Runtime Analysis of Simple Interactive Evolutionary Biobjective Optimization Algorithms* **

Dimo Brockhoff[1], Manuel López-Ibáñez[2], Boris Naujoks[3], and Günter Rudolph[4]

[1] INRIA Lille - Nord Europe, DOLPHIN Team, 59650 Villeneuve d'Ascq, France
dimo.brockhoff@inria.fr
[2] IRIDIA, Université Libre de Bruxelles (ULB), Av. F. Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
manuel.lopez-ibanez@ulb.ac.be
[3] Institute for Informatics, Cologne University of Applied Sciences, Steinmüllerallee 1, D-51643 Gummersbach, Germany
boris.naujoks@fh-koeln.de
[4] Fakultät für Informatik, Technische Universität Dortmund, 44221 Dortmund, Germany
Guenter.Rudolph@tu-dortmund.de

Abstract. Development and deployment of interactive evolutionary multiobjective optimization algorithms (EMOAs) have recently gained broad interest. In this study, first steps towards a theory of interactive EMOAs are made by deriving bounds on the expected number of function evaluations and queries to a decision maker. We analyze randomized local search and the (1+1)-EA on the biobjective problems LOTZ and COCZ under the scenario that the decision maker interacts with these algorithms by providing a subjective preference whenever solutions are incomparable. It is assumed that this decision is based on the decision maker's internal utility function. We show that the performance of the interactive EMOAs may dramatically worsen if the utility function is non-linear instead of linear.

1 Introduction

Interactive algorithms for multi-criteria decision making are typically based on real-world case studies and designed to work well in practice. However, to the best of our knowledge, approaches combining interactive decision making with EMOAs have not been analyzed mathematically in terms of their runtimes and the expected number of questions asked (queries) to the decision maker (DM). For example, it is sometimes claimed that only a small number of queries are performed in practice [5, p. 135], but we are not aware of any rigorous analysis of how many queries are actually necessary to obtain the most preferred solution or an approximation thereof. Specifically for interactive EMOAs, which are *randomized* search heuristics where no guarantee can be given on when good search points are found, a theoretical understanding of their *expected* optimization time will be extremely helpful when comparing approaches and predicting their performance on future unknown problems.

* **Erratum:** The drift function in Eq. 1 in the published PPSN'2012 manuscript was originally wrongly stated and should read $g(x) = n - \left\lfloor i + \frac{1-w_1}{w_1} \cdot j \right\rfloor$. Moreover, the definition of the LOTZ problem was wrong initially and should read $TZ(x) = \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j)$ for the second objective.

** This is an updated version of the original paper, published at Parallel Problem Solving from Nature (PPSN'2012). The original publication is available at www.springerlink.com.

This paper provides a first attempt to theoretically analyze the runtime and the number of queries to the DM of interactive EMOAs. To this end, we propose two simple interactive randomized search heuristics and analyze them on the two standard binary test problems Leading Ones Trailing Zeros (LOTZ, [4]) and Counting Ones Counting Zeros (COCZ, [3]). The algorithms are interactive versions of the well-known randomized local search (RLS) and the (1+1)-EA and neither do not know the DM's preferences nor do they assume anything about the DM. Whenever the Pareto-dominance relation does not indicate a search direction between the current search point and the mutated offspring because the two solutions are incomparable, the algorithms query the DM about her opinion and accept the solution that was favored by the DM. For several *simulated* DMs, where a specific underlying utility function of the DM is assumed, tight bounds on the expected runtimes and the expected number of queries to the DM for the two interactive algorithms can be proven—using established proof techniques that have been developed to bound the expected runtime of other randomized search heuristics.

Our proofs should be seen as a starting point for analyzing interactive approaches. More involved algorithms and more complicated models of the DM will have to be analyzed in the future. It is also a first step towards understanding the consequences of adding increasingly more complicated utility functions and how algorithms should be designed to deal with them. One major conclusion from our analyses is that the performance of simple interactive EMOAs dramatically changes when the DM is acting according to linear or non-linear utility functions.

Section 2 provides a mathematical prelude and some basic assumptions before the two proposed interactive algorithms are presented in Sec. 3. The biobjective test problems are described in Sec. 4. The runtime analysis starts in Sec. 5 for the LOTZ problem, which is followed in Sec. 6 by the runtime analysis of the iRLS on COCZ. Section 7 summarizes our findings and provides an outlook to future research.

