UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

On the Behavior of the (1+1) Evolutionary Algorithm on Quadratic Pseudo-Boolean Functions

Ingo Wegener and Carsten Witt

No. CI-97/00

Technical Report ISSN 1433-3325 September 2000

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

On the Behavior of the (1+1) Evolutionary Algorithm on Quadratic Pseudo-Boolean Functions*

Ingo Wegener
FB Informatik
Univ. Dortmund
44221 Dortmund, Germany
wegener@ls2.cs.uni-dortmund.de

Carsten Witt
FB Informatik
Univ. Dortmund
44221 Dortmund, Germany
witt@ls2.cs.uni-dortmund.de

August 15th, 2000

Abstract

Evolutionary algorithms are randomized search heuristics, which are often used as function optimizers. In this paper the well-known (1+1) Evolutionary Algorithm ((1+1) EA) and its multistart variants are studied. Several results on the expected runtime of the (1+1) EA on linear or unimodal functions have already been presented by other authors. This paper is focused on quadratic pseudo-boolean functions, i. e., functions of degree 2. Whereas quadratic pseudo-boolean functions without negative coefficients can be optimized efficiently by the (1+1) EA, there are quadratic functions for which the expected runtime is exponential. However, multistart variants of the (1+1) EA are very efficient for many of these functions. This is not the case with a special quadratic function for which the (1+1) EA requires exponential time with a probability exponentially close to 1. At last, some necessary conditions for exponential runtimes are examined, and an "easy" subclass within the class of quadratic functions is presented.

Keywords: Evolutionary algorithms, evolution strategies, quadratic functions, boolean functions, complexity analysis

*This work was supported by the Deutsche Forschungsgemeinschaft as part of the Collaborative Research Center "Computational Intelligence" (531).

1 Introduction

Evolutionary algorithms are randomized search heuristics which are applied in numerous areas such as function optimization, machine learning etc. Since their origin in the late 1960s, many flavors of evolutionary algorithms have emerged, amongst them Evolution Strategies [Sch95], Evolutionary Programming [Fog95], Genetic Algorithms [Hol75, Gol89], and Genetic Programming [Koz92]. Although their seemingly robust behavior in various optimization tasks was confirmed by many experiments, a solid and comprehensive theory of evolutionary algorithms is still missing. Therefore, we concentrate on the simplest variant of evolutionary algorithms, a (1+1) evolution strategy with Boolean inputs, the so-called (1+1) EA. Our credo is that theoretical results concerning the efficiency of the (1+1) EA can predict an average behavior of evolutionary algorithms incorporating larger populations, since we expect that larger populations can be imitated by parallel executions of the (1+1) EA, i. e., by *multistart strategies*. Moreover, thereby we examine an instance of evolution strategies lacking any crossover operator, which simplifies the analysis involved considerably. We do not believe that crossover can decrease the runtime of evolutionary algorithms significantly if we restrict the class of considered functions to pseudo-boolean functions of degree 2. In general, it is hard to analyze the crossover operator, and—to our knowledge—up to now only one formal proof demonstrating a helpful influence of crossover is known (see [JW99]). In all, we expect an exhaustive theory of the (1+1) EA, regarding varying functions input into the (1+1) EA, to provide insight into many aspects of the performance of more complicated evolutionary algorithms.

First, we state a formal definition of the (1+1) EA.

Definition 1 *The (1+1) EA on pseudo-boolean* fitness functions $f : \{0,1\}^n \to \mathbb{R}$ *is given by:*

1. *Set* $p_m := 1/n$.
2. *Choose randomly an initial bit string* $x \in \{0,1\}^n$.
3. *Repeat the following mutation step:*
   (a) *Compute* x' *by flipping independently each bit* x_i *with probability* p_m.
   (b) *Replace* x *by* x' *iff* $f(x') \geq f(x)$.

Since we want the (1+1) EA to be an universal optimization strategy regardless of the fitness function, we omit a stopping criterion and are only interested in the first point of time X_f at which the (1+1) EA has created an optimal bit string, i. e., an $x \in \{0,1\}^n$ such that $f(x)$ is maximal[1]. We denote the expected value of X_f as the *expected runtime* of the (1+1) EA. Besides, we often consider the so-called *success probability* $s_f(t)$, which indicates the probability that the (1+1) EA is able to find the global optimum of f within t, $t \in \mathbb{N}$, steps. If, for small values of t, $s_f(t)$ is not too small (e. g., constant or even bounded below by $1/poly(n)$), the expected runtime X_f

[1] W. l. o. g., we assume f to be maximized, since otherwise we may replace f by $-f$.

may well be exponential, although multistart strategies with an appropriate number of parallel instances of the (1+1) EA normally behave efficiently.

A common approach in analyzing the behavior of the (1+1) EA is studying its expected runtimes and its success probabilities on different fitness functions or, more generally, on different classes of fitness functions. Distinguishing fitness functions according to their degree seems to be one of the simplest and most natural ways of classifying them. Formally, we define the degree of a fitness function with respect to its unique representation as a polynomial.

Definition 2 *With a fitness function $f : \{0,1\}^n \to \mathbb{R}$ we identify its unique representation as a polynomial, i. e.,*

$$f(x_1, \ldots, x_n) = \sum_{I \subseteq \{1,\ldots,n\}} c_f(I) \cdot \prod_{i \in I} x_i$$

with coefficients $c_f(I) \in \mathbb{R}$.

Definition 3 *The degree of f is defined as*

$$\deg(f) := \max\{i \in \{0, \ldots, n\} \mid \exists I \textit{ with } |I| = i \textit{ and } c_f(I) \neq 0\} \ .$$

Functions of degree 0 are constant and thus are optimized trivially. The simplest and yet interesting class of fitness functions is the class of *linear functions*, which already has been subject to intense research by Droste, Jansen, and Wegener. In [DJW98b] they prove the upper and lower bound $\Theta(n \ln n)$ on the expected runtime of the (1+1) EA for all linear functions. Furthermore, they give hints on the optimality of the choice of the mutation probability $p := 1/n$ at least with respect to linear functions. On the other hand, they illustrate that already functions of degree 2 as well as unimodal functions (cf. Definition 8) cause the (1+1) EA to take an exponential expected number of steps. In this paper, we intend to examine the behavior of the (1+1) EA on quadratic functions in more detail.

In Section 2, we introduce some basic conventions and techniques which will be utilized throughout the paper. Especially, a simple method for showing upper bounds on expected runtimes is presented.

Section 3 deals with a specific subclass of quadratic functions, i. e., quadratic functions having only non-negative coefficients. They are in fact easy for the (1+1) EA in that the expected runtime is bounded by a polynomial.

As opposed to this result, in Section 4 we depict a simple quadratic function with negative coefficients which makes the (1+1) EA work for an exponential number of steps on average. Nonetheless, it does not constitute any problem to multistart variants of the (1+1) EA.

Thereafter, in Section 5, we undertake some studies on the structure of quadratic functions. A formal proof demonstrates that quadratic functions which are separable into linear functions defined on small domains cannot provoke exponential expected runtimes of the (1+1) EA.

Due to the NP-completeness of maximization of quadratic functions, we do not expect the (1+1) EA or its multistart variants to operate efficiently on quadratic functions in any case. This is dealt with in Section 6. We present a quadratic function which causes the (1+1) EA to work for an exponential time with a probability exponentially close to 1.

At last, Section 7 is devoted to another subclass of quadratic functions, i. e., *squares of linear functions*. We demonstrate that they are not difficult to optimize at least with respect to multistart variants of the (1+1) EA.

2 Basic Definitions and Techniques

We start off with some assumptions that we can make without loss of generality in order to simplify the representation of pseudo-boolean functions of degree 2, i. e., *quadratic* functions.

Definition 4 *A pseudo-boolean function* $f : \{0,1\}^n \to \mathbb{R}$*, given by* $f(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$ *with* $w_i, w_{ij} \in \mathbb{Z}$*, is called* quadratic.

Remark 1 Since $x^2 = x$ for $x \in \{0,1\}$, we drop w. l. o. g. any squares in this representation. The assumption $w_i, w_{ij} \in \mathbb{Z}$ does not constitute any restriction either, for real-valued coefficients may be approximated by rational ones, which in turn can be multiplied by their smallest common denominator. As additional constant terms have no influence on the behavior of the (1+1) EA, we assume w_0 to be zero in the following. Finally, we combine the terms $w_{ij} x_i x_j$ and $w_{ji} x_j x_i$ as commutativity holds.

