where the interactive algorithms have an exponential expected optimization time.

After having examined multi-objective problems with linear objective functions, we turn our attention to the LeadingOnes problem. We examine multi-objective versions motivated by recent studies in the area of black box complexity [7]. Our results point out situations where iRLS and (1+1) iEA have difficulties in obtaining optimal solution according to the linear decision maker.

The outline of the paper is as follows. In Section 2 we introduce the setting for interactive multi-objective optimization and the algorithms that are subject to our analysis. In Section 3, we show how the decision maker may prevent Deteriorative Cycles where a new produced solution is worst than the previously obtained ones .In Section 4, we present a general study on a linear decision maker for multi-objective problems having linear objective functions. For a decision maker using the Chebyshev utility function we show in Section 5 that there are instances of knapsack problem leading to an exponential optimization time. Finally, we consider the general LeadingOnes problems in Section 6 in order to point out the situations where the linear decision maker runs into difficulties and finish with some concluding remarks.

2 Interactive Multi-Objective Optimization

Throughout this paper, we investigate the impact of a decision maker who is involved in the optimization process for a given multi-objective problem. A

multi-objective optimization problem is given by a function $f\colon X \to \mathbb{R}^d$ that assigns to each element $x \in X$ of the considered search space X a vector $f(x) = (f_1(x), \ldots, f_d(x))$ consisting of d objective values. If not otherwise stated we assume that each of the d objectives should be minimized. A search point x weakly dominates a search point y ($x \preceq y$) iff $f_i(x) \leq f_i(y)$, $1 \leq i \leq d$. We say that x strongly dominates y ($x \prec y$) iff $f_i(x) \leq f_i(y)$, $1 \leq i \leq d$ and there exists an $j \in \{1, \ldots, d\}$ with $f_j(x) < f_j(y)$. Often the different objectives are in conflict with each other which means that there is no single solution which gives the minimal value for all objectives at the same time. We say that x and y are incomparable ($x \parallel y$) if neither $x \preceq y$ nor $y \preceq x$ holds. The set $X^* = \{x \in X \mid \nexists y \in X \text{ with } y \prec x\}$ is called the Pareto optimal set and the set of corresponding objective vectors $PF = \{f(x) \mid x \in X^*\}$ is called the Pareto front.

The classical goal in multi-objective optimization is to compute a set of solutions that contains for each element of PF a corresponding solution. An alternative to computing such a set of trade-offs first and presenting it later on to a decision maker who picks one of the solutions for implementation, is to involve the decision maker in the optimization process. Asking the decision maker can be in particular very helpful when making decisions between solutions that are incomparable according to the Pareto dominance relation. Our goal is to study such approaches from a theoretical perspective and examine the influence of different types of decision makers on the optimization time.

Algorithm 1 ((1+1) iEA)

1. *Choose $x \in \{0,1\}^n$ uniformly at random*
2. *Repeat*
   - *Obtain y by flipping each bit of x with probability $1/n$.*
   - *If $y \preceq x$ then $x := y$*
   - *else if $x \parallel y$ then $x := D(x, y)$.*

In this practice, we consider the interactive version of the classical (1+1) EA. This algorithm called (1+1) iEA has been introduced in [3] and is shown in Algorithm 1. (1+1) iEA starts with a solution chosen uniformly at random from the search space $X = \{0,1\}^n$. In each iteration, a new solution y is produced by flipping each bit of the current solution x with probability $1/n$. The search point x is replaced by y if y weakly dominates x ($y \preceq x$). If y is dominated by x ($x \prec y$) then x remains unchanged. If x and y are incomparable ($x \parallel y$) then the decision maker decides. The decision maker is a function $D : X \times X \to X$ which takes two search points x and y and returns one of them.

We study our algorithm with respect to the number of fitness evaluations until for the first time a Pareto optimal solution with the highest preference of the decision maker has been obtained. We call this the *optimization time* of the algorithm on a given problem. The expected number of fitness evaluations until this goal has been achieved is called the *expected optimization time*. When considering single-objective optimization problems, the *expected optimization time* is defined as the expected number of fitness evaluations until the algorithm has

produced for the first time an optimal solution with respect to the given objective function.