2 Mathematical Prelude and Basic Assumptions

Let $\mathbb{B}^n = \{0,1\}^n$ be the finite discrete search space of binary strings of length $n \in \mathbb{N}$. We consider the simultaneous maximization of two objective functions $f\colon \mathbb{B}^n \to \mathbb{N}_0^2$ with $f(x) = (f_1(x), f_2(x))$ and define the weak dominance relation $\succeq$ on the search space $\mathbb{B}^n$ via $x \succeq y$ iff $f_1(x) \geq f_1(y)$ and $f_2(x) \geq f_2(y)$ for $x, y \in \mathbb{B}^n$ and the dominance relation $\succ$ via $x \succ y$ iff $x \succeq y$ and $f(x) \neq f(y)$. The decision maker is interpreted as a black box or oracle that is queried to decide which of two given *objective vectors* ("$f(x)$ or $f(y)$?") is better.[5]

In its simplest form, we assume that the DM has an underlying *value* or *utility function* $u(x)$ [2, p. 68 & 80f] according to which she is making her decision

$$\mathrm{DM}(f(x), f(y)) = f(x) \cdot \mathbb{I}_{\{u(f(x))>u(f(y))\}} + f(y) \cdot \mathbb{I}_{\{u(f(x))\leq u(f(y))\}}$$

[5] Limiting the queries to objective vectors simplifies the proofs presented in this paper. In principle, the DM can also be queried about her preference among solutions x and y directly, but the theoretical results will be different: it is expected that the number of queries to the DM will increase since, for discrete problems, the total number of solution pairs is typically larger than the number of objective vector pairs.

where $\mathbb{I}_A$ is the indicator function, giving 1 iff its argument A is true, and 0 otherwise.

The utility function u is thereby a function that maps an objective vector $f(x)$ to a real value, and an objective vector $f(x)$ is said to be preferred by the DM over a vector $f(y)$ iff $u(f(x)) > u(f(y))$. In case of a *weighted sum*, the utility function is $u(f(x)) = w_1 \cdot f_1(x) + (1 - w_1) \cdot f_2(x)$ and for a *weighted Chebyshev utility function*, u is defined as $u(f(x)) = \max_{i \in \{1,2\}} \{w_i \cdot |z_i^* - f_i(x)|\}$ for every $x \in \mathbb{B}^n$, with $w = (w_1, 1-w_1) \in \mathbb{R}^2$ being the corresponding weight vector and $z^* = (z_1^*, z_2^*) \in \mathbb{R}^2$ a pre-defined utopian vector with $z_i^* \geq \max_{x \in \mathbb{B}^n} f_i(x)$ for $i \in \{1, 2\}$.

Both the weighted sum and the weighted Chebyshev utility function ensure that an objective vector $f(x)$ has a higher utility than another vector $f(y)$ if x is dominating y. Utility functions with this property are termed Pareto compliant.

3 The Interactive (1+1)-EA and (1+1)-RLS Algorithms

The simplest interactive evolutionary algorithm imaginable is based on the (1+1)-EA [1] and it is shown in Algorithm 1. After drawing a parent uniformly at random from $\mathbb{B}^n$, the algorithm iterates the following steps until the DM decides to terminate it. First, an offspring is generated by a mutation of the parent. If only a single bit chosen uniformly at random is flipped then we shall call the algorithm iRLS, whereas it is termed (1+1)-iEA if each bit is flipped independently with probability $1/n$. If the offspring dominates or equals the parent, then it is accepted and serves as parent of the next iteration. If the offspring is dominated by the parent, then it is rejected and the parent passes to the next iteration. If offspring and parent are incomparable then the DM decides which of them is accepted. We assume that the DM can be modeled by a scalar utility function so that the individual with larger utility is accepted. This utility function is assumed to be Pareto compliant such that the algorithms decide according to the DM's preferences in case of dominated or dominating offspring.

We further assume that the algorithm stores all objective vector pairs presented to the DM so far, such that the DM is never asked to rank the same pair of objective vectors more than once in order to keep the queries to the DM as low as possible.