Remark 2 From now on, we shall somewhat informally speak of *linear weights* when regarding the coefficients w_i in the linear terms $w_i x_i$, and of *quadratic weights* when regarding the coefficients w_{ij} in the quadratic terms.

In order to show upper bounds on the expected runtime of the (1+1) EA on pseudo-boolean fitness functions, we now introduce a simple proof technique which is helpful in several cases.

Definition 5 *Let* $f : \{0,1\}^n \to \mathbb{R}$ *be a pseudo-boolean function. Given two disjoint subsets* $A, B \subseteq \{0,1\}^n$ *with* $A \neq \emptyset \neq B$*, the relation* $A <_f B$ *holds iff* $f(a) < f(b)$ *for all* $a \in A$ *and* $b \in B$*.*

Definition 6 *Let* $A_1, \ldots, A_m$ *be a partition of* $\{0,1\}^n$*, i. e.,* $A_1 \cup \cdots \cup A_m = \{0,1\}^n$*.* $(A_1, \ldots, A_m, <_f)$ *is called an* f-based partition *iff* $A_i \neq \emptyset$ *for all* $i \in \{1, \ldots, m\}$*,* $A_1 <_f \cdots <_f A_m$*, and* A_m *merely consists of optimal vectors, i. e.,* $f(a) = \max\{f(x) \mid x \in \{0,1\}^n\}$ *for all* $a \in A_m$*.*

Definition 7 *Let* $(A_1, \ldots, A_m, <_f)$ *be an* f*-based partition. For* $a \in A_i$*,* $i < m$*, let* $s(a)$ *be the probability that a mutation of the (1+1) EA leads from* a *to an* $a' \in A_{i+1} \cup \cdots \cup A_m$*. Besides, let* s_i *denote the minimum of* $s(a)$ *for all* $a \in A_i$*.*

Lemma 1 *Given an f-based partition, let $p(A_i)$ be the probability that the initial bit string of the (1+1) EA belongs to A_i. Then the following upper bound on the expected runtime $E(X_f)$ on the corresponding function f holds:*

$$E(X_f) \le \sum_{i=1}^{m-1} p(A_i)(s_i^{-1} + \cdots + s_{m-1}^{-1}) \le s_1^{-1} + \cdots + s_{m-1}^{-1} \ .$$

Proof: Since A_i will be reached never again once the (1+1) EA has left A_i, s_i constitutes a lower bound on the probability of leaving A_i. Thus s_i^{-1} yields an upper bound on the expected number of steps until a string belonging to $A_{i+1} \cup \cdots \cup A_m$ is created. □

Of course, we do not expect the upper bounds gained by this technique to be tight in general. They particularly depend on a reasonable choice of the f-based partition. However, it is surprising how many tight or at least almost tight bounds can be obtained by this simple technique. For instance, we will make use of it in the next section to show a polynomial upper bound for a specific class of functions.

Concluding this section, we state a formal definition of unimodality, which is considered to be a property that makes a function easy to optimize. This claim is at least true with unimodal quadratic functions.

Definition 8 *A function $f : \{0,1\}^n \to \mathbb{R}$ is called unimodal iff it only has one local maximum, i. e.,*

$$\forall x \in \{0,1\}^n, x \text{ suboptimal}\colon \exists x' \in \{0,1\}^n \text{ with } H(x,x') = 1 \text{ and } f(x') > f(x) \ ,$$

where $H(x,x') := \sum_{i=1}^{n} |x_i - x'_i|$ is the Hamming distance *of the bit strings x and x'.*

3 Quadratic Functions without Negative Weights

As mentioned in the previous section, quadratic functions whose weights are positive have the property of being unimodal. (For the function to be unimodal, it is even sufficient that merely the linear weights are positive. An arbitrary number of quadratic weights may be zero.) In the following section, we consider quadratic functions with non-negative weights. By allowing arbitrary weights to be zero, the corresponding functions are not necessarily unimodal any more. However, as long as a global optimum has not yet been reached, there is a mutation flipping at most two bits which increases the function value. That guarantees an expected runtime of at most $O(n^4)$ steps.

Proposition 1 *Let $f : \{0,1\}^n \to \mathbb{R}$ be a quadratic pseudo-boolean function having only non-negative weights. Furthermore, let N denote the number of positive, i. e., non-zero, weights. Then it takes the (1+1) EA an expected time of at most $O(Nn^2)$ to optimize f.*

Proof: We sort all $N \le \binom{n}{2} + n$ weights differing from zero according to a decreasing order, represented by the sequence $w_1^* \ge \cdots \ge w_N^*$. By means of this sequence, we define N sets A_i, $i \in \{0, \ldots, N-1\}$, partitioning the domain $\{0,1\}^n$, according to

$$A_i := \{x \mid w_1^* + \cdots + w_i^* \le f(x) < w_1^* + \cdots + w_{i+1}^*\} \ .$$

Besides, set $A_N := \{(1, \ldots, 1)\}$. This is an f-based partition, for all w_i^* are positive. In order to apply Lemma 1, a lower bound on the corresponding probabilities s_i, $i \in \{0, \ldots, N-1\}$, remains to be proven. Since the w_i^* are decreasing, the following claim holds for all $a \in A_i$ and all $i \in \{0, \ldots, N-1\}$: There is a mutation flipping at most two bits simultaneously which creates an $a' \in \{0,1\}^n$ from a such that a' belongs to $A_{i+1} \cup \cdots \cup A_N$. First, we see that at least one of the weights $w_1^*, \ldots, w_{i+1}^*$ is not "activated", which means that the variable(s) appearing in the monomial which bears the weight we consider is/are zero. Otherwise, if all $w_1^*, \ldots, w_{i+1}^*$ were activated, the bit string a would pertain to $A_{i+1} \cup \cdots \cup A_N$. Secondly, we may activate a weight by flipping at most two bits from zero to one because f is a quadratic function, i. e., all monomials contain at most two variables. Since no weight is negative, thereby the value of f is increased by at least w_{i+1}^*. The probability of flipping two specific bits simultaneously is bounded below by $(1 - 1/n)^{n-2} \cdot n^{-2} \ge e^{-1}n^{-2} = \Omega(n^{-2})$ and the probability of flipping exactly one specific bit is even larger. Summing up the expected times $O(n^2)$ for all N sets, the proposition follows. □

Remark 3 We may view the index i of the set A_i to which a bit string a belongs as the minimal number of ones in a bit string which, on the imaginary linear function $g(x) = \sum_{j=1}^{N} w_j^* x_j$, yields a function value of at least $w_1^* + \cdots + w_i^*$. As m ones activate at most $\binom{m}{2} + m$ weights of the quadratic function f, the corresponding bit string must contain at least $\sqrt{i}$ ones. Hence n^2 increases of i are sufficient to reach the optimal bit string.

Remark 4 We do not expect to find a quadratic function with non-negative weights which takes the (1+1) EA an expected number of $\Omega(n^4)$ steps to optimize. However, we believe that a smaller lower bound would require much more effort in order to be shown. This is due to the following property of our technique from Lemma 1. By a partition of $\{0,1\}^n$, which respects the relation "$<_f$", it guarantees a kind of "progress" which will never be lost once having reached A_i. In contrast, using another kind of progress measure such as the number of ones in the current bit string might inevitably result in the analysis of a random walk. (Recall that the number of ones in a bit string normally does not determine the value of the function.) Although we suppose that an analysis regarding the number of ones in the current bit string will provide a better upper bound, we believe that taking into account the number of ones would lead to a proof being as difficult as the proof of the upper bound for linear functions by Droste, Jansen and Wegener (cf. [DJW98b]).

Of course, the probability of flipping k possibly selected bits is bounded below by $\Omega(n^{-k})$. Thus the following corollary for functions of degree k holds.