2.1 Decision Makers

To model the decision maker, we have to specify the function $D : X \times X \to X$. We examine the two decision makers modelled in [3]. In the following, we assume that all objective functions should be minimized, but the setting can be easily adjusted in the case that some of the given objectives should be maximized.

The first is the weighted sum approach. For a given problem, the decision maker chooses a parameter $\lambda_i \in [0,1]$, $1 \leq i \leq d$ with $\sum_{i=1}^{d} \lambda_i = 1$, and sets $D(x,y) = y$ if

$$\sum_{i=1}^{d} \lambda_i f_i(y) \leq \sum_{i=1}^{d} \lambda_i f_i(x),$$

and $D(x,y) = x$ otherwise. We call this the linear (or weighted sum) decision maker.

We also consider a decision maker using the Chebyshev utility function

$$u_c(f(x)) = \max_{i \in \{1,2,\ldots,d\}} \{\lambda_i \cdot |z_i^* - f_i(x)|\}$$

where $z^* = (z_1^*, z_2^*, \ldots, z_d^*)$ is a pre-defined utopian point and $\lambda_i \in [0,1]$, $1 \leq i \leq d$ with $\sum_{i=1}^{d} \lambda_i = 1$ are the weights determined by the decision maker.

We have $D(x,y) = y$ for the decision maker using the Chebyshev utility function iff $u_c(f(y)) \leq u_c(f(x))$, and $D(x,y) = x$ otherwise.

3 Deteriorative Cycles

During the optimization run evolutionary algorithms for multi-objective optimization may produce solutions that are worse than solutions obtained previously with respect to the Pareto dominance relation [9]. Evolutionary algorithms for multi-objective optimization problems often encounter the problem of such deteriorative cycles. This is, in particular, the case if the algorithm has already obtained solutions that are close to the Pareto front. In this section, we study how the decision maker may prevent such behaviour. For a detailed discussion on the underlying principles of deteriorative cycles in the context of evolutionary multi-objective optimization we refer the reader to [2].

The decision maker does not necessarily impose a total order on the search space as it is the case for single-objective problems. The reason is that the order among the search points may not be transitive.

As an example (see Figure 1) considers three search points a, b, c with objective vectors $f(a) = (5,5)$, $f(b) = (4,4)$, $f(c) = (3,6)$ and let the preference of the decision maker be $D(b,c) = c$ and $D(c,a) = a$. In this way, an algorithm could move from a to b to c and back to a. Hence, an arbitrary decision maker does not prevent the presence of deteriorative cycles.

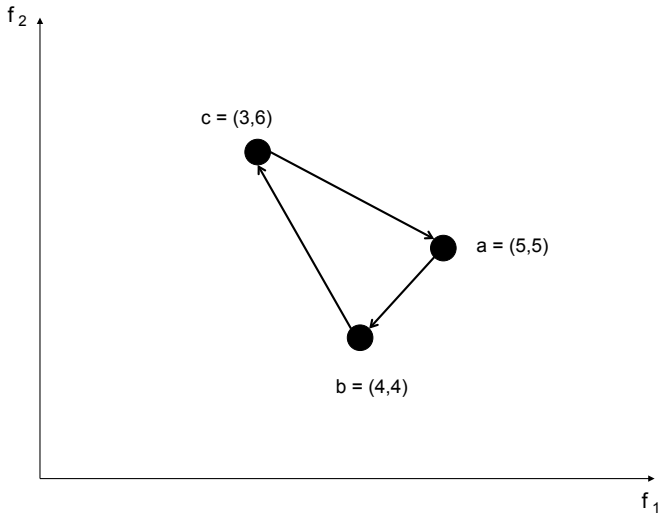

Fig. 1. Deteriorative cycle $a \to b \to c \to a$

3.1 Linear Decision Maker and Deteriorative Cycles

In the following, we show that the linear decision maker imposes a total ordering on the underlying search space which means that such an algorithm does not encounter deteriorative cycles.

Theorem 2. *The weighted sum decision maker induces a total order on the search space X.*

Proof. We show that an algorithm working with the Pareto dominance relation when considering comparable search points and working with the linear decision maker when encountering incomparable search points leads to a total order on the search space X.