4 The LOTZ and COCZ Problems

In the following section, we will present the analyses of the runtime behavior of the (1+1)-iEA and iRLS on two simple well-known test problems:

Definition 1. *The bi-objective maximization problem with objective function* $f(x) = (LO(x), TZ(x))$ *where*

$$LO(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j \qquad \textit{and} \qquad TZ(x) = \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j)$$

for $x \in \mathbb{B}^n$ *is termed the* Leading Ones Trailing Zeros (LOTZ) *problem.*

Algorithm 1 (1+1)-iEA and iRLS

```
1: init: choose x^0 uniformly at random; t = 0
2: repeat
3:    y ← mutate(x^t) { iRLS: 1 bit flip; (1+1)-iEA: independent bit flip }
4:    if f(y) ⪰ f(x^t) then
5:       x^{t+1} ← y
6:    else if f(x^t) ≻ f(y) then
7:       x^{t+1} ← x^t
8:    else
9:       u* ← DM(f(x^t), f(y)) {only asks the DM once about x^t and y}
10:      if u* = f(x^t) then
11:         x^{t+1} ← x^t
12:      else
13:         x^{t+1} ← y
14:   t ← t + 1
15: until DM terminates
```

Definition 2. *The bi-objective maximization problem with objective function* $f(x) = (CO(x), CZ(x))$ *where*

$$CO(x) = \sum_{i=1}^{n} x_i \qquad \textit{and} \qquad CZ(x) = \sum_{i=1}^{\lfloor n/2 \rfloor} x_i + \sum_{i=\lfloor n/2 \rfloor + 1}^{n} (1 - x_i)$$

for $x \in \mathbb{B}^n$ *is termed the* Counting Ones Counting Zeros (COCZ) *problem.*

Both problems have been the basis of the first runtime analyses of simple MOEAs such as the SEMO and global SEMO algorithms. These problems have a linear Pareto front with $n + 1$ (LOTZ) and $\lceil n/2 \rceil + 1$ (COCZ) different Pareto-optimal objective vectors. The total number of different objective vectors is $\Theta(n^2)$ in both cases. Figure 1 illustrates their objective space and the neighborhood of objective vectors with respect to 1-bit mutations.

5 Runtime Analysis of iRLS and (1+1)-iEA on LOTZ

Let us now investigate the runtime and the number of DM queries for the iRLS and the (1+1)-iEA until the most preferred solution of the LOTZ problem is found.

Theorem 1. *When optimizing the LOTZ function, the iRLS needs in expectation* $\Theta(n^2)$ *function evaluations to find the solution that is most preferred by the DM if the utility function of the DM is either a weighted sum or the weighted Chebyshev. To this end, an expected number of* $O(n)$ *queries to the DM are performed.*

Proof. With the same arguments as for the SEMO algorithm [3], the iRLS will start the optimization with high probability in the lower left corner of the objective space and needs $\Theta(n)$ improvements to reach the Pareto front. Before reaching the Pareto front for the first time, any solution produced by a 1-bit flip either dominates or is dominated

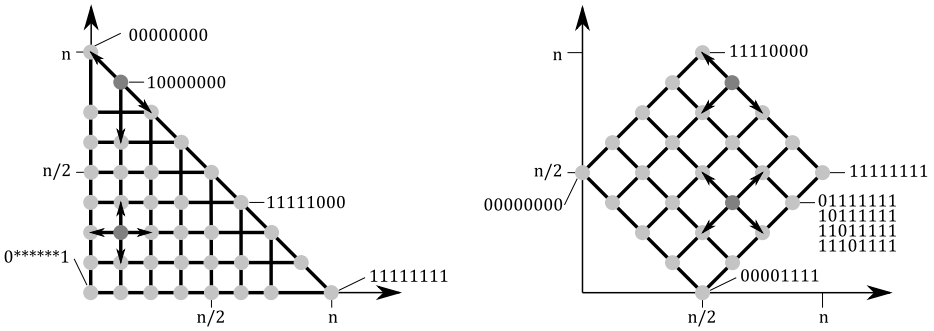

Fig. 1. Illustration of the objective space of the bi-objective LOTZ (left) and COCZ (right) problems for bitstrings of length $n = 8$. The neighborhood between objective vectors in terms of the 1-bit mutation of the iRLS is indicated with black lines and the neighbors of two example solutions for each problem are shown with arrows.

by the current solution, and, hence, the DM is never asked. An improvement towards the Pareto front is possible by flipping the first 0-bit or the last 1-bit in the current solution, which happens with a probability of $2/n$ in each iteration and corresponds to a waiting time of $O(n)$. Hence, the Pareto front is reached in $O(n^2)$ iterations. Since all bits are chosen uniformly at random in the beginning and no incentive is given to bias this uniform probability during the search, also the lower bound of $\Omega(n^2)$ steps to reach the Pareto front holds because in expectation there is less than one "free-rider" bit per improvement, cf. the argumentation in [1].