Corollary 1 *Let $f : \{0,1\}^n \to \mathbb{R}$ be a pseudo-boolean function of degree k which only has non-negative weights. Besides, let N denote the number of positive weights. Then it takes the (1+1) EA an expected time of at most $O(Nn^k)$ to optimize f.*

We have seen that quadratic functions without negative weights are optimized by the (1+1) EA quite efficiently. Clearly, a trivial algorithm always outputting $(1,\ldots,1)$ would even solve this task in constant time. However, our result holds for all quadratic functions obtained by replacing some x_i by $(1-x_i)$ (implying that each $a \in \{0,1\}^n$ can be optimal). Moreover, we are considering the scenario of black box optimization here implying that the specific nature of the fitness function is not known.

Optimization of quadratic pseudo-boolean functions becomes an NP-hard problem as soon as general negative weights are allowed. The NP-completeness of the corresponding decision problem can be demonstrated by a simple reduction from the NP-complete MAX-2-SAT problem (cf. [GJ79]). Thus we are convinced that there are quadratic functions which cause the simple (1+1) EA to fail, i. e., to take an exponential expected runtime. In the next sections, we will deal with some functions which, in this respect, prove to be difficult.

4 A Simple Quadratic Function with Expected Runtime $\Omega(n^n)$

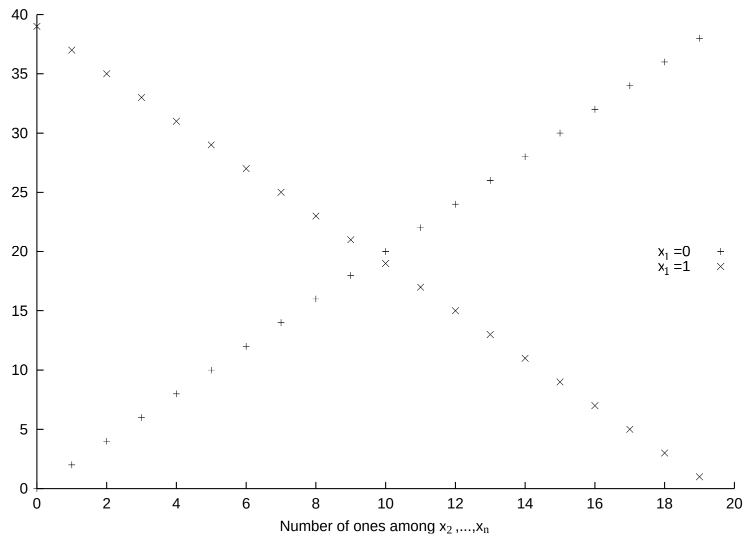

Figure 1: Plot of CROSSING for $n = 20$

By constructing a quadratic function which is essentially a combination of two linear functions that are, in a sense, "opposed" to each other, we are able to deceive the (1+1) EA and to provoke an exponential expected runtime.

Definition 9 *The function* CROSSING : $\{0,1\}^n \to \mathbb{R}$ *is defined by*

$$\text{Crossing}(x) = (2n-1)x_1 + \sum_{i=2}^{n} 2x_i + \sum_{i=2}^{n}(-4)x_1 x_i \enspace .$$

Subject to the assignment of x_1, CROSSING resolves into two different linear functions on $x_2, \ldots, x_n$. If $x_1 = 0$, we obtain

$$f_0(x_2, \ldots, x_n) = 2 \cdot (x_2 + \cdots + x_n) \enspace ,$$

i. e., $2 \cdot \text{OneMax}(x_2, \ldots, x_n)$. (ONEMAX was introduced by Mühlenbein [Müh92]. For an extensive examination cf. [Rud97].) If $x_1 = 1$, we have

$$f_1(x_2, \ldots, x_n) = (2n-1) - 2 \cdot (x_2 + \cdots + x_n) \enspace ,$$

corresponding with $(2n-1) - 2 \cdot \text{OneMax}(x_2, \ldots, x_n)$. Obviously, CROSSING takes its maximum value $2n-1$ on $(1, 0, \ldots, 0)$, but owns a local maximum in $(0, 1, \ldots, 1)$ with the value $2n-2$ (see also Figure 1). Already the probability 2^{-n} of starting in $(0, 1, \ldots, 1)$ suffices to show an exponential runtime of at least $(n/2)^n$. In order to cross the "Hamming cliff" at $(0, 1, \ldots, 1)$, it is necessary to flip all n bits at once, which has probability n^{-n}. The expected waiting time for this event equals n^n. Moreover, we are able to prove that on the one hand, exponential runtimes occur with high probability, but on the other hand, polynomial runtimes occur at least as often as exponential ones.

Proposition 2 *With a probability of at least* $1/4 - \epsilon$, $\epsilon > 0$ *arbitrarily small, the (1+1) EA requires* $\Omega(n^n)$ *steps to optimize* CROSSING. *But the probability of reaching the global optimum within* $O(n \log n)$ *steps is bounded below by* $1/4 - \epsilon'$, $\epsilon' > 0$ *arbitrarily small.*

Proof: With probability 1/2, the initial bit string x^* has its bit x_1^* set to 1. Independently thereof, the bits $x_2^*, \ldots, x_n^*$ contain at least $(n-1)/2 + (n-1)^{1/4}$ ones with a probability of at least $1/2 - o(1)$. This follows from the fact that the number of ones is binomially distributed according to $n-1$ and $1/2$. By Stirling's formula, we see that x^* contains exactly k ones among $x_2^*, \ldots, x_n^*$ with a probability bounded above by $O(n^{-1/2})$ for all $k \in \{0, \ldots, n-1\}$. Therefore, the probability of initializing x^* with at least $(n-1)/2$ and less than $(n-1)/2 + (n-1)^{1/4}$ ones in the positions $x_2^*, \ldots, x_n^*$ is bounded above by $O(n^{-1/4})$.

As different positions in x^* are initialized independently, both of the events described above occur with probability $1/4 - o(1)$. Subsequently, the (1+1) EA behaves like on the linear function ONEMAX and takes on average $cn \ln n$ steps to reach the global optimum unless a mutation flipping at least $k := 2(n-1)^{1/4}$ bits simultaneously is executed. The latter has a probability of at most $\binom{n}{k}(1/n)^k \leq (n^k/k!)n^{-k} = 1/k!$, which yields at most $2^{-\Omega(n^{1/4} \log n)}$ according to Stirling's formula. With a probability bounded below by $1 - c'cn \ln n 2^{-\Omega(n^{1/4} \log n)} = 1 - o(1)$ it never happens in $c'cn \ln n$ steps that k bits flip simultaneously. Applying Markov's inequality, we upper bound the probability of not reaching the global maximum of ONEMAX during $c'cn \ln n$ steps.

If we choose c' large enough, the probabilities $1/c' + o(1)$ of these "errors" are at most ϵ', thus proving the second part of the proposition.

The first claim is proven by dual arguments. With a probability of at least $1/4-\epsilon$, the (1+1) EA optimizes towards $(0, 1, \ldots, 1)$ and gets caught in the local optimum. Thereupon it needs at least $\delta n^n - \delta$ steps to flip all bits with a probability of at least $(1-n^{-n})^{\delta(n^n-1)} \geq e^{-\delta}$, which is greater than $1-\epsilon$ if we choose δ small enough. □

Remark 5 We even believe that, in Proposition 2, the constant 1/4 may be replaced with 1/2. We content ourselves with 1/4 in order to keep the proof simple.

Proposition 2 indicates that optimizing Crossing is not really a difficult task if utilizing multistart strategies. By running k independent instances of the (1+1) EA, on average approximately at least $k/4$ instances succeed in finding the global maximum within $\Theta(n \log n)$ steps. The probability of all instances' failing is bounded above by $(3/4+\epsilon)^k$ and hence even gets exponentially small in k.

In Section 6, we present a quadratic function which is "especially difficult" in so far as the (1+1) EA has to work unsuccessfully for $2^{\Omega(n \ln n)}$ steps with a probability exponentially close to one. With reference to that function, the success probability $s_f(t)$ is still exponentially small after an exponential time.

Beforehand, we depict some necessary conditions for quadratic functions which make the (1+1) EA need an exponential number of steps on average.

5 On the Influence of Separability on the Expected Runtime

Taking a closer look at the function Crossing from Definition 9, we realize that its $n-1$ quadratic weights "connect" each of the variables $x_2, \ldots, x_n$ with the variable x_1. It seems impossible to separate Crossing into distinct linear functions operating on disjoint variable sets. In contrast, one may conjecture that quadratic functions whose variable set is separable in such a way are optimized by the (1+1) EA more efficiently, particulary since the (1+1) EA finds the optimum of linear (bitwise separable) functions in polynomial time. We will prove this rigorously by means of an equivalence relation, which reflects the separability of the underlying function.

