On the Optimization of Monotone Polynomials by the (1+1) EA and Randomized Local Search

Ingo Wegener* and Carsten Witt*

FB Informatik, LS 2
Univ. Dortmund
44221 Dortmund, Germany
{wegener, witt}@ls2.cs.uni-dortmund.de

Abstract. Randomized search heuristics like evolutionary algorithms and simulated annealing find many applications, especially in situations where no full information on the problem instance is available. In order to understand how these heuristics work, it is necessary to analyze their behavior on classes of functions. Such an analysis is performed here for the class of monotone pseudo-boolean polynomials. Results depending on the degree and the number of terms of the polynomial are obtained. The class of monotone polynomials is of special interest since simple functions of this kind can have an image set of exponential size, improvements can increase the Hamming distance to the optimum and, in order to find a better search point, it can be necessary to search within a large plateau of search points with the same fitness value.

1 Introduction

Randomized search heuristics like random local search, simulated annealing, and all variants of evolutionary algorithms have many applications and practitioners report surprisingly good results. However, there are few theoretical papers on the design and analysis of randomized search heuristics.

In this paper, we investigate general randomized search heuristics, namely a random local search algorithm and a mutation-based evolutionary algorithm. It should be obvious that they do not improve heuristics with well-chosen problem-specific modules. Our motivation to investigate these algorithms is that such algorithms are used in many applications and that only an analysis will provide us with some knowledge to understand these algorithms better. This will give us the chance to improve these heuristics, to decide when to apply them, and also to teach them. The idea is to analyze randomized search heuristics for complexity-theoretical easy scenarios. One may hope that the heuristics behave similarly also on those functions which are "close" to the considered ones.

Each pseudo-boolean function $f\colon \{0,1\}^n \to \mathbb{R}$ can be written uniquely as a polynomial

* Supported in part by the Deutsche Forschungsgemeinschaft as a part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 622–633, 2003.

$$f(x) = \sum_{A \subseteq \{1,\ldots,n\}} w_A \cdot \prod_{i \in A} x_i \ .$$

The degree $d := \max\{|A| \mid w_A \neq 0\}$ and the number N of non-vanishing terms $w_A \neq 0$ are parameters describing properties of f. Note that the value of N can vary if we exchange the meanings of ones and zeros for some variables, i. e., replace some x_i by their negations, $1-x_i$. For instance, the product of all $(1-x_i)$ has the maximal number of 2^n non-vanishing terms but only one non-vanishing term if we replace x_i by $y_i := 1 - x_i$. The parameter N will be relevant in some upper bounds presented in this paper. However, all search heuristics that we will consider treat zeros and ones in the same way. Therefore, we may silently assume that in the polynomial representation of some monotone polynomial f, variables x_i have possibly been replaced by their negations $1-x_i$ in such a way that N takes its minimum value. Droste, Jansen and Wegener (2) have analyzed evolutionary algorithms on polynomials of degree $d = 1$ and Wegener and Witt (14) have investigated polynomials of degree $d = 2$. The last case is known to be NP-hard in general. A simpler subcase is the case of monotone polynomials where f can be written as a polynomial with non-negative weights on some variable set $z_1, \ldots, z_n$, where $z_i = x_i$ or $z_i = 1-x_i$. In the first case, the function is monotone increasing with respect to x_i and, in the second case, monotone decreasing. In this paper, we investigate randomized search heuristics for the maximization of monotone polynomials of degree bounded by some parameter d. Since all considered heuristics treat zeros and ones in the same way, we can restrict our analysis to monotone increasing polynomials where $z_i = x_i$ for all i. The results hold for all monotone polynomials.

The investigation of polynomials of small degree is well motivated since many problems lead to polynomials of bounded degree. Monotonicity is a restriction that simplifies the problem. However, in the general setting it is unknown whether the function is monotone increasing or decreasing with respect to x_i.

Evolutionary algorithms are general problem solvers that eventually optimize each $f\colon \{0,1\}^n \to \mathbb{R}$. We conjecture that our arguments would not lead to better upper bounds when we allow large populations and/or crossover. Indeed, it seems to be the case that they increase the optimization time moderately. Therefore, we investigate a simple standard evolutionary algorithm (EA), which is mutation-based and works with population size 1. This so-called (1+1) EA consists of an initialization step and an infinite loop.

(1+1) EA

Initialization:	Choose $a \in \{0,1\}^n$ randomly.
Loop:	The loop consists of a mutation and a selection step.
Mutation:	For each position i, decide independently whether a_i should be flipped (replaced by $1 - a_i$). The flipping probability equals $1/n$.
Selection:	Replace a by a' iff $f(a') \geq f(a)$.