Once the Pareto front is reached for the first time, all new solutions are either dominated by the current one, and, hence, discarded immediately, or incomparable. In the latter case, the DM is asked which solution is preferred. If the DM's underlying utility function is a weighted sum, then either all Pareto-optimal solutions are equally preferred (for equal weights) or one of the extremes (1^n or 0^n) is the most preferred, due to the fact that all Pareto-optimal objective vectors of LOTZ lie on a line. In the case of equal weights, the expected runtime is, therefore, $\Theta(1)$. Otherwise, the time to reach the most preferred extreme depends on its distance from the first Pareto-optimal solution found and the number of iterations to move from one to the other. The first Pareto-optimal solution found has, with high probability, a number of leading ones (or trailing zeros) within $[1/4n, 3/4n]$ due to Chernoff bounds and the fact that the expected number of leading ones in the first Pareto-optimal point found is $n/2$ for symmetry reasons. Hence, the algorithm needs $\Theta(n)$ successful steps to move to the most preferred extreme. Moreover, the probability of each successful step is $1/n$ and its waiting time $\Theta(n)$. Hence, the most preferred extreme is reached in $\Theta(n^2)$ iterations.

Finally, the expected number of queries to the DM is $\Theta(n)$ in the case of unequal weights (and zero in the case of equal weights), because there are only $\Theta(n)$ nondominated objective vectors in the Pareto front and from each Pareto-optimal search point, at most two others can be generated by 1-bit mutations.

If the underlying utility function is a weighted Chebyshev, any Pareto-optimal solution can be the most preferred one, and, hence, depending on the location of this

solution, the number of iterations to reach it (resp. the number of DM queries) may be closer to $\Theta(n^2)$ (resp. $\Theta(n)$) or closer to $\Theta(1)$. In any case, the above upper bounds on the runtime ($O(n^2)$) and number of DM queries ($O(n)$) hold. □

Overall, we proved the expected runtime of the iRLS to be $\Theta(n^2)$: $\Theta(n^2)$ to reach the Pareto front and an additional $O(n^2)$ to find the most preferred Pareto-optimal point. If a weighted sum is assumed as the DM's utility function, the expected number of queries to the DM is either $\Theta(1)$, if all Pareto-optimal points are equally preferred, or $\Theta(n)$, otherwise. If the utility function of the DM turns out to be a weighted Chebyshev function, the expected number of DM queries depends on the weights of the Chebyshev function and can be only bounded by $O(n)$ from above and $\Omega(1)$ from below as both cases are possible (if one of the extremes or the point on the diagonal is preferred).

It turns out that the above proven bounds on the runtime and the number of DM queries do not hold for arbitrary utility functions:

Observation 1 *Already for quadratic utility functions which are Pareto compliant, it can happen that the expected runtime of the iRLS is not finite anymore.*

Proof. Assuming that the quadratic utility function $u(f(x)) = -(n - f_1(x) + 1) \cdot (n - f_2(x) + 2)$ has to be maximized, the most preferred solution is the rightmost Pareto-optimal point, i.e., the objective vector with largest f_1-value. When comparing two incomparable solutions to the left of the objective space's diagonal, the objective vector with smaller f_1- and larger f_2-value will be preferred by the DM. Hence, iRLS will converge towards the leftmost Pareto-optimal point, from which the most preferred search point cannot be reached anymore because an n-bit flip would be necessary to jump there. As the probability of reaching this left extreme of the Pareto-optimal front is non-zero, the expected runtime is not finite anymore. □

With a similar argumentation, it can be shown that the (1+1)-iEA can have an exponential runtime if the above quadratic utility function has to be maximized. Hence, let us consider the (1+1)-iEA with its independent bit flip mutation only for the case of a weighted sum utility function. It turns out that, in this case, the expected runtime is the same as for the iRLS.