Let $f\colon X \to \mathbb{R}^d$ and $x \preceq y$ iff $f_i(x) \leq f_i(y)$, $1 \leq i \leq d$. We define the order $(\preceq_L)$ given by the Pareto dominance relation $(\preceq)$ and the one by the linear decision maker as

$$x \preceq_L y \Leftrightarrow \sum_{i=1} \lambda_i f_i(x) \leq \sum_{i=1} \lambda_i f_i(y)$$

$\preceq_L$ is a total order as each search point is assigned a real value that is given by the weighted sum of its objectives.

Obviously, if $x \parallel y$ (according to the Pareto dominance relation) then the decision maker decides according to $\preceq_L$.

If $x \preceq y$ holds, then $f_i(x) \leq f_i(y)$, $1 \leq i \leq d$ and as a consequence we have

$$\sum_{i=1} \lambda_i f_i(x) \leq \sum_{i=1} \lambda_i f_i(y)$$

and hence $x \preceq_L y$. □

The previous theorem shows that the linear decision maker prevents the presence of deteriorative cycles when working with algorithms such as (1+1) iEA. For (1+1) iEA, it also ensures convergence to the set of optimal solutions as the mutation operator has a positive probability of sampling any point in the search space $\{0,1\}^n$. Having produced a solution that is minimal with respect to $\preceq_L$ implies that (1+1) iEA will never accept a solution that is not minimal with respect to $\preceq_L$. We refer the reader to [8] for an n^n upper bound on any function defined on the search space $\{0,1\}^n$. Note, that there may be more than one optimal solution with respect to the utility function of the linear decision maker.

4 Linear Decision Maker and Linear Objective Functions

Many combinatorial optimization problems have a linear objective function that has to be optimized under a given set of constraints. This includes well known problems such as the knapsack problem or the minimum spanning tree problem. In this section, we study binary optimization problems that have linear objective functions.

4.1 Linear Objective Functions

Brockhoff et al. [3] have already made the observation that if the objective functions are linear and the underlying utility function of the decision maker is the weighted sum, then the expected optimization time of iRLS and (1+1) iEA is $\Theta(n \log n)$ (see Observation 2 in [3]).

Within this section, we want to examine this observation in greater detail by studying problems with d linear objective functions and some additional constraints. Our goal is to fit classical combinatorial optimization problems into this framework. Many combinatorial optimization problems have linear objective functions, but some additional constraints. Because of the presence of constraints the optimization time is usually not $\Theta(n \log n)$. However, we are able to relate the expected optimization time to the corresponding single-objective variants with using a decision maker working with the weighted sum.

Let P be a binary optimization problem, i. e. a problem consisting of r components where the ith component is chosen iff $x_i = 1$. We say that a binary problem P with r components has a linear objective function iff the fitness of a feasible search point x is given by $f(x) := \sum_{i=1}^{r} w_i x_i$.

Note, that we currently don't assume any restrictions on the constraints that have to be met in order to obtain a feasible solution.

W.l.o.g. we consider the case where we minimize d functions $f_1, f_2, \ldots, f_d$. Cases where at least one of the objectives has to be maximized can be treated in a similar way.

In the following, we assume that a solution x is either feasible for all objective functions or feasible for none of them. We consider evolutionary algorithms

where feasible solutions are always better than infeasible solutions. For iRLS and (1+1) iEA this implies that after the algorithms have obtained a feasible solution for the first time, they will never accept an infeasible one. For the following theorem, we assume that the algorithms have already obtained a feasible solution.

Theorem 3. *Let P be a binary problem with a linear objective function and T be an upper bound on the expected optimization time of (1+1) EA on any input instance I of P when starting with an arbitrary feasible solution. Then the expected optimization time of (1+1) iEA using a linear decision maker is upper bounded by T when starting with a feasible solution.*

Proof. Let $I_1, I_2, \ldots, I_d$ be the single objective problems with objective functions

$$f_j(x) := \sum_{i=1}^{r} w_i^j x_i \qquad 1 \leq j \leq d$$

For a given fixed λ_j, $1 \leq j \leq d$, with $\sum_{j=1}^{d} \lambda_j = 1$, let

$$\begin{aligned} g(x) &= \sum_{j=1}^{d} \lambda_j f_j(x) = \sum_{j=1}^{d} \lambda_j \left(\sum_{i=1}^{r} w_i^j x_i \right) \\ &= \sum_{i=1}^{r} \left(\sum_{j=1}^{d} \lambda_j w_i^j \right) x_i = \sum_{i=1}^{r} g_i x_i \end{aligned}$$

where $g_i = \sum_{j=1}^{d} \lambda_j w_i^j$. Note that g_i is completely determined by the input and the linear preference of the decision maker expressed by the choice of λ_j, $1 \leq j \leq d$.