Definition 10 *With respect to a quadratic fitness function* $f : \{0,1\}^n \to \mathbb{R}$ *with* $f(x) = \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n w_{ij} x_i x_j$*, two indices* $i, j \in \{1, \ldots, n\}$ *are in* quadratic relation *to each other, denoted by* $i \equiv_f j$*, iff* $i = j$*, or there is a sequence of pairwise different indices* $i_1, \ldots, i_k$ *with* $i_1 = i$ *and* $i_k = j$ *such that* $w_{i_l, i_{l+1}} \neq 0$ *or* $w_{i_{l+1}, i_l} \neq 0$ *holds for all* $l \in \{1, \ldots, k-1\}$.

As usual, we use the notation $[i] := \{j \mid j \equiv_f i\}$ to denote the equivalence class of i.

Definition 11 *Let* $f(x) = \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n w_{ij} x_i x_j$ *be a quadratic function.*

For an index $i \in \{1, \ldots, n\}$, we define the subfunction induced by $[i]$ as

$$f_{[i]}(\{x_j \mid j \in [i]\}) := \sum_{j \in [i]} w_j x_j + \sum_{\substack{j,k \in [i] \\ j<k}} w_{jk} x_j x_k \quad .$$

As intended, the relation "$\equiv_f$" reflects the separability of a function.

Lemma 2 *Let $f(x) = \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n w_{ij} x_i x_j$ be a quadratic function. Besides, let $R \subseteq \{1, \ldots, n\}$ be a minimal complete system of representatives for the relation $\equiv_f$. Then f can be separated into $\#R$ functions according to*

$$f(x_1, \ldots, x_n) = \sum_{i \in R} f_{[i]} \left(\{x_j | j \in [i]\}\right) \quad .$$

Proof: The identity of the corresponding polynomial representations

$$S_1 := \sum_{i=1}^n w_i x_i + \sum_{j=1}^n \sum_{k=j+1}^n w_{jk} x_j x_k$$
$$S_2 := \sum_{i \in R} \sum_{j \in [i]} w_j x_j + \sum_{i \in R} \sum_{\substack{j,k \in [i] \\ j<k}} w_{jk} x_j x_k$$

can be shown easily by verifying that each monomial from S_1 that has a non-zero weight is contained in S_2 exactly once, and vice versa. This follows from the fact that R is a minimal complete system of representatives for the equivalence relation $\equiv_f$. $\square$

Now we are able to state an upper bound on the expected runtime of the (1+1) EA which solely depends on the cardinality of the largest equivalence class of $\equiv_f$.

Proposition 3 *Let $f(x) = \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n w_{ij} x_i x_j$ be a quadratic function. Furthermore, let $m := \max\{\#[i] \mid i \in \{1, \ldots, n\}\}$ denote the maximal cardinality of all equivalence classes with respect to $\equiv_f$. Then the (1+1) EA optimizes f in an expected number of $O(2^m n^{m+1})$ steps.*

Proof: We intend to apply Lemma 1 and thus need a sensible partition of the domain $\{0,1\}^n$. As f decomposes into at most $n - m + 1$ subfunctions (see Lemma 2), which take at most 2^m different values each, we define this partition by means of differences between "consecutive" function values of these subfunctions.

Formally, first we set up a minimal complete system of representatives, denoted by R, for the corresponding equivalence relation $\equiv_f$. We define $l_{[i]} := \min\left\{f_{[i]}(x) \mid x \in \{0,1\}^{\#[i]}\right\}$ as the minimal value of the subfunction $f_{[i]}$, $i \in R$, and accordingly $h_{[i]}$ as the maximal value of a subfunction. For each subfunction $f_{[i]}$, $i \in R$, we define the non-negative sequence $d_{[i]}^j := v_{[i]}^{j+1} - v_{[i]}^j$, $j \in \left\{0, \ldots, v_{[i]} - 1\right\}$ representing the differences between "consecutive" values $v_{[i]}^j$ and $v_{[i]}^{j+1}$ of the corresponding subfunction. Here "consecutive" means that there is no value of $f_{[i]}$ which is strictly greater than $v_{[i]}^j$ and strictly less than $v_{[i]}^{j+1}$; besides $v_{[i]}$ denotes the number of different values of $f_{[i]}$. Clearly, $\sum_{j=0}^{v_{[i]}-1} d_{[i]}^j = h_{[i]} - l_{[i]}$ holds, as the $v_{[i]}^j$ telescope. After that, we sort the sequence obtained by enumerating $d_{[i]}^j$ for all $i \in R$ and for all $j \in \{0, \ldots, v_{[i]} - 1\}$ according to a monotonously decreasing sequence, denoted by v_i^*, with $i \in \{1, \ldots, v\}$, $v := \sum_{i \in R} v_{[i]}$. Finally, we set up v sets which are compliant with Lemma 1 as

$$A_i := \left\{x \mid f_{\min} + v_1^* + \cdots + v_i^* \leq f(x) < f_{\min} + v_1^* + \cdots + v_{i+1}^*\right\} \quad ,$$

where $f_{\min} := \min\{f(x) \mid x \in \{0,1\}^n\}$ and i ranges from 0 to $v - 1$.

Due to the separability of f, we have $f_{\min} = \sum_{i \in R} l_{[i]}$ and $f_{\max} = \sum_{i \in R} h_{[i]}$. ($f_{\max}$ is to denote the maximal value of f.) Moreover, it follows by the definition of v_i^* that

$$\sum_{i=1}^{v} v_i^* + f_{\min} = \sum_{i=1}^{\#R} \sum_{j=0}^{v_{[i]}-1} d_{[i]}^j + f_{\min} = \sum_{i=1}^{\#R} (h_{[i]} - l_{[i]}) + f_{\min} = f_{\max} \quad .$$

A lower bound on the probability of leaving A_k, $k \in \{0, \ldots, v-1\}$, remains to be proven. If the current bit string of the (1+1) EA pertains to A_k, we claim that there is a subfunction whose value can be increased by at least v_{k+1}^*. Otherwise, if no subfunction could be increased by v_{k+1}^*, this would imply $f_{[i]}(\{x_j | j \in [i]\}) > h_{[i]} - v_{k+1}^*$ for all $i \in R$. By summing up over all equivalence classes we would obtain $f(x) \geq f_{\max} - \sum_{j=k+2}^{v} v_j^*$, for only the variables $v_{k+2}^*, \ldots, v_v^*$ can be less than v_{k+1}^*. (Remember that the v_i^* comprise all differences of consecutive values of subfunctions.) Since $f_{\max} - \sum_{j=k+2}^{v} v_j^* = f_{\min} + \sum_{j=1}^{k+1} v_j^*$, this would however imply $a \in A_{k+1}$. Hence, there is at least one subfunction whose value can be increased by at least v_{k+1}^*. A sufficient condition is a mutation of at most m selected bits, the probability of which is bounded below by $(1 - 1/n)^{n-m} n^{-m} \geq e^{-1} n^{-m} = \Omega(n^{-m})$. As the number of linear subfunctions is at most n, each of which can take at most 2^m different values, $v = O(n2^m)$ holds, i. e., there are at most $O(n2^m)$ different sets A_k. This implies the bound $O(n2^m n^m) = O(2^m n^{m+1})$ on the expected runtime. $\square$

Remark 6 Proposition 3 is even valid for arbitrary pseudo-boolean functions which decompose into linear functions defined on variable sets comprising at most m variables.

Corollary 2 *Let f be given as in Proposition 3 and be m defined accordingly. If $m = O(1)$, the (1+1) EA optimizes f in an expected number of at most $O(n^{m+1})$ steps.*

Now we can draw some conclusions concerning "almost linear" functions, i. e., quadratic pseudo-boolean functions whose number of quadratic weights is bounded above by a constant. If only k quadratic weights, $k \in \mathbb{N}$ constant, are allowed, we can set at most $k+1$ indices in relation with each other. Thereby we get the upper bound $O(n^{k+2})$ for quadratic functions possessing only k quadratic weights that differ from zero. Conversely, we can easily depict in how far k quadratic weights suffice to evoke expected runtimes of $\Omega(n^{k+1})$.