The advantage of the (1+1) EA is that each point can be created from each point with positive probability but steps flipping only a few bits are preferred. Therefore, it is not necessary (as in simulated annealing) to accept worsenings.

Random local search (RLS) flips only one bit per step. The algorithm is also called *random mutation hill-climbing* (Mitchell, Holland and Forrest 10).

RLS works like the (1+1) EA with a different mutation operator.
Mutation: Choose $i \in \{1, \ldots, n\}$ randomly and flip a_i.

RLS cannot escape from local optima, where the local neighborhood is the Hamming ball with distance 1. However, it can optimize monotone polynomials (by our assumption the optimum is 1^n) since, for each a, there exists a sequence $a_0 = a, a_1, \ldots, a_m = 1^n$ such that $m \leq n$, the Hamming distance of a_i and a_{i+1} equals 1, and $f(a_0) \leq f(a_1) \leq \cdots \leq f(a_m)$. The analysis of RLS will be much easier than the analysis of the (1+1) EA; however, only the (1+1) EA is a general problem solver. The difficulty is that accepted steps can increase the number of zeros and the Hamming distance to the optimum. The problem for all heuristics is that it can be necessary to change many bits (up to d) until one finds a search point with larger fitness.

We have to discuss how we analyze RLS and the (1+1) EA, which are defined as infinite loops. In applications, we need a stopping criterion; however, this is not the essential problem. Hence, we are interested in the random optimization time X_f, defined as the minimum time step t where an optimal search point is created. Its mean value $E(X_f)$ is called the expected optimization time and $\text{Prob}(X_f \leq t)$ describes the success probability within t steps.

We present monotone polynomials of degree d where the expected optimization time equals $\Theta((n/d) \cdot \log(n/d+1) \cdot 2^d)$ for RLS and the (1+1) EA, and we believe that the upper bound holds for all monotone polynomials. This can be proved for RLS, but our best bound for the (1+1) EA is worse and depends on N. For this reason, we also investigate a class of algorithms that bridge the difference between RLS and the (1+1) EA. The first idea is to reduce the mutation probability $1/n$ of the (1+1) EA. However, then we increase the probability of useless steps flipping no bit. Hence, we guarantee that at least one bit is flipped. We call the new algorithm RLS_p since it is a modification of RLS.

RLS$_p$ works like the (1+1) EA and RLS with a different mutation operator.
Mutation: Choose $i \in \{1, \ldots, n\}$ randomly and flip a_i. For each $j \neq i$, flip a_j independently of the other positions with probability p.

Obviously, RLS_p equals RLS for $p = 0$. For $p = 1/n$, RLS_p is close to the (1+1) EA, but omits steps without flipped bit. Hence, we investigate RLS_p only for $0 \leq p \leq 1/n$ and try to maximize p such that we can prove the upper bound $O((n/d) \cdot \log(n/d+1) \cdot 2^d)$ on the expected optimization time of RLS_p on monotone polynomials.

The paper is structured as follows. Search heuristics with population size 1 lead to a Markov chain on $\{0,1\}^n$. Therefore, we have developed some results on such Markov chains. The results are presented without proof in an appendix. In Sect. 2, we investigate the very special case of monomials, i. e., monotone polynomials where $N = 1$. These results are crucial since we later consider how long it takes to maximize a special monomial in the presence of many other monomials in the polynomial. In the Sections 3, 5, and 6, we prove upper bounds on the

expected optimization time of the algorithms RLS, RLS_p and the (1+1) EA on monotone polynomials. In Sect. 4, we present a worst-case monotone polynomial for RLS, which is conjectured to also be a worst-case monotone polynomial for RLS_p and the (1+1) EA. We finish with some conclusions. Some preliminary ideas of this paper have been given in a survey (Wegener 13).

2 The Optimization of Monomials

Because of the symmetry of all considered algorithms with respect to the bit positions and the role of zeros and ones, we can investigate w.l.o.g. the monomial $m(x) = x_1 \cdots x_d$. The following result has been proved by Garnier, Kallel and Schoenauer (4) for the special case $d = n$ and for the algorithms RLS_0 and (1+1) EA. We omit the proof of our generalizations.

Theorem 1. *The algorithms RLS_p, $p \leq 1/n$, and (1+1) EA in their pure form and under the condition of omitting all steps flipping more than two bits optimize monomials of degree d in an expected time of $\Theta((n/d) \cdot 2^d)$. The upper bounds also hold if the initialization is replaced by the deterministic choice of any $a \in \{0,1\}^n$.*

3 On the Analysis of Random Local Search

The random local search algorithm, RLS, is easy to analyze since it flips one bit per step. This implies that activated monomials, i. e., monomials where all bits are 1, never get passive again.