We claim that (1+1) EA working on g accepts an offspring y of x iff (1+1) iEA working on $(f_1, f_2, \ldots f_d)$ accepts the offspring y of x.

We first assume that x and y are incomparable ($x \parallel y$). In this case, the decision maker involved in (1+1) iEA accepts y iff $g(y) \leq g(x)$. Hence, y is accepted iff it is accepted by (1+1) EA working on g.

Secondly, we assume that x and y are comparable. If $y \preceq x$ then $g(y) \leq g(x)$ and y is accepted by (1+1) EA and (1+1) iEA. If $x \prec y$, then $g(x) < g(y)$ and y is rejected by the (1+1) EA and (1+1) iEA. □

4.2 The Knapsack Problem

In the knapsack problem the input is given by n items $1, \ldots, n$ where each item has a positive profit p_i and a positive weight w_i.

We consider the multi-objective setting for the problem where the goal is to maximize the overall profit and minimize the overall weight of the set of chosen items. We consider the search space $\{0,1\}^n$. For a bit-string x, item i is chosen iff $x_i = 1$. The fitness function $f : \{0,1\}^n \to \mathbb{R}^2$ is given by

$$f(x) = (p(x), w(x))$$

with

$$p(x) = \sum_{i=1}^{n} p_i x_i \text{ and } w(x) = \sum_{i=1}^{n} w_i x_i.$$

In the multi-objective setting, our goal is to maximize p and minimize w which introduces a partial order on the search points. $x \preceq y$ holds iff $p(x) \geq p(y)$ and $w(x) \leq w(y)$.

In order, to put it into our framework of minimizing all objectives, we can consider the case where we minimize w and minimize $-p$. If $x \parallel y$, the decision maker decides whether the new solution is accepted. For a fixed $\lambda \in [0, 1]$, D(x,y)=y holds iff

$$(1 - \lambda)w(y) - \lambda p(y) \leq (1 - \lambda)w(x) - \lambda p(x)$$

and $D(x, y) = x$ otherwise.

The multi-objective formulation of the knapsack problem consists of two linear functions without any additional constraints. It is well-known that RLS and (1+1) EA optimize each linear function in time $O(n \log n)$ [8]. Together with Theorem 3, we get the following result.

Theorem 4. *Using the weighted sum utility function, the expected optimization of (1+1) iEA for the Knapsack problem is $O(n \log n)$.*

Using Theorem 3, similar results can be obtained for other multi-objective versions of classical combinatorial optimization problems having linear objective functions. For example, the runtime results on minimum spanning trees [11] and single-source shortest paths [1] can be transferred to the corresponding multi-objective problems when considering a linear decision maker.

5 Chebyshev utility function and the knapsack problem

In the following, we examine the use of a decision maker using the Chebyshev utility function. Our goal is to show that this decision maker makes it much more difficult to find the solution with the optimal preference even if there are two linear objective functions without any additional constraints.

We consider the following trap instance called KNAP2 which has been introduced in [13] in the context of the runtime analysis of evolutionary algorithms for constraint optimization. Let $p_1 = n, p_2 = \cdots = p_n = 1$ and $w_1 = n - 1, w_2 = \cdots = w_n = 1$. For the weight bound $W = n - 1$ has been chosen in [13] which implies that in the optimal solution only the first item is chosen.

For the multi-objective setting and the Chebyshev utility function we set the utopian point to $z^* = (2n, -2)$. This meets the requirement of an utopian point as $\sum_{i=1}^{n} p_i = 2n - 1 < 2n$ and each weight is positive and therefore greater than -2. Furthermore, we set $\lambda_1 = \lambda_2 = 1/2$.