Definition 12 *The function* $\text{CROSSING}_k : \{0,1\}^n \to \mathbb{R}$*,* $k \in \mathbb{N}$*, is defined by*

$$\text{CROSSING}(x) = (2n^2 - n)x_1 + \sum_{i=2}^{k+1} 2nx_i + \sum_{i=2}^{k+1}(-4n)x_1x_i + \sum_{i=k+2}^{n} x_i \ .$$

Restricted to the bits $x_1, \dots, x_{k+1}$, this function resembles CROSSING (albeit multiplied by n). On the other hand, any subfunction restricted to the variables $x_{k+2}, \dots, x_n$ behaves like ONEMAX apart from an additional constant. The comparatively large weights at the first $k+1$ positions simplify the following proof.

Proposition 4 *On average at least* $\Omega(n^{k+1})$ *steps are necessary until the (1+1) EA reaches the global optimum of* CROSSING_k*.*

Proof: It is easy to verify that the assignment of the first $k+1$ bits remains fixed at $x_1 = 0$, $x_2 = \dots = x_{k+1} = 1$ irrespective of the assignment of $x_{k+2}, \dots, x_n$ unless all of the first $k+1$ bits flip simultaneously. (This is due to the large weights which result in any change of $x_1, \dots, x_{k+1}$ making the value of CROSSING_k decrease by at least $n - (n-k-1) = k+1$, except for the mutation that flips all $k+1$ bits in one step.) Since the probability of initializing $x_1, \dots, x_{k+1}$ corresponding to $(0, 1, \dots, 1)$ equals $2^{-k-1} = \Omega(1)$ and the average waiting time until $k+1$ selected bits flip at once is bounded by $\Omega(n^{k+1})$, this contributes at least $\Omega(n^{k+1})$ to the expected runtime. $\Box$

In this section we have demonstrated that quadratic functions that are decomposable into linear functions are optimized in an expected time which is bounded by $O(2^m n^{m+1})$, where m denotes the maximum number of variables on which a linear subfunction depends. If m is a constant, the bound simplifies to $O(n^{m+1})$. This has implications on quadratic functions whose number of quadratic weights is bounded above by a constant k; in that case the expected runtime is bounded by $O(n^{k+2})$.

6 A Quadratic Function Causing Exponential Runtimes with Overwhelming Probability

As announced in Section 2, due to NP-completeness one may not hope to find a deterministic algorithm which is able to optimize all quadratic functions within polynomial time. Unless RP=NP, there cannot be a randomized algorithm solving this problem in polynomial expected time either. Therefore we are convinced that there is at least one quadratic function which even makes multistart strategies fail. A necessary condition is that the probability of finding the global optimum of this "especially difficult" function in polynomial time cannot by bounded below by $\Omega(n^{-k})$ for all $k \in \mathbb{N}$. Then a polynomial number of instances of the (1+1) EA is not sufficent to solve the problem in polynomial expected overall time implying that a superpolynomial number of instances and thus superpolynomial expected overall time would be needed.

The "especially difficult" function which we present in this section was constructed by means of an application of a polynomial reduction originally discoverd by Rosenberg [Ros75]. Thereby, he reduces the (decision) problem of maximizing an arbitrary pseudo-boolean function to the one of maximizing a quadratic pseudo-boolean function. We use Rosenberg's reduction to transform the function

$$\textsc{Trap}_n(x) = -\sum_{i=1}^{n} x_i + (n+1)\prod_{i=1}^{n} x_i \ ,$$

at first considered by Ackley [Ack87] and later on explored in more detail by Droste, Jansen, and Wegener [DJW98a][2], to the quadratic function

$$\begin{aligned}\textsc{Trap}^*_n(x) = -\sum_{i=1}^{n} x_i + (n+1)x_i x_{2n-2} \\ - (n+1)\sum_{i=1}^{2n-2} (x_{n-i}x_{n+i-1} + x_{n+i}(3 - 2x_{n-i} - 2x_{n+i-1})) \ ,\end{aligned}$$

which is a function defined on $2n-2$ variables. First of all, we consider the so-called "penalty terms" $x_{n-i}x_{n+i-1} + x_{n+i}(3 - 2x_{n-i} - 2x_{n+i-1})$. By checking all 8 assignments to x_{n-i}, x_{n+i-1} and x_{n+i} we conclude that the penalty term is zero iff $x_{n-i}x_{n+i-1} = x_{n+i}$ holds, and becomes either 1 or 3 otherwise. As it is the case with the original Trap function, the only optimal bit string is $\vec{1} = (1,\dots,1)$ yielding a function value of 1. To verify this, we realize that positive values can only be obtained by setting $x_1 = x_{2n-2} = 1$ and making sure that no penalty term takes a value differing from zero. The latter can only be accomplished if $x_{n-1}x_n = x_{n+1}$, $x_{n-2}x_{n+1} = x_{n+2}$, $\dots$, $x_3x_{2n-4} = x_{2n-3}$ and $x_2x_{2n-3} = x_{2n-2}$. As $x_{2n-2} = 1$, this implies $x_2 = x_{2n-3} = 1$, then $x_3 = x_{2n-4} = 1$ and so forth. Hence $\vec{1}$ is the only global optimum.

In terms of suboptimal $x \in \{0,1\}^n$, the value of $\textsc{Trap}^*_n$ can always be represented by $-j - k(n+1)$, where $k \in \{0,\dots,3(n-2)\}$ and $j := \sum_{i=1}^{n} x_i$, i. e., j denotes the number of ones in the first n positions of the string x. This is due to the above-mentioned properties of the $n-2$ penalty terms.

Bearing this in mind, we are now able to prove that $\textsc{Trap}^*_n$ is an especially difficult function.

Proposition 5 *With a probability of* $1 - 2^{-\Omega(n)}$, *the (1+1) EA requires at least* $2^{\Omega(n \log n)}$ *steps to optimize* $\textsc{Trap}^*_n$.

Proof: Concerning bit strings $x \in \{0,1\}^n$, we distinguish their "left" parts $x_1,\dots,x_n$ and their "right" parts $x_{n+1},\dots,x_{2n-2}$, corresponding to the original n variables and the additional variables introduced by the reduction. It follows by Chernoff's inequality (see [MR95]) that the initial bit string of the (1+1) EA contains at least $(2/5)n$ ones in its left part with a probability of at least $1 - 2^{-\Omega(n)}$. For the right part, we apply the "Principle of Deferred Decisions" (cf. [MR95]) pretending that all n bits of x are initialized after each other. (Due to independency of the bits, this constitutes a permissible assumption.) Regarded like that, x_{n+i} "hits" the fixed

[2] In that paper they prove that the (1+1) EA requires $\Omega(n^n)$ steps to optimize $\textsc{Trap}_n$ with a probability exponentially close to 1.

value of $x_{n-i}x_{n+i-1}$ with a probability of 1/2 for all $i \in \{1, \dots, n-2\}$ implying that $(n-2)/2$ penalty terms are zero on average. Again Chernoff's bound can be applied such that, with a probability of $1 - 2^{-\Omega(n)}$, at least $(2/5)(n-2)$ penalty terms are zero after initialization. In the following, we assume both events considered above to have occured. As long as the (1+1) EA has not yet reached the optimum, the value of TRAP^*_n is given by $-j - k(n+1)$, where $k \leq (9/5)(n-2)$, and j denotes the number of ones in the left part. A necessary condition for the value of j to decrease (as a result of a successful mutation) is a decrease of k due to the same mutation. Obviously, k can decrease at most $(9/5)(n-2)$ times. We want to estimate the probability that during at most $(9/5)(n-2)$ mutations decreasing k many zeros in the left part flip to one. Pessimistically, we assume that the left part contains $(3/5)n$ ones after initialization and that none of these ones ever flips to zero. For the remaining $(2/5)n$ zeros in the first part the overall number of bits flipped during at most $(9/5)(n-2)$ mutations has an expected value of at most $(2/5)n \cdot (9/5)(n-2)/(2n-2) \leq (9/25)n$, since each bit is flipped with probability $1/(2n-2)$ per step. As $(2/5 - 9/25)n = (1/25)n$, we conclude that an expected value of at least $(1/25)n$ zeros remains in the left part even if k has fallen to its minimal value. Bounding this by another application of Chernoff's bound, we obtain that at least $(1/30)n$ zeros remain with a probability of at least $1 - 2^{-\Omega(n)}$ even if k has decreased to zero. Afterwards, the number of zeros in the first part can only grow unless all zeros flip to one simultaneously. The latter has a probability of at most $(1/(2n-2))^{n/30} = 2^{-\Omega(n \log n)}$. By utilizing an estimation like in the proof of Proposition 2, we can upper bound the probability of such a success during $2^{\epsilon n \log n}$ steps by $2^{-\Omega(n)}$ if ϵ is small enough. This completes the proof. $\square$

Remark 7 The proof of Proposition 5 includes several unrealistically optimistic assumptions about the number of bits which have to flip simultaneously in order to reach the global optimum of TRAP^*_n. In nearly all cases we expect the number of ones in the current bit string of the (1+1) EA to decrease towards zero such that a mutation flipping all bits simultaneously is required. We presume that even an expected waiting time $\Omega(2^{n \log n})$ (i. e., the constant in the exponent equals 1) is necessary with a probability exponentially close to 1, as it is the case with the original TRAP function.