Theorem 2. *The expected optimization time of RLS on a monotone polynomial of degree d is bounded by $O((n/d) \cdot \log(n/d+1) \cdot 2^d)$.*

Sketch of proof. First, we investigate the case of polynomials with pairwise non-overlapping monomials, i. e., monomials that do not share variables. For each monomial of degree i, the probability of activating it in $O(2^i)$ steps is at least $1/2$ (see Theorem 1) if we count only steps flipping bits of the monomial. Now the arguments for proving the famous Coupon Collector's Theorem (see Motwani and Raghavan 11) can be applied to obtain the result.

In the general case, we choose a maximal set M_1 of pairwise non-overlapping monomials and consider the time T_1 until all monomials of M_1 are activated. The existence of further monotone monomials can only decrease the time for activating the monomials of M_1. Here the property of monotonicity is essential. Hence, by the considerations above, $E(T_1)$ is bounded by $O((n/d) \cdot \log(n/d+1) \cdot 2^d)$. The key observation is that afterwards each passive monomial contains at least one variable that is shared by an active monomial and, therefore, is fixed to 1. Hence, we are essentially in the situation of monomials whose degree is bounded by $d-1$. This argument can be iterated and we obtain an upper bound on the expected optimization time, which is the sum of all $O((n/i) \cdot \log(n/i+1) \cdot 2^i)$, $1 \leq i \leq d$. Simple calculations show that this sum is only by a constant factor larger than the term for $i = d$. This proves the theorem. □

4 Royal Roads as a Worst-Case Example

It is interesting that the *royal road functions* RR_d introduced by Mitchell, Forrest and Holland (9) are the most difficult monotone polynomials for RLS and presumably also for RLS_p and the (1+1) EA. The function RR_d is defined for $n = kd$ by

$$\mathrm{RR}_d(x) = \sum_{i=0}^{k-1} x_{id+1} \cdots x_{id+d} \ ,$$

or the number of blocks of length d containing ones only. Theorem 2 contains an upper bound of $O((n/d) \cdot \log(n/d+1) \cdot 2^d)$ for the expected optimization time of RLS on RR_d, and this can be easily generalized to RLS_p, where $p \leq 1/n$, and the (1+1) EA. The result for RLS was also shown by Mitchell, Holland and Forrest (10). The mentioned upper bound disproved the conjecture that RR_d are royal roads for the crossover operator. Real royal roads have been presented only recently by Jansen and Wegener (7,8). Here we prove matching lower bounds on the expected optimization time for RR_d. First, we investigate RLS and, afterwards, we transfer the results to RLS_p and the (1+1) EA.

Theorem 3. *The probability that RLS has optimized the function* RR_d *within* $(n/d) \cdot \log(n/d) \cdot 2^{d-5}$ *steps is* $o(1)$ (*convergent to* 0) *if* $d = o(n)$. *The expected optimization time of RLS on* RR_d *equals* $\Theta((n/d) \cdot \log(n/d+1) \cdot 2^d)$.

Sketch of proof. We only have to prove the lower bounds. For $d = \Theta(n)$, the result has been proved by Droste, Jansen, Tinnefeld and Wegener (1) for all considered algorithms. For $d = O(1)$, the bounds follow easily by considering the time until each bit initialized as 0 has flipped once (again the Coupon Collector's Theorem).

In the following, we assume that $d = \omega(1)$ and $d = o(n)$. First, we investigate the essential steps for a single monomial m, i. e., those steps flipping a bit of m. Let τ be the random number of essential steps until m is activated. Garnier, Kallel and Schoenauer (4) have proved that this process is essentially memoryless. More precisely

$$|\mathrm{Prob}(\tau \geq t) - \mathrm{Prob}(\tau \geq t + t' \mid \tau \geq t')| = O(1/d) \ ,$$

and $\mathrm{Prob}(\tau \geq t)$ is approximately $1 - e^{-t}$. Hence, since $d = \omega(1)$, we have $\mathrm{Prob}(\tau \leq 2^{d-1} + t' \mid \tau \geq t') \leq 1/2$ for all t'.

The next idea is that all monomials are affected by essentially the same number of steps. However, many steps for one monomial imply less steps for the other monomials. We partition the $k \cdot (\log k) \cdot 2^{d-5}$ steps into $(\log k)/4$ phases of length $k \cdot 2^{d-3}$ each. Let p_i be the random number of passive monomials after phase i. We claim that the following events all have an exponentially small probability with respect to $k^{1/4}$: the event $p_0 < k/2$ and the events $p_i < p_{i-1}/8$. Hence, the probability that none of these events happens is still $1 - o(1)$. This implies the existence of at least $p_0 \cdot (1/8)^{(\log k)/4} \geq k^{1/4}/2$ passive monomials at the end of the last phase implying that RR_d is not optimized.