The optimal solution is the string $x^* = (1, 0 \dots 0)$ where $f(x^*) = (n, n-1)$ and $u_c(f(x^*)) = \max\{\frac{1}{2} \cdot (2n-n), \frac{1}{2} \cdot |-2-(n-1)|\} = \frac{n+1}{2}$. x^* dominates the search point $x_l = (0, 1 \dots 1)$ with $f(x_l) = (n-1, n-1)$ and $u_c(f(x_l)) = \max\{\frac{1}{2} \cdot (2n-(n-1)), \frac{1}{2} \cdot |-2-(n-1)|\} = \frac{n+1}{2}$. Furthermore, x^* and x_l are incomparable to any other search point $y \in \{0,1\}^n \setminus \{x^*, x_l\}$.

Consider a search point y where $y = (0, y_1)$ which starts with a 0-bit and has i, $0 \leq i \leq n-2$, ones in the remaining part y_1. Clearly $f(x) = (i, i)$ and $u_c(f(y)) = \max\{\frac{1}{2} \cdot (2n-i), \frac{1}{2} \cdot (i+2)\} = \frac{2n-i}{2} \geq \frac{n+2}{2} > \frac{n+1}{2}$.

Consider a search point y where $y = (1, y_1)$ which starts with a 1-bit and has i, $1 \leq i \leq n-1$, ones in the remaining part y_1. Clearly $f(x) = (n+i, n+i-1)$ and $u_c(f(y)) = \max\{\frac{1}{2} \cdot (n-i), \frac{1}{2} \cdot (n+i-1+2)\} = \frac{n+i+1}{2} \geq \frac{n+2}{2} > \frac{n+1}{2}$.

Theorem 5. *Using the weighted Chebyshev utility function u_c with $z^* = (2n, -2)$ and $\lambda_1 = \lambda_2 = 1/2$, the optimization time of the (1+1) iEA on KNAP2 is $e^{\Omega(n)}$ with probability $\alpha = \Omega(1)$.*

Proof. The first bit is set with probability 1/2 to 1 and with probability 1/2 to 0 in the initial solution. We claim that this decides on whether the algorithm ends up in the local optimum x_l or the global one x^*.

Let $x^i = (x_1^i y_1)$ be the initial solution. Suppose that $x_1^i = 0$ holds (which happens with probability 1/2). The part y_1 has at least $n/2 - \epsilon n$, $\epsilon > 0$ a constant, 1-bits with probability $1 - e^{-\Omega(n)}$ using Chernoff bounds.

Consider a phase of $T = cn$ steps where c is an appropriate constant. We claim that the number of 1-bits in y_1 is at least $n/2 + \epsilon n$ and that the bit x_1 has not been flipped during this phase. A solution with a 0 at the first bit and i 1-bits in the y_1 part has fitness (i, i) and utility value $\frac{(2n-i)}{2}$. Hence, a solution increasing the number of 1-bits in y_1 is accepted.

As long as y_1 does not contain at least $n/2 + \epsilon n$ 1-bits, the probability of increasing the number of 1-bits in y_1 is at least

$$(n/2 - \epsilon n) \frac{1}{n} \cdot (1 - 1/n)^{n-1} \geq (n/2 - \epsilon n)/(en).$$

The expected time to have obtained a solution with at least $n/2 + \epsilon n$, $\epsilon > 0$ a small constant, 1-bits is at most

$$2\epsilon n(en/(n/2 - \epsilon n)) \leq 2\epsilon 3en = 6e\epsilon n.$$

The probability that the bit x_1 has not been flipped in T steps is

$$(1 - 1/n)^T = (1 - 1/n)^{cn} > \left(\frac{1}{2e}\right)^c.$$

We set $T = \alpha_1 \cdot 6e\epsilon n$. This implies that the probability of not having obtain at least $n/2 + \epsilon n$ 1-bits in y_1 is upper bounded by $1/(\alpha_1)$ using Markov's inequality and the probability that x_1 has not been flipped is at least