Supporting the common assumptions of Complexity Theory, we have proven that multistart variants of the (1+1) EA fail on certain quadratic functions. Even if employing a polynomial number of parallel instances of the (1+1) EA, the expected number of instances that are successful within polynomial time converges exponentially towards zero.

7 Squares of Linear Functions

Naturally, quadratic pseudo-boolean functions arise by taking linear functions to the power of two.

Definition 13 *Let $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^n w_i x_i$ be a linear function. By $f^2(x) := \left(w_0 + \sum_{i=1}^n w_i x_i\right)^2$ we denote the* square *of the linear function f.*

W. l. o. g., we now may assume all weights of the linear function to be positive integers and to be sorted, i. e., $w_1 \geq \cdots \geq w_n > 0$, $w_i \in \mathbb{N}$. (If $w_i < 0$, we replace x_i by $1 - x_i$, which has no influence on the behavior of the (1+1) EA.) However, we may not rely on $w_0 = 0$. Imagine that $f(x) = w_0 + \sum_{i=1}^{n} w_i x_i$ is a linear function with $w_0 \geq 0$. Then the (1+1) EA behaves on f^2 like on f, for $x \mapsto x^2$ is a strictly increasing mapping on $\mathbb{R}_0^+$. A similar situation arises if $w_0 \leq -\sum_{i=1}^{n} w_i$. In that case $f(x) \leq 0$ holds for all $x \in \{0,1\}^n$ such that the (1+1) EA behaves on f^2 like on $-f$ due to $x \mapsto x^2$ being a strictly decreasing mapping on R_0^-. If a linear function takes both positive and negative values, its square gets interesting properties and does not appear "linear" to the (1+1) EA any more.

Definition 14 *Let $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^{n} w_i x_i$ be a linear function. With respect to f, we partition $\{0,1\}^n = N(f) \cup P(f)$ according to*

$$N(f) := \{x \in \{0,1\}^n \mid f(x) < 0\} \text{ and } P(f) := \{x \in \{0,1\}^n \mid f(x) \geq 0\} \ .$$

We have just seen that the square of a linear function can only be interesting if $N(f) \neq \emptyset \neq P(f)$. Restricted to either $P(f)$ or $N(f)$, both the linear function and its square are unimodal with the maximum lying in $\vec{0} = (0, \ldots, 0)$ and $\vec{1} = (1, \ldots, 1)$, respectively. Thus f^2 may have two local maxima, namely one in $\vec{0}$ and one in $\vec{1}$. (The function f^2 does not necessarily possess two local maxima since "Hamming neighbors" of $\vec{0}$ may belong to $P(f)$, and vice versa for $\vec{1}$.) Again we may exchange the meaning of x_i and $1 - x_i$ for $i \in \{1, \ldots, n\}$ if needed in order to ensure that w. l. o. g., $\vec{1}$ is the global maximum of f^2.

We can easily construct linear functions, where w_0 is "close" to $-w/2$, $w := \sum_{i=1}^{n} w_i$, such that in terms of f^2 merely the global maximum in $\vec{1}$ yields a better value than the local maximum in $\vec{0}$. Consider, e. g., the function $f(x) = \text{OneMax}(x) - n/2 + 1/3$. For its square f^2 (introduced in [DJW98a] and called Distance there), we have $f^2(\vec{0}) = (n/2 - 1/3)^2$ and $f^2(\vec{1}) = (n/2 + 1/3)^2$ as well as $f^2(x) \leq (n/2 - 2/3)^2$ for all $x \in \{0,1\}^n \setminus \{\vec{0}, \vec{1}\}$. Therefore, the (1+1) EA can get stuck in the local maximum in $\vec{0}$, which results in an average waiting time n^n until a mutation flipping all bits occurs. Yet it is probable that the (1+1) EA only creates bit strings from $P(f)$ where it behaves like on OneMax and is able to reach the string $\vec{1}$ within $O(n \log n)$ steps. The situation is similar to the one in Proposition 2; we assume that the (1+1) EA is able to reach the global optimum of f^2 with a probability of about 1/2 but likewise has to wait $\Omega(n^n)$ steps with a probability of approximately 1/2.

In the following, we want to prove that the (1+1) EA is able quickly to encounter a local maximum on the square of an arbitrary linear function. In addition, we intend to demonstrate that it finds the global maximum $\vec{1}$ within polynomial time with a probability bounded below by a constant, irrespective of the weights of the underlying linear function.

Lemma 3 *On the square f^2 of a linear function f, the expected time until the (1+1) EA reaches either the string $\vec{0}$ or the string $\vec{1}$ is $O(n^2)$.*

Proof: Anew we make use of Lemma 1. However, we define two partitions according to the linear functions f and $-f$, namely an f-based partition

$$A_j = \{x \mid w_0 + w_1 + \cdots + w_j \leq f(x) < w_0 + w_1 + \cdots + w_{j+1}\}$$

and a $-f$-based partition

$$B_j = \{x \mid -w_0 - w_{n-j+1} - \cdots - w_n \geq -f(x) > -w_0 - w_{n-j} - \cdots - w_n\}$$

with $j \in \{0, \ldots, n-1\}$. Moreover, set $A_n = \vec{1}$ and $B_n = \vec{0}$. For all $x \in A_j$, all $x' \in P(f)$ with $f(x') \geq f(x)$ have at least j ones. Analogously, for $x \in B_j$, all $x' \in N(f)$ with $-f(x') \geq -f(x)$ contain at least j zeros. If $a \in A_j \cap P(f)$, all strings $x' \in P(f)$ which the (1+1) EA is able to reach belong to $A_{j+1} \cup \cdots \cup A_n$; an analogous statement holds for $B_j \cap N(f)$. Obviously, all $a \in A_j \cap P(f)$ contain at least one zero amongst the first $j+1$ positions; thus there is a mutation flipping a possibly specific bit which leads from a to $a' \in (A_{j+1} \cup \cdots \cup A_n) \cap P(f)$. By analogy, we obtain that an arbitrary $x \in B_j \cap N(f)$ can be mutated to $a' \in (B_{j+1} \cup \cdots \cup B_n) \cap N(f)$ by a mutation flipping a one bit to zero.

During the algorithm, we evaluate bit strings by means of triples $(i, j, a) \in \{0, \ldots, n\} \times \{0, \ldots, n\} \times \{0, 1\}$. If the initial bit string of the (1+1) EA belongs to $A_j \cap P(f)$, we assign the value $(j, 0, 0)$ to it; if it comes from $B_j \cap P(f)$, we assign $(0, j, 1)$. In general, the value (i, j, a) assigned to a bit string $x \in \{0,1\}^n$ indicates that x belongs to $A_i \cap P(f)$ or to $B_j \cap N(f)$, which is dependent on a. If $a = 0$, the bit string x is in $A_i \cap P(f)$, while the last string from $N(f)$ was belonging to $B_j \cap N(f)$. (In case that there never was a string from $N(f)$, we set $j = 0$). If $a = 1$, the roles of $A_i \cap P(f)$ and $B_j \cap N(f)$ are exchanged.