The expected value of p_0 equals $k \cdot (1-2^{-d})$, and, therefore, the probability of the event $p_0 < k/2$ can be estimated by Chernoff bounds. If $p_j \geq p_{j-1}/8$ for all $j < i$, there are at least $k^{1/4}/2$ passive monomials at the end of phase $i-1$. The expected number of steps essential for one of the passive monomials in phase i equals $p_{i-1} \cdot 2^{d-3}$, and the probability that this number is less than $p_{i-1} \cdot 2^{d-2}$ is exponentially close to 1. By the pigeon-hole principle, there are at most $p_{i-1}/2$ monomials with at least 2^{d-1} essential steps each. Pessimistically, we assume all these monomials to become active in phase i. We have proved before that each other monomial activates with probability at most 1/2. By Chernoff bounds, the probability of activating at least 3/4 of these and altogether at least 7/8 of the passive monomials is exponentially small. This proves the theorem. □

Theorem 4. *For each $\varepsilon > 0$, the probability that the (1+1) EA is on RR_d by a factor of $1+\varepsilon$ faster than RLS is $O(1/n)$. The same holds for RLS$_p$, $p \leq 1/n$, and the factor $2+\varepsilon$.*

Sketch of proof. We prove the result on the (1+1) EA by replacing the (1+1) EA by a faster algorithm $(1+1)^*$ EA and comparing the faster algorithm with RLS. A step of the $(1+1)^*$ EA works as follows. First, the number k of flipped bits is chosen according to the same distribution as for the (1+1) EA. Then the $(1+1)^*$ EA flips a random subset of k bits. This can be realized as follows. In each step, one random bit is flipped until one obtains a point of Hamming distance k to the given one. Now the new search point of the $(1+1)^*$ EA is obtained as follows. The selection procedure of RLS is applied after each step. This implies by the properties of the royal road functions that we obtain a search point a^* compared to the search point a of the (1+1) EA such that $a \leq a^*$ according to the componentwise partial order. This implies that the $(1+1)^*$ EA reaches the optimal string 1^n no later than the (1+1) EA.

However, the $(1+1)^*$ EA chooses flipped bits as RLS, and it uses the same selection procedure. The difference is that the $(1+1)^*$ EA sometimes simulates many steps of RLS in one step, while the $(1+1)^*$ EA flips on average one bit per step. It is easy to see that we have to consider $t = \Omega(n)$ steps. Then it is for each $\gamma > 0$ very likely that the $(1+1)^*$ EA flips not more than $(1+\gamma)t$ bits within t steps. Moreover, with high probability the number of flipped bits is bounded by δn in each step, $\delta > 0$ a constant. Let a be the starting point of the simulation of one step. The probability of increasing the Hamming distance to a with the next flipped bit is at least $1-\delta$. Hence, with large probability we have among t steps an overhead of $(1-3\delta)t$ distance-increasing steps. Hence, the probability that $(1+\gamma) \cdot t/(1-3\delta)$ steps of RLS do not suffice to simulate t steps of the $(1+1)^*$ EA is exponentially small. Choosing γ and δ such that $(1+\gamma)/(1-3\delta) = 1+\varepsilon$, we are done. The statement on RLS$_p$ follows in the same way taking into account that RLS$_p$, $p \leq 1/n$, flips on average not more than two bits per step. □

5 On the Analysis of RLS$_p$

In contrast to RLS, RLS$_p$ with $p > 0$ can deactivate monomials by simultaneously activating other monomials. Even the analysis of the time until a single

monomial is activated becomes much more difficult. Steps where two bits of the monomial flip from 0 to 1 and only one bit flips from 1 to 0 may decrease the fitness and be rejected. Hence, we do not obtain simple Markov chains as in the case of RLS or in the case of single monomials.

We can rule out the event of three or more flipped bits contained in the same monomial if its degree is not too large, more precisely $d = O(\log n)$. This make sense since Theorems 3 and 4 have shown that we cannot obtain polynomial upper bounds otherwise. To analyze the optimization process of RLS_p on a monotone polynomial, we first consider some fixed, passive monomial and estimate the time until it becomes active for the first time. The best possible bound $O((n/d) \cdot 2^d)$ can be proved if p is small enough. Afterwards, we apply this result in order to bound the expected optimization time on the monotone polynomial. The bound we obtain here is close to the lower bound from Theorem 4.