Theorem 2. *When optimizing the LOTZ function and if the utility function of the DM is a weighted sum, the (1+1)-iEA needs $\Theta(n^2)$ function evaluations, in expectation, to find the most preferred solution.*

Proof. For a weighted sum as utility function, we assume w.l.o.g. $w_1 \geq 0.5 \geq 1 - w_1$ and consider the following drift function $g(x)$ when the current search point x of the (1+1)-iEA is mapped to an objective vector (i, j):

$$g(x) = n - \left\lfloor i + \frac{1 - w_1}{w_1} \cdot j \right\rfloor \quad . \tag{1}$$

The intuition behind $g(x)$ is to consider the maximum number of consecutive f_1-improvements needed in the future course of the algorithm to reach the Pareto front, i.e., the maximum distance to the Pareto front in f_1 direction over all points that have

a weighted sum utility which is not smaller than the one for the current search point. As $w_1 \geq 0.5$, it is clear that this largest distance is upper bounded by the distance in f_1-direction between the search point with objective vector $(\lfloor w_1 \cdot i + (1 - w_1) \cdot j \rfloor, 0)$ and the Pareto front, which is exactly $g(x)$.

Now, let us consider the course of $g(x)$. We have $g(x) \leq n$ and $g(x)$ never increases due to the selection of the (1+1)-iEA. Furthermore, if $g(x) \leq 1$, then the current search point is Pareto-optimal and only Pareto-optimal points will be accepted until the most preferred solution is found. We now divide the analysis into two phases: the first phase ends when the first Pareto-optimal point is found, while the second phase starts with the first Pareto-optimal point found and ends when the most preferred solution is found. The length of the first phase can be bounded from above by the time until $g(x)$ becomes smaller than 1 under the assumption that x stays non-Pareto-optimal as long as $g(x) > 1$. In this case, the probability to increase f_1 by 1 while f_2 stays constant by mutation is at least $1/n \cdot (1 - 1/n)^{n-1} \geq 1/en$ with the Euler constant $e \approx 2.71$ and therefore, in expectation, at most en steps of the (1+1)-iEA are necessary for this event, by which $g(x)$ decreases by $w_1 \geq 0.5$. Hence, in expectation, $2en$ of those steps are sufficient to decrease $g(x)$ by 1 and overall at most $2en^2$ many steps are needed in expectation to reach the Pareto front. By Chernoff bounds, the probability that in $4en^2$ steps, the Pareto front is not reached is exponentially small and the runtime bound for the first phase is proven to be $O(n^2)$.

Considering phase two, we distinguish two further cases. Either, the new solution is also Pareto-optimal or we are back in the scenario of phase one, i.e., $g(x) > 1$ and the new solution is non-Pareto-optimal. In the latter case, $g(x)$ is further decreased but we do not spend additional time because we accounted for it already in phase one. In the first case, at most $n - 1$ improvements in the first objective function value are necessary to reach the most preferred point where such an improvement happens with a specific 1-bit flip (i.e. again with a probability of at least $1/en$). The expected number of steps needed for at most $n-1$ of those improvements is then smaller than en^2 and, by Chernoff bounds, the probability to need more than $2en^2$ steps is exponentially small and the runtime for the second phase is also $O(n^2)$. □

In the above case of the (1+1)-iEA optimizing the LOTZ problem, the number of DM queries is trivially upper-bounded by $O(n^2)$ since the number of possible objective vector pairs is bounded by $O(n^2)$. However, one can show a stronger result for which we only sketch the proof here due to space limitations.

Theorem 3. *When optimizing the LOTZ function and if the utility function of the DM is a weighted sum, the (1+1)-iEA queries the DM in expectation $O(n)$ times until the most preferred solution is found.*

Sketch of Proof: From the proof of Theorem 2, we know already that the algorithm typically needs $O(n)$ improvements to reach the most preferred solution for which we need to wait $O(n)$ function evaluations each. If we know the probabilities $p_{i,j}$ to reach an incomparable search point from the current objective vector (i, j) with $1 \leq i, j \leq n$ and $i + j \leq n$ and we can upper bound them by a constant p, we can upper bound the expected number of incomparable solutions produced from (i, j) within a phase of cn

steps ($c \in \mathbb{N}$ a constant) in which an improvement is likely by the expectation of the binomially distribution with parameters p and cn:

$$\sum_{k=1}^{cn} k \cdot \binom{cn}{k} (p_{i,j})^k (1-p_{i,j})^{cn-k} \leq \sum_{k=1}^{cn} k \cdot \binom{cn}{k} p^k (1-p)^{cn-k} = cnp \ . \quad (2)$$