$$\left(\frac{1}{2e}\right)^{\alpha_1 \cdot 6e\epsilon} = \Omega(1).$$

Having obtained this solution starting with 0 and having at least $n/2+\epsilon n$ 1-bits in the y_1 part, the utility value is at most $\frac{1.5n-\epsilon n}{2}$. A solution starting with a 1-bit and having at least $n/2$ 1-bits in the y_1-part has utility value at least $\frac{1.5n+1}{2}$ and is therefore not accepted as an offspring. Hence, only a solution having a 1-bit at the first position is accepted if at least ϵn bits flip at the same time in a single mutation step. The probability that ϵn bits flip in a single mutation steps is asymptotically Poisson distributed with parameter 1 and therefore $e^{-\Omega(\epsilon n)}$. This implies that the optimization time of (1+1) iEA is $e^{\Omega(n)}$ with probability $\Omega(1)$. □

6 The Multi-Objective Leading Ones Problem

In this section, we investigate when using a linear utility function to model the decision maker leads to problems in the optimization process. To do this, we consider a generalization of the classical Leading Ones problem and present exponential lower bounds for the considered multi-objective problem.

Leading Ones (LO) problem was first introduced in [12] and counts the number of leading ones in a given bitstring. It is defined as

$$LO(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$$

Motivated by the work in [7], where the complexity of black-box optimization on LO was analyzed, we introduce a new problem similar to the traditional Leading Ones. Given a predefined vector $a \in \{0,1\}^n$,

$$LO_a(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} (1 - |x_j - a_j|)$$

counts the number of leading bits of the given solution x that agrees with a. Given two vectors $a, b \in \{0,1\}^n$, we consider a bi-objective maximization problem with objective function $MLO_{a,b}(x) = (LO_a(x), LO_b(x))$. The goal is to maximize both objective functions. Obviously it is possible to generalize the problem to d objectives by having d bitstrings and measuring the agreement of a given solution with respect to them such that d objective values are computed. In this section, we are interested in showing lower bounds for iRLS and (1+1) iEA when working with the linear decision maker. We will investigate the bi-objective problem $MLO_{a,b}$ with $a = 1^t0^{n-t}$ and $b = 1^n$ for a given fixed value t and show when the algorithms are not able to obtain a solution with the maximal preference of the decision maker. Note that the problem has two Pareto optimal solutions, namely the strings $a = 1^t0^{n-t}$ and $b = 1^n$ and that the weightening of the objectives decides on which one is the string with the maximum preference according to the linear decision maker.

As we are deadling with bi-objective problems, the weightening is decided by one parameter λ, $0 \leq \lambda \leq 1$, and utility value according to the decision maker

is given by

$$\lambda \cdot LO_a(x) + (1-\lambda) \cdot LO_b(x).$$

As we are dealing with maximizing problems the utility value should be maximized as well. Note, that if $\lambda = 1/2$ then both Pareto optimal solutions have maximum utility, whereas $\lambda > 1/2$ implies that a is the optimal solution and $\lambda < 1/2$ implies that b is the optimal solution.

In the following, we assume that $0 < \lambda < 1$ as $\lambda = 0$ or $\lambda = 1$ implies that one of the objectives can be neglected and the expected optimization time would be $\Theta(n^2)$. Furthermore, we assume $\lambda > 1/2$ such that the algorithm favours a as the optimal solution. The case $\lambda < 1/2$ can be handled in a symmetric way.

Theorem 6. *Let $\lambda > 1/2$, $a = 1^t 0^{n-t}$, and $b = 1^n$. Then the optimization time of (1+1) iEA on $MLO_{a,b}$ is at least $n^{\frac{k(1-\lambda)}{2\lambda}}$ with probability $2^{-k} \cdot (1 - n^{-1/2})$ where $t = n/2$ and $t + k < n$ holds.*

Proof. If $LO_a(x) = LO_b(x) < t$ holds, the probability of increasing $LO_a(x)$ and $LO_b(x)$ is at least $1/n$. Hence, after an expected number of $O(nt)$ steps, $LO_a(x) \geq t$ and $LO_b(x) \geq t$.