Lemma 1. *Let f be a monotone polynomial of degree $d \leq c \log n$ and let m be one if its monomials. There is a constant $\alpha > 0$ such that RLS_p with $p = \min\{1/n, \alpha/(n^{c/2} \log n)\}$ activates m in an expected time of $O((n/d) \cdot 2^d)$ steps.*

Sketch of proof. The idea is to prove that RLS_p activates m with a constant probability $\varepsilon > 0$ within a phase of $c' \cdot (n/d) \cdot 2^d$ steps, for some constant c'. Since our analysis does not depend on the starting point, this implies an upper bound $c' \cdot (n/d) \cdot 2^d/\varepsilon$ on the expected time to activate m. We assume w.l.o.g. that $m = x_1 \cdots x_d$ and call it the *prefix* of the search point.

We bound the probability of three events we consider as a failure. The first one is that we have a step flipping at least three prefix bits in the phase. The second one is that, under the condition that the first type of failure does not happen, we do not create a search point where m is active in the phase. The third one occurs if the first search point in which m is active is not accepted. If none of the failures occurs, m is obviously activated.

The first and third type of failure can be handled by standard techniques. A simple calculation shows that the first type of failure occurs with probability at most d^3p^2/n in one step. Multiplying by the length of the phase, we obtain a failure probability bounded by a constant if α is small enough. For the third failure type it is necessary that at least one of the suffix bits $x_{d+1}, \ldots, x_n$ flips. Since we assume m to be activated in the considered step, the related conditional probability of not flipping a suffix bit can be bounded below by the constant $1/(2e)$. All this holds also under the condition that the first two types of failure do not happen.

For the second type of failure, we apply the techniques developed in the appendix by comparing the Markov chains Y_0 and Y_1. Y_0 equals RLS_p^*, namely RLS_p on the monomial m, where the condition holds that no step flips more than two bits of the prefix. Y_1 equals RLS_p^* on the monotone polynomial f, which again equals RLS_p under the condition that no step flips more than two prefix bits. Both Markov chains are investigated on the compressed state space $D = \{0, \ldots, d\}$ representing the number of 1-bits in the prefix. We can ignore the fact that the Markov chain Y_1 is not time-homogeneous by deriving bounds on its transition probabilities that hold for all search points. We denote these bounds

still by $P_1(i,j)$. Then the following conditions for Lemma 7 imply the bound $O((n/d) \cdot 2^d)$ of the lemma. (See also Definitions 1, 2 and 3 in the appendix.)

1. Y_1 has a relative advantage to Y_0 for c-values such that $c_{\min} \geq 1/(2e)$,
2. Y_0 has a $(2e-1)$-advantage, and
3. $E(\tau_0^i) = O((n/d) \cdot 2^d)$ for all i.

The third claim is shown in Theorem 1. The second one follows from Lemma 4 since $d \leq (n-1)/(4e+1)$ if n is large enough. For the first claim, recall that at most two prefix bits flip. Now Definition 3 implies that we have to consider $c(i,j) = P_1(i,j)/P_0(i,j)$ for $j \in \{i-2, i-1, i+1, i+2\}$ and to prove that

1. $1/(2e) \leq c(i,i+1) \leq 1$
2. $c(i,i+2) \geq c(i,i+1)$,
3. $c(i,i-1) \leq c(i,i+1)$, and
4. $c(i,i-2) \leq c(i,i+1)$ (or even $c(i,i-2) \leq c(i,i-1)$).

The inequality $c(i,i+1) \leq 1$ holds since RLS_p^* on m accepts each new string as long as the optimum is not found. The bound $c(i,i+1) \geq 1/(2e)$ follows from the fact that RLS_p^* on the monotone polynomial f accepts a step where one prefix bit flips from 0 to 1 and no suffix bit flips.

For the remaining inequalities, observe that $c(i,j)$ is the conditional probability of RLS_p^* accepting (for f) a search point x given that x contains j prefix ones and has been created from a string with i prefix ones. The idea is to condition these probabilities even further by considering a fixed change of the suffix bits. Let the suffix change from c to c', and let b be a prefix containing i ones. If RLS_p^* accepts the string (b',c'), where b' is obtained from b by flipping a zero to one, then RLS_p^* also accepts (b'',c'), where b'' is obtained from b' by flipping another zero. Estimating the number of such strings (b'',c') leads to $c(i,i+2) \geq c(i,i+1)$. By a dual argument, we prove $c(i,i-2) \leq c(i,i-1)$. Finally, the inequality $c(i,i+1) \geq c(i,i-1)$ follows from the following observation. If there is at least one string (b',c') that is not accepted and where b' has been obtained by flipping a zero of b, then all strings (b'',c'), where b'' has been obtained by flipping a one of b, are also rejected. This completes the proof. □