The first two components of an assignment (i, j, a) can never decrease since A_j and B_j are f-based and $-f$-based partitions, respectively. As soon as a component has increased to n, the (1+1) EA has created $\vec{0}$ or $\vec{1}$. As that is the case after at most $2n - 1$ increases of i or j, $O(n)$ mutations which flip a (selected) bit in order to increase the value of the current component suffice. It is already known that the expected waiting time for such a mutation is $O(n)$. Putting this together yields the upper bound $O(n^2)$. □

Up to now, we only have an upper bound on the time until reaching one of the local optima. In order to prove a lower bound on the probability of reaching the global optimum $\vec{1}$ within polynomial time, some prerequisites are necessary.

We want to find out some more details on the distribution of the initial value of the function to be optimized. That is obviously influenced by the random assignment of the initial bit string.

Definition 15 *By x^* we denote a random variable obeying the distribution of the initital bit string of the (1+1) EA. The variable x^* is the n-ary cartesian product of independent random variables X_i, $i \in \{1, \ldots, n\}$, where X_i is 0 or 1 with probability $1/2$ each.*

With respect to a linear function $f(x) = w_0 + \sum_{i=1}^{n} w_i x_i$, henceforth we use the abbreviation $w := \sum_{i=1}^{n} w_i$.

Lemma 4 *Let $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^n w_i x_i$ be a linear function. For the random variable $V := f(x^*)$ yielding the random initial value of f we obtain $E(V) = w_0 + w/2$.*

Proof: The claim follows immediately from the linearity of expectation. □

Lemma 5 *Let $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^n w_i x_i$ be a linear function. For the random variable $V := f(x^*)$ yielding the random initial value of f the following claim regarding its distribution holds:* $\text{Prob}(V \geq w_0 + w/2) \geq 1/2$.

Proof: W. l. o. g., we assume $w_0 = 0$. By enumerating all 2^{n-1} pairs $(x, \bar{x})$, consisting of $x \in \{0,1\}^n$ and its bitwise complement, we obtain

$$\sum_{i=1}^n w_i x_i < w/2 \Rightarrow \sum_{i=1}^n w_i(1 - x_i) > w - w/2 = w/2$$

for each pair. In order words, at least one half of all bit strings $x \in \{0,1\}^n$ yield at least $w/2$. □

If a bit string $x \in \{0,1\}^n$ consists of k ones, its complement has $n - k$ ones. Thus the following corollary holds.

Corollary 3 *Let $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^n w_i x_i$ be a linear function. Besides, let q_k denote the probability that a bit string $x \in \{0,1\}^n$ which has exactly k ones yields an f-value bounded below by $w_0 + w/2$. Then $q_k + q_{n-k} \geq 1$ holds for all $k \in \{0, \ldots, n\}$.*

We mentioned above that, regarding the square of a linear function $f(x) = w_0 + \sum_{i=1}^n w_i x_i$, we may w. l. o. g. assume that $w_0 \geq -w/2$. In connection with Lemma 5 we conclude that, initially, the (1+1) EA creates a bit string from $P(f)$ with a probability of at least 1/2. For the proof of the main proposition, we even need more. The following lemma states a lower bound on the probability that the random variable $f(x^*)$ deviates from its expected value $w_0 + w/2$ towards "more positive" bit strings.

Lemma 6 *Let $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^n w_i x_i$ be a linear function. Besides, let $w_d^* := \sum_{i=0}^{d-1} w_{n-i}$ denote the sum of the d smallest weights. For the random variable $V := f(x^*)$ yielding the random initial value of f the relationship* $\text{Prob}(V \geq w_0 + w/2 + w_d^*) \geq 1/2 - 2d/n^{1/2}$ *holds.*

Proof: If all weights are identical, the claim follows immediately from properties of the binomial distribution. Similar to the proof of Proposition 2 we conclude that there are at most $(d/n^{1/2}) \cdot 2^n$ bit strings containing at least $n/2$ and less than $n/2 + d$ ones. In general, this argument can easily be used to show a probability of least $1/4 - d/n^{1/2}$ for the event mentioned above. Obviously, d ones yield a function value of at least w_d^*. With a probability of at least $1/2 - d/n^{1/2}$, the initial bit string has at least $n/2 + d$ ones. With a probability bounded below by 1/2, its "remaining" $n/2$ ones together have a weight which is at least as large as the weight of the remaining at most $n/2$ zeros. To show the lower bound $1/2 - 2d/n^{1/2}$, we have to be more careful.

Again we may assume w_0 to be zero. At first, consider an urn containing n balls with the corresponding weights $w_1, \ldots, w_n$. For $k \geq d$, let q_k^* be the probability that after having drawn k balls we attain an overall weight of at least $w/2 + w_d^*$. The probability that after having drawn $k - d$ balls the overall weight amounts to at least $w/2$ was already denoted by q_{k-d}. As d balls weigh at least w_d^*, we obtain the relationship $q_k^* \geq q_{k-d}$. Since by assumption $w_0 = 0$, we have to consider the event $V \geq w/2 + w_d^*$ and we estimate the number of vectors where the event holds according to

$$
\begin{aligned}
\sum_{k=0}^{n} q_k^* \binom{n}{k} &\geq \sum_{k=d}^{n} q_{k-d} \binom{n}{k} \geq \sum_{k=2d}^{n/2} q_{k-d} \binom{n}{k} + \sum_{k=n/2+2d}^{n} q_{k-d} \binom{n}{k} \\
&\geq \sum_{k=2d}^{n/2} q_{k-d} \binom{n}{k-2d} + \sum_{k=n/2}^{n-2d} q_{k+d} \binom{n}{k+2d} \text{ (increasing } \tbinom{n}{k} \text{ for } k \leq n/2) \\
&\geq \sum_{k=2d}^{n/2} q_{k-d} \binom{n}{k-2d} + \sum_{k=n/2}^{n-2d} (1 - q_{n-k-d}) \binom{n}{k+2d} \text{ (as } q_{k+d} \geq 1 - q_{n-k-d}) \\
&= \sum_{k=2d}^{n/2} q_{k-d} \binom{n}{k-2d} + \sum_{k=2d}^{n/2} (1 - q_{k-d}) \binom{n}{n-k+2d} \\
&= \sum_{k=2d}^{n/2} q_{k-d} \binom{n}{k-2d} + \sum_{k=2d}^{n/2} (1 - q_{k-d}) \binom{n}{k-2d} \text{ (symmetry of } \tbinom{n}{k}) \\
&= \sum_{k=2d}^{n/2} \binom{n}{k-2d} = \sum_{k=0}^{n/2-2d} \binom{n}{k} ,
\end{aligned}
$$

where we made use of the equivalence $k \leq k' \Leftrightarrow \binom{n}{k} \leq \binom{n}{k'}$ as well as the identity $\binom{n}{k} = \binom{n}{n-k}$ for $k, k' \in \{0, \ldots, n/2\}$.

Essentially, we are confronted with the sum $\sum_{k=0}^{n/2} \binom{n}{k} \geq \frac{1}{2} 2^n$, where the last $2d$ terms are missing. We already know that the missing terms sum up to at most $(2d/n^{1/2})2^n$ such that the claim follows. □

By setting, for example, $d := n^{1/3}$, Lemma 6 states that the probability that an initial bit string yielding a function value of at least $w_0 + w/2 + \sum_{i=0}^{d-1} w_i$ is created still converges towards $1/2$. This signifies that the probability of creating an initial bit string with the "surplus" $\sum_{i=0}^{d-1} w_i$ above its expected value $w_0 + w/2$ is still $1/2 - o(1)$. If this surplus has occurred, from the initial bit string (which then belongs to $P(f)$) merely bit strings from $N(f)$ having a value of at most $w_0 + w/2 - \sum_{i=0}^{d-1} w_i$ can be reached (due to $w_0 > -w/2$). In other words only a mutation that would decrease the value of f by at least twice the surplus could be accepted during the optimization of f^2. That phenomenon constitutes the main idea of the following proof.