Note that the Eq. 2 does not take into account the fact that for each objective vector pair the DM is only asked once such that the expected number of incomparable solutions in cn steps of the (1+1)-iEA is actually smaller. When looking at the exact probabilities [6] $p_{i,j}$, it turns out that they can be easily bounded from above by $4/n$ for large enough n—independent of i and j such that Eq. 2 becomes $4c$. That means that in each phase of length cn with n large enough, only a constant number of incomparable solution is generated in expectation, which results, with high probability, in $O(n)$ incomparable solution pairs for which the DM is queried until linearly many improvements have been achieved. □

6 Runtime Analysis of the iRLS on COCZ and Linear Functions

In addition to the LOTZ problem, we now analyze the iRLS on the COCZ problem and point out how the result is related to the optimization of linear functions if the weighted sum is assumed as underlying utility function of the DM.

Theorem 4. *When optimizing the COCZ function, the iRLS needs, in expectation, $\Theta(n \log n)$ function evaluations and $O(n)$ queries to the DM to find the solution that is most preferred by the DM if the utility function of the DM is a weighted sum.*

Proof. Similar to the proof of SEMO's runtime [3], we partition the search space into sets F_i ($0 \leq i \leq \lfloor n/2 \rfloor$) such that all solutions with a number of i 1-bits in the first half of their bitstrings are in set F_i. The most important observation for proving the above theorem is that the 1-bit mutation of the iRLS allows only two scenarios. Either (i) the mutation happens in the first half of the bitstring; in this case, both objectives are perfectly correlated such that the mutated offspring y is dominating the previous solution x_t or it is dominated by it; in any case, the DM is not asked in this situation and the current search point will never fall back to a set F_i with smaller index. Or (ii) the mutation takes place in the second half of the bitstring; then, the mutated offspring y is incomparable to x_t due to the fact that both objectives are anti-correlated; both solutions belong to the same set F_i and the DM is asked to compare them.

With these observations, it is easy to prove upper and lower bounds on the running time of the iRLS on COCZ. With probability $\frac{\lfloor n/2 \rfloor - i}{n}$, the iRLS leaves the set F_i (case (i)), namely if one of the $\lfloor n/2 \rfloor - i$ zeros in the first half of the bitstring of x_t is flipped. This results in a runtime until the first Pareto-optimal point is found, of $\Theta(n \log n)$, see the argumentation for the (1+1)EA on the ONEMAX function, e.g., in [6].

[6] For each non-Pareto-optimal solution with objective vector (i, j), for example, one can upper bound the probabilities to reach every incomparable non-Pareto-optimal objective vector $(i + \Delta i, j - \Delta j)$ with $1 \leq \Delta j \leq j$ and $1 \leq \Delta i \leq n - i - j - 2 - \Delta j$ by $\frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{i+j+\Delta i - \Delta j}$ and sum those probabilities up for all possible values of $(\Delta i, \Delta j)$.

With a similar argumentation, the most preferred Pareto-optimal solution is found after (a possibly additional) $O(n \log n)$ steps as also in the second half of the bitstring, the iRLS has to perform the optimization of ONEMAX (or ZEROMAX, depending on the weight w_1). Overall, the iRLS needs an expected number of $\Theta(n \log n)$ function evaluations until the most preferred search point is found.

Regarding the number of DM queries, we argue that the algorithm performs two independent movements: (i) towards the front, where it is solely the number of ones in the first half of the bitstring that determines how far the current search point is away from the front and for which the DM is never queried, and (ii) the movement towards the extremes of the (local) Pareto front(s) F_i, where it is only important how many ones are present in the second half of the bitstring. Let us denote the objective vector of the current search point by a tuple (i, j) where i indicates the number of 1-bits in the first half and j the number of 1 bits in the second half of the bitstring of the current search point. Assuming without loss of generality, that the weight w_1 of the DM's weighted sum utility function is larger than 0.5, the rightmost Pareto-optimal point and thus the all-one-string is the most preferred solution. All accepted objective vectors, and therefore all visited (i, j) positions will lie on a line connecting the objective vector of the initial search point and this most preferred point whereas the current search point will never decrease in its i and j coordinates due to the 1-bit mutation and the acceptance step of the iRLS. Then, there are only $O(n)$ different objective vectors on this line. On the contrary, there are in expectation also $\Omega(n)$ many objective vectors on that line because, with high probability, the initial search point has about half its bits set to 0 and the other half set to 1 and thus starts with a j-value of about $n/4$. With the additional argument that for each of the $\Theta(n)$ objective vectors accepted throughout the search, maximally two questions can be asked to the DM, the overall amount of expected DM queries is proven to be $O(n)$. □

As we have seen, the iRLS asks the DM in expectation $O(n)$ times on the COCZ problem. This upper bound is due to the fact that the algorithm keeps track about which objective vector pairs have been presented to the DM in order to not ask her twice. If this property of the algorithm is relaxed towards an approach without memory, the number of DM calls increases to $\Theta(n \log n)$ in expectation.