Theorem 5. *The expected optimization time of RLS_p on a monotone polynomial f of degree $d \leq c \log n$ is bounded above by $O((n^2/d) \cdot 2^d)$ if $0 < p \leq \min\{(1-\gamma)/(2dn), \alpha/(n^{c/2} \cdot \log n)\}$ for the constant α from Lemma 1 and each constant $\gamma > 0$.*

Sketch of proof. The optimization process is not reflected by the f-value of the current search point. An f-value of v can be due to a single monomial of degree 1 or to many monomials of large degree. Instead, we count the number of *essential ones* (with respect to f). A 1-entry of a search point is called essential if it is contained in an activated monomial of f. All other 1-entries may flip to 0 without decreasing the f-value and are therefore called inessential. 0-entries are always called inessential. An essential one can only become inessential if simultaneously some monomial is activated. A step where a monomial is activated is called

essential. By Lemma 1, it suffices to prove an $O(n)$ bound on the expected number of essential steps.

To prove this bound, we apply Lemma 8, an approach sometimes called *drift analysis* (see Hajek 5; Sasaki and Hajek 12; He and Yao 6). Let X_i be the number of essential ones after the i-th essential step, i. e., X_0 is the number of essential ones after initialization. Let $D_0 = X_0$ and $D_i = X_i - X_{i-1}$ for $i \geq 1$. Then we are interested in τ, the minimal i where $D_0 + D_1 + \cdots + D_i = n$. Some conditions of Lemma 8 are verified easily. We have $|D_i| \leq n$ and $E(\tau) < \infty$ since there is always a probability of at least p^n to create the optimal string. If we can prove that $E(D_i \mid \tau \geq i) \geq \varepsilon$ for some $\varepsilon > 0$, Lemma 8 implies $E(\tau) = O(n)$.

At least one monomial is activated in an essential step, i. e., at least one bit turns from inessential into essential. We have to bound the expected number of bits turning from essential into inessential. Since the assumption that the new search point is accepted only decreases this number, we consider the number of flipped ones under the condition that a 0-bit is flipped.

Let Y be the random number of additional bits flipped by RLS_p under the assumption that a specified bit (activating a monomial) flips. A lengthy calculation shows that $E(Y) \leq (1-\varepsilon)/d$ for some $\varepsilon > 0$ since $p \leq (1-\gamma)/(2dn)$. The problem is that given $Y = i$, more than i bits may become inessential. Therefore, we upper bound the expected number of bits turning from essential into inessential if $Y = i$. In the worst case, these i flipped bits contain essential ones. Since we do not take into account whether the new search point is accepted, each subset of size i of the essential ones has the same probability of being the flipped ones. We apply the accounting method on the random number L of essential ones becoming inessential if a random essential one flips. The idea is as follows. In order to make the essential one in bit j inessential, some essential one contained in all monomials that contain x_j flips. This leads to $E(L) \leq d$. Then we can show that by flipping i essential ones, we lose on average at most id essential ones. Since $E(Y) \leq (1-\varepsilon)/d$, the expected number of essential ones becoming inessential is at most $1 - \varepsilon$. Since at least one bit gets essential, this implies $E(D_i \mid \tau \geq i) \geq \varepsilon$ and the theorem. □

6 On the Analysis of the (1+1) EA

Since the (1+1) EA flips too many bits in a step, the bound of Theorem 5 cannot be transferred to the (1+1) EA, and we only obtain a bound depending on the parameter N here. However, a result corresponding to Lemma 1 can be proved.

Lemma 2. *Let f be a monotone polynomial of degree d, and let m be one of its monomials. There is a constant α such that the (1+1) EA activates m in an expected number of $O((n/d) \cdot 2^d)$ steps if $d \leq 2 \log n - 2 \log \log n - \alpha$.*

Sketch of proof. We follow the same structure as in the proof of Lemma 1 and need only few different arguments.