Let x be the first solution in the run of the algorithm for which $LO_a(x) \geq t$ and $LO_b(x) \geq t$ holds. Since all bits at positions greater than t are still uniformly at random in x, we have $x = 1^t 1^k 0[0,1]^{n-k-t-1}$ for $n-t \geq k \geq 2$ with probability 2^{-k}. In order to reach the optimal solution a, the algorithm has to accept an offspring y of x that is incomparable to x. Consider a potential offspring $y = 1^t 0^c 1[0,1]^{n-t-c-1}$ of x such that x and y are incomparable. The solution y is accepted iff

$$g(x) - g(y) = k - \lambda k - \lambda c = k(1-\lambda) - \lambda c \leq 0 \Leftrightarrow \quad k \leq \frac{\lambda c}{1-\lambda}.$$

This implies that in one single mutation step, $c \geq \frac{k(1-\lambda)}{\lambda}$ specific bits of the current solution x must be flipped. The probability for such a mutation is at most $n^{-\frac{k(1-\lambda)}{\lambda}}$. Let $T = n^{\frac{k(1-\lambda)}{2\lambda}}$, then the probability to obtain such a solution in T steps is at most $n^{-1/2}$. Hence, with probability $2^{-k} \cdot (1-n^{-1/2})$, the runtime of (1+1) iEA on $MLO_{a,b}$ is at least $n^{\frac{k(1-\lambda)}{2\lambda}}$. □

The previous result shows that even multi-objective versions of simple problems such as Leading Ones can become difficult to solve when using a decision maker with a linear utility function.

7 Conclusions

Incorporating the decision maker into the optimization process of evolutionary multi-objective optimization has become a very popular approach. In this paper, we have studied simple evolutionary algorithms from a theoretical perspective. Our studies show that important multi-objective combinatorial optimization problems such as the multi-objective formulation of the knapsack problems,

multi-objective minimum spanning trees or multi-objective shortest paths become as easy for iRLS and (1+1) iEA as their single-objective counterparts when working with a linear decision maker. Furthermore, we have pointed out for the knapsack problem that this is in general not the case when working with the Chebyshev utility function. Our studies for the multi-objective LeadingOnes problem show situations where the algorithms using the linear decision maker fail to obtain a solution of maximal preference in expected polynomial time.

References

1. Baswana, S., Biswas, S., Doerr, B., Friedrich, T., Kurur, P.P., Neumann, F.: Computing single source shortest paths using single-objective fitness functions. In: Jansen, T., Garibay, I., Wiegand, R.P., Wu, A.S. (eds.) Proceedings of the 10th International Workshop on Foundations of Genetic Algorithms (FOGA 2009). pp. 59–66. ACM Press, Orlando, USA (2009)
2. Berghammer, R., Friedrich, T., Neumann, F.: Convergence of set-based multi-objective optimization, indicators and deteriorative cycles. Theor. Comput. Sci. 456, 2–17 (2012)
3. Brockhoff, D., López-Ibáñez, M., Naujoks, B., Rudolph, G.: Runtime analysis of simple interactive evolutionary biobjective optimization algorithms. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN (1). Lecture Notes in Computer Science, vol. 7491, pp. 123–132. Springer (2012)
4. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002)
5. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, UK (2001)
6. Deb, K., Sinha, A., Korhonen, P.J., Wallenius, J.: An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. IEEE Trans. Evolutionary Computation 14(5), 723–739 (2010)
7. Doerr, B., Winzen, C.: Black-box complexity: Breaking the o(n logn) barrier of leadingones. In: Hao, J.K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) Artificial Evolution. Lecture Notes in Computer Science, vol. 7401, pp. 205–216. Springer (2011)
8. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. 276, 51–81 (2002)
9. Hanne, T.: On the convergence of multiobjective evolutionary algorithms. European Journal of Operational Research 117(3), 553–564 (1999)
10. Jaszkiewicz, A., Branke, J.: Interactive multiobjective evolutionary algorithms. In: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (eds.) Multiobjective Optimization. Lecture Notes in Computer Science, vol. 5252, pp. 179–193. Springer (2008)
11. Neumann, F., Wegener, I.: Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theor. Comput. Sci. 378(1), 32–40 (2007)
12. Rudolph, G.: Convergence properties of evolutionary algorithms. Hamburg: Kovac (1997)
13. Zhou, Y., He, J.: A runtime analysis of evolutionary algorithms for constrained optimization problems. IEEE Trans. Evolutionary Computation 11(5), 608–619 (2007)