As both objective functions of the COCZ problem are linear, this result can also be obtained from a more general analysis which even holds for the (1+1)-iEA.

Observation 2 *If both objective functions are linear functions and the underlying utility function of the DM is the weighted sum, the overall fitness function of the DM is also linear; in case, we ask the DM all the time, this will be like having to solve a linear function. It is well known that randomized local search and the (1+1)-EA, to which the iRLS and (1+1)-iEA reduce if the DM is asked at every iteration have an expected runtime on linear functions of $\Theta(n \log n)$ [7].*

7 Summary and Outlook

The theoretical analysis of interactive multiobjective evolutionary algorithms is a necessary step towards better understanding interactive approaches in order to be able to

recommend certain algorithms over others in practical optimization. In this study, we have provided the first of such analysis. The algorithms iRLS and (1+1)-iEA have been proposed which are simple variants of the well-known algorithms RLS and (1+1)-EA for single-objective optimization which ask the decision maker whenever the mutation step produces an incomparable search point. Rigorous analyses of their expected optimization time and the expected number of DM calls until the most preferred solution is found have been performed on the two well-known bi-objective binary problems LOTZ and COCZ. It turns out that the expected runtime and the number of DM calls highly depend on the assumed model of the DM, i.e., her underlying utility function. The LOTZ problem is one example where the change from a linear to a quadratic preference model changes the runtime of the iRLS from polynomial to infinite.

Though performed on basic test functions and simple algorithms which do not assume anything about the DM's preferences, our analyses open up a new research direction of analyzing more involved interactive optimization algorithms on more realistic multiobjective optimization problems. The proof techniques used in this study are standard and highly related to the proof techniques previously used to analyze population-based evolutionary multiobjective optimization algorithms. Hence, we expect that interactive approaches can be also analyzed on more complicated problems and with more complicated models of the decision maker. Furthermore, we hope that this study will initiate a discussion about the consequences of assuming increasingly more realistic utility functions and on how algorithms should be designed to deal with those.

Acknowledgments. The authors would like to thank the Research Center Schloss Dagstuhl[7] for hosting the "Learning in Multiobjective Optimization" seminar (id 12041) from which this work originates. We would also like to thank Anne Auger for many valuable discussions about the proofs and the anonymous reviewers for spotting some wrong arguments in the earlier manuscript.

References

1. Droste, S., Jansen, T., Wegener, I.: On the Analysis of the (1+1) Evolutionary Algorithm. Theoretical Computer Science 276, 51–81 (2002)
2. Keeney, R.L., Raiffa, H.: Decision with Multiple Objectives: Preferences and Value Tradeoffs. Wiley, New York (1976)
3. Laumanns, M., Thiele, L., Zitzler, E.: Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions. IEEE Transactions on Evolutionary Computation 8(2), 170–182 (2004)
4. Laumanns, M., Thiele, L., Zitzler, E., Welzl, E., Deb, K.: Running Time Analysis of Multi-Objective Evolutionary Algorithms on a Simple Discrete Optimization Problem. In: Conference on Parallel Problem Solving From Nature (PPSN VII). pp. 44–53. Springer (2002)
5. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston, MA, USA (1999)
6. Oliveto, P.S., Yao, X.: Runtime Analysis of Evolutionary Algorithms for Discrete Optimization. In: Auger, A., Doerr, B. (eds.) Theory of Randomized Search Heuristics: Foundations and Recent Developments, pp. 21–52. World Scientific Publishing (2011)

[7] `http://www.dagstuhl.de/en`

7. Witt, C.: Optimizing Linear Functions with Randomized Search Heuristics - The Robustness of Mutation. In: Dürr, C., Wilke, T. (eds.) Symposium on Theoretical Aspects of Computer Science (STACS 2012). Leibniz International Proceedings in Informatics (LIPIcs), vol. 14, pp. 420–431. Schloss Dagstuhl Leibniz-Center for Informatics (2012)