First, the probability of at least three flipped prefix bits in one step is bounded above by $d^3 n^{-3}/6$ for the (1+1) EA. Therefore, the probability that such a step

happens in a phase of length $c \cdot (n/d) \cdot 2^d$ for some constant c is still smaller than 1 by choosing α large enough. Second, the probability that no suffix bit flips is at least $(1-1/n)^{n-1} \geq 1/e$. Also Lemma 7 can be applied with a value $c_{\min} \geq 1/e$ and an $(e-1)$-advantage of Y_0. It is again possible to apply Theorem 1. Instead of Lemma 4, Lemma 3 is applied. Here it is sufficient that $d \leq (n-1)/e$. Finally, the argument that Y_1 has a relative advantage to Y_0 for c-values such that $c_{\min} \geq 1/e$ can be used in the same way here. □

Theorem 6. *The expected optimization time of the* (1+1) *EA on a monotone polynomial with N monomials and degree $d \leq 2 \log n - 2 \log \log n - \alpha$ for the constant α from Lemma 2 is bounded above by $O(N \cdot (n/d) \cdot 2^d)$.*

Sketch of proof. Here we use the method of measuring the progress by fitness layers. Let the positive weights of the N monomials be sorted, i. e., $w_1 \geq \cdots \geq w_N > 0$. We partition the search space $\{0,1\}^n$ into $N+1$ layers $L_0, \ldots, L_N$, where

$$L_i = \{a \mid w_1 + \cdots + w_i \leq f(a) < w_1 + \cdots + w_{i+1}\}$$

for $i < N$, and L_N contains all optimal search points. Each layer L_i, $i < N$, is left at most once. Hence, it is sufficient to prove a bound of $O((n/d) \cdot 2^d)$ on the expected time to leave L_i. Let $a \in L_i$. Then there exists some $j \leq i+1$ such that the monomial m_j corresponding to w_j is passive. By Lemma 2, the expected time until m_j is activated is bounded by $O((n/d) \cdot 2^d)$. We can bound the probability of not leaving L_i in the step activating m_j by $1 - e^{-1}$. The expected number of such phases is therefore bounded by e. □

Conclusions

We have analyzed randomized search heuristics like random local search and a simple evolutionary algorithm on monotone polynomials. The conjecture is that all these algorithms optimize monotone polynomials of degree d in an expected number of $O((n/d) \cdot \log(n/d+1) \cdot 2^d)$ steps. It has been shown that some functions need that amount of time. Moreover, for random local search the bound has been verified. If the expected number of flipped bits per step is limited, a little weaker bound is proved. However, for the evolutionary algorithm only a bound depending on the number of monomials with non-zero weights has been obtained. Although there is room for improvement, the bounds and methods are a step to understand how randomized search heuristics work on simple problems.

References

Droste, S., Jansen, T., Tinnefeld, K., Wegener, I.: A new framework for the valuation of algorithms for black-box optimization. In: Proc. of FOGA 7. (2002) 197–214. Final version of the proceedings to appear in 2003.

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

Feller, W.: An Introduction to Probability Theory and its Applications. Wiley, New York (1971)
Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. Evolutionary Computation **7** (1999) 173–203
Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. Advances in Applied Probability **14** (1982) 502–525
He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence **127** (2001) 57–85
Jansen, T., Wegener, I.: Real royal road functions – where crossover provably is essential. In: Proc. of GECCO 2001. (2001) 375–382
Jansen, T., Wegener, I.: The analysis of evolutionary algorithms – a proof that crossover really can help. Algorithmica **34** (2002) 47–66
Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and GA performance. In Varela, F.J., Bourgine, P., eds.: Proc. of the First European Conference on Artificial Life, Paris, MIT Press (1992) 245–254
Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing. In Cowan, J.D., Tesauro, G., Alspector, J., eds.: Advances in Neural Information Processing Systems. Volume 6., Morgan Kaufmann (1994) 51–58
Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)
Sasaki, G.H., Hajek, B.: The time complexity of maximum matching by simulated annealing. Journal of the ACM **35** (1988) 387–403
Wegener, I.: Theoretical aspects of evolutionary algorithms (invited paper). In: Proc. of ICALP 2001. Number 2076 in LNCS (2001) 64–78
Wegener, I., Witt, C.: On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions. To appear in Journal of Discrete Algorithms (2003).

A Some Results on Markov Chains

The behavior of randomized search heuristics on single monomials is of special interest. For a monomial of degree d, the current state can be identified with the number of ones among the variables of the monomial. This leads to the state space $D = \{0, \ldots, d\}$. In order to obtain an ergodic Markov chain, we replace the selection operator by the selection operator that accepts each a', i. e., a' always replaces a. Then we are interested in the minimal t such that in time step t the state d is reached. The transition probabilities for the (1+1) EA under the condition that each step changes the state number by at most 2 are denoted by $Q(i,j)$ and the corresponding transition probabilities for RLS_p by $R(i,j)$.

We prove that these Markov chains have the property that it is more likely to reach from i "higher" states than from $i-1$. This intuitive notion is formalized as follows.