Proposition 6 *Let $f : \{0,1\}^n \to \mathbb{R}$ with $f(x) = w_0 + \sum_{i=1}^{n} w_i x_i$ be a linear function. With a probability of at least $1/8 - \epsilon$, $\epsilon > 0$ arbitrarily small, the (1+1) EA optimizes the square f^2 within $O(n \log n)$ steps.*

Proof: Recall that we assume the weights w_i to be sorted according to $w_1 \geq \cdots \geq w_n > 0$, and that the global maximum of f^2 is located in $\vec{1}$, i. e., $w_0 \geq -w/2$. We examine the probability that the initial bit string x^* yields a value of at least $f(x^*) \geq w_0 + w/2 + s$, where s is a "surplus" to its expected value $w_0 + w/2$. Under the assumption that the surplus has occurred, we analyze the probability that within $O(n \log n)$ steps a mutation decreasing the value of the linear function f by at least $2s$ (hereinafter called *bad mutation*) occurs at least once. Otherwise, the (1+1) EA will never "notice" that it runs on f^2 instead of f during these $O(n \log n)$ steps. Our goal is to prove that with a probability bounded below by $1/8 - o(1)$, the surplus is large enough for the probability of performing a bad mutation to converge towards zero.

To accomplish this, we divide bit strings $x \in \{0,1\}^n$ into three parts. With k being set to $k := n^{1/3} + 1$, we consider the first part consisting only of x_1, the second part ranging from x_2 to x_k, and the third part which comprises the bits x_{k+1} to x_n. Clearly, with probability $1/2$, the event $x_1^* = 1$ occurs. In addition, according to Lemma 6, we have

$$\sum_{i=2}^{k} w_i x_i^* \geq \frac{1}{2} \sum_{i=2}^{k} w_i + \sum_{i=0}^{(k-1)^{1/3}-1} w_{k-i}$$

with a probability of at least $1/2 - 2(k-1)^{1/3}(k-1)^{-1/2} = 1/2 - o(1)$. Thirdly, we use Lemma 6 once more to show that $\sum_{i=k+1}^{n} w_i x_i^* \geq \left(\sum_{i=k+1}^{n} w_i\right)/2$ occurs with a probability of at least $1/2$ (simply set $d := 0$). Since these events concern disjoint positions, which are initialized independently, we conclude that

$$\sum_{i=1}^{n} w_i x_i^* \geq \frac{w}{2} + \frac{w_1}{2} + \sum_{i=0}^{(k-1)^{1/3}-1} w_{k-i}$$

occurs with a probability of at least $1/8 - o(1)$. Then the surplus amounts to at least $s := w_1/2 + \sum_{i=0}^{(k-1)^{1/3}-1} w_{k-i}$. To overcome this surplus, i. e., to reach a bit string from $N(f)$, a mutation decreasing the value of f by at least $2s$ would have to be executed. Due to the choice of k and the decreasing order of the weights, we have $2s \geq w_1 + n^{1/9} w_k$. It remains to estimate how likely the event of at least one bad mutation during $c'cn \log n$ steps, $c, c' \in \mathbb{R}$, is. As we make no further assumptions about the weights of the linear function, two cases must be distinguished. Note that the distinction holds for all $n \in \mathbb{N}$, i. e., our asymptotic statement is valid independently of the magnitude of the weights.

Case 1: $w_k \geq w_1/n^{1/18}$.

This implies $n^{1/9} w_k \geq w_1 n^{1/18} \Rightarrow 2s \geq w_1 n^{1/18}$. Since no weight is larger than w_1, at least $n^{1/18}$ bits would have to flip simultaneously in order to execute a bad mutation. The latter has a probability of at most $1/(n^{1/18})! = 2^{-\Omega(n^{1/18} \log n)}$, which converges towards zero even after having been multiplied by an arbitrary polynomial in n. This means that especially the probability of a bad mutation within $c'cn \log n$ steps converges towards zero.

Case 2: $w_k < w_1/n^{1/18}$.
Once again consider the amount $2s \geq w_1 + n^{1/9} w_k$ by which the value of f would have to be decreased at least. We want to verify that the probability of flipping two bits from the first or second part of a bit string simultaneously is so small that this event even does not normally occur in $c'cn \log n$ steps. Obviously, the probability in one step is bounded above by $\binom{k}{2}\frac{1}{n^2} \leq n^{2/3}/n^2 = n^{-4/3}$. Therefore the probability of at least one mutation flipping two bits from $x_1, \ldots, x_k$ simultaneously is bounded above by $(c'cn \log n)n^{-4/3} = o(1)$. Another event that could represent a bad mutation would be a mutation flipping one bit from the first or second part of the string and some bits from the third part. But in that case, at least $n^{1/9}$ bits from the third part would have to flip simultaneously to decrease the value of f by at least $n^{1/9} w_k$ in addition to w_1. Of course the probability of flipping at least $n^{1/9}$ bits at once converges towards zero even in $c'cn \log n$ steps.

Having verified that the probability of at least one bad mutation in $c'cn \log n$ steps converges towards zero, we restrict the probability of not reaching the global optimum of f within $c'cn \log n$ steps by $1/c'$ using Markov's inequality. If c' is large enough, the probabilities of the errors "insufficient surplus", "at least one bad mutation in $O(n \log n)$ steps" and "time to optimize f larger than $c'cn \log n$" in common are bounded above by ϵ provided n is large enough. □

Despite the quite lenghty proof, the conclusions to be drawn from this result are easy. We have seen that squares of linear functions are easy if employing multistart variants of the (1+1) EA. Moreover, the result is valid for all even powers of linear functions f^k, $k \in \mathbb{N}$ even, which, for the (1+1) EA, are indistinguishable from the corresponding square. Marginally, we remark that odd powers f^k, $k \in \mathbb{N}$ even, of linear functions f are not distinguished from the linear functions themselves and thus are optimized within $O(n \ln n)$ steps.

8 Conclusion

This paper highlights some techniques to analyze the (1+1) EA and especially stresses the significance of the measures "expected runtime" and "success probability". As opposed to linear functions, which the (1+1) EA optimizes within $\Theta(n \log n)$ expected steps, already quadratic pseudo-boolean functions are an interesting class of functions which may pose severe problems to the (1+1) EA. Quadratic functions with non-negative weights are optimized within polynomial expected time, but as soon as general negative weights are allowed, the optimization problem becomes NP-hard. So it is not astonishing that we were able to find quadratic functions which provoke exponential expected runtimes of the (1+1) EA. But in many cases (e. g., concerning squares of linear functions) the success probability after a polynomial number of steps is so large that multistart variants of the (1+1) EA are very efficient. On the other hand, we created an "especially difficult" function called TRAP* which makes the (1+1) EA work an exponential time with a probability exponentially close to one. Here we cannot resort to multistart variants of the (1+1) EA. In fact, we even believe that more sophisticated evolutionary algorithms incorporating crossover operations will not succeed in optimizing TRAP* within polynomial expected time. Anyway, it serves as an argument against the hypothesis that evolutionary algorithms could easily cope with pseudo-boolean functions of a small degree.

References

[Ack87] D. Ackley. *A Connectionist Machine for Genetic Hillclimbers.* Kluwer Academic Publishers, Boston, 1987.

[DJW98a] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. Technical Report, Collaborative Research Center 531 CI-21/98, University of Dortmund, Germany, 1998.

[DJW98b] S. Droste, T. Jansen, and I. Wegener. A rigorous complexity analysis of the (1+1) evolutionary algorithm for linear functions with boolean inputs. *Evolutionary Computation*, 6(2):185–196, 1998.

[Fog95] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, Piscataway, NJ, 1995.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, 1989.

[Hol75] J. H. Holland. *Adaption in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, 1975.

[JW99] T. Jansen and I. Wegener. On the analysis of evolutionary algorithms – a proof that crossover really can help. In J. Nešetřil, editor, *Proceedings of the 7th Ann. European Symposium on Algorithms (ESA '99)*, pages 184–193, Berlin, 1999. Springer.

[Koz92] J. R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, Cambridge, MA, 1992.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[Müh92] H. Mühlenbein. How genetic algorithms really work I. Mutation and hillclimbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 15–25, Amsterdam, 1992. North Holland.

[Ros75] I. G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Etudes de Recherche Operationnelle*, 17:71–74, 1975.

[Rud97] G. Rudolph. *Convergence Properties of Evolutionary Algorithms.* PhD thesis, University of Dortmund, 1997. Verlag Dr. Kovač, Hamburg.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley and Sons, 1995.