Definition 1. *Let $P(i,j)$ be the transition probabilities of a time-homogeneous Markov chain on $D = \{0, \ldots, d\}$. The Markov chain has an ε-advantage, $\varepsilon \geq 0$, if for all $i \in \{0, \ldots, d-2\}$ the following properties hold.*

1. $P(i,j) \geq (1+\varepsilon) \cdot P(i+1,j)$ for $j \leq i$,
2. $P(i+1,j) \geq (1+\varepsilon) \cdot P(i,j)$ for $j > i$.

Lemma 3. *Let $\varepsilon \geq 0$ and $d \leq (n-1)/(1+\varepsilon)$. Then the Markov chain with transition probabilities $Q(i,j)$ has an ε-advantage.*

Lemma 4. *Let $\varepsilon \geq 0$ and $d \leq (n-1)/(3+2\varepsilon)$. Then the Markov chain with transition probabilities $R(i,j)$ has an ε-advantage.*

We are interested in the random variable τ^k describing for a time-homogeneous Markov chain Y on D with transition probabilities $P(i,j)$ the first point of time when it reaches state d if it starts in state k. If Y has a 0-advantage, it should be advantageous to start in a "higher state." This is made precise in the following lemma.

Lemma 5. *Let $P(i,j)$ be the transition probabilities of a time-homogeneous Markov chain with 0-advantage on $D = \{0,\dots,d\}$. Then $\mathrm{Prob}(\tau^i \geq t) \geq \mathrm{Prob}(\tau^{i+1} \geq t)$ for $0 \leq i \leq d-1$ and each t. Moreover, $E(\tau^i) \geq E(\tau^{i+1})$.*

We compare different Markov chains. The complicated Markov chain Y_1, describing a randomized search heuristic on a monotone polynomial with many terms, is compared with the simple Markov chain Y_0, describing a randomized search heuristic on a single monomial. The idea is to use results for Y_0 to obtain results for Y_1. We denote by τ_0^i and τ_1^i the random time to reach state d from state i with respect to Y_0 and Y_1, respectively.

Definition 2. *Let $P_0(i,j)$ and $P_1(i,j)$ be the transition probabilities of the time-homogeneous Markov chains Y_0 and Y_1 on $D = \{0,\dots,d\}$. The Markov chain Y_1 has an advantage compared to Y_0 if $P_1(i,j) \geq P_0(i,j)$ for $j \geq i+1$ and $P_1(i,j) \leq P_0(i,j)$ for $j \leq i-1$.*

Lemma 6. *If Y_1 has an advantage compared to Y_0 and Y_0 has a 0-advantage, then $\mathrm{Prob}(\tau_1^i \geq t) \leq \mathrm{Prob}(\tau_0^i \geq t)$ and $E(\tau_1^i) \leq E(\tau_0^i)$.*

Finally, we apply Lemma 6 to compare two Markov chains Y_0 and Y_1 where weaker conditions hold than in Lemma 6. We compare Y_0 and Y_1 by parameters $c(i,j)$ such that $P_1(i,j) = c(i,j) \cdot P_0(i,j)$. This includes an arbitrary choice of $c(i,j)$ if $P_0(i,j) = P_1(i,j) = 0$.

Definition 3. *Let $P_0(i,j)$ and $P_1(i,j)$ be the transition probabilities of Y_0 and Y_1 such that $P_1(i,j) = c(i,j) \cdot P_0(i,j)$ for some $c(i,j)$. Then Y_1 has a relative advantage compared to Y_0 if $c(i,j) \geq c(i,i+1)$ for $j \geq i+1$, $c(i,j) \leq c(i,i+1)$ for $j \leq i-1$, and $0 < c(i,i+1) \leq 1$ for all $i \leq d-1$.*

Lemma 7. *If Y_1 has a relative advantage compared to Y_0 and Y_0 has a $(c_{\min}^{-1}-1)$-advantage, then $E(\tau_1^i) \leq c_{\min}^{-1} \cdot E(\tau_0^i)$ for $c_{\min} := \min\{c(i,i+1) \mid 0 \leq i \leq d-1\}$.*

The last result in this technical section is a generalization of Wald's identity (see Feller 3). We do not claim to be the first to prove this result, but we have not found it in the literature.

Lemma 8. *Let D_i, $i \in \mathbb{N}$, be a sequence of random variables such that $|D_i| \leq c$ for a constant c. For $s > 0$, let τ_s be the minimal i where $D_1 + \cdots + D_i = s$. If $E(\tau_s) < \infty$ and $E(D_i \mid \tau_s \geq i)$ is bounded below by a positive constant ℓ for all i where $\mathrm{Prob}(\tau_s \geq i) > 0$, then $E(\tau_s) \leq s/\ell$.*