Not All Linear Functions Are Equally Difficult for the Compact Genetic Algorithm

Stefan Droste [*]
LS Informatik 2
Universität Dortmund
44221 Dortmund
Germany
stefan.droste@udo.edu

ABSTRACT

Estimation of distribution algorithms (EDAs) try to solve an optimization problem by finding a probability distribution focussed around its optima. For this purpose they conduct a sampling-evaluation-adjustment cycle, where search points are sampled with respect to a probability distribution, which is adjusted according to the evaluation of the sampled points. Although there are many successful experiments suggesting the usefulness of EDAs, there are only few rigorous theoretical results apart from convergence results without time bounds. Here we present first rigorous runtime analyses of a simple EDA, the compact genetic algorithm, for linear pseudo-boolean functions on n variables. We prove a number of results showing that not all linear functions have the same asymptotical runtime.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms, Performance

Keywords

Theoretical Analysis, Runtime, Compact Genetic Algorithm

1. INTRODUCTION

Evolutionary algorithms (EAs) (see [1]) essentially depend on a population of search points evolving during the run of the EA. Estimation of distribution algorithms (EDAs), on the other hand, get by without a population to store all the information gained so far about the problem at hand. Instead, EDAs utilize a probability distribution to generate new search points. Then the distribution is modified to give those search points with a good objective function value a higher probability. This process is repeated (until some stopping criterion holds) by using the modified probability distribution as a basis for the next sampling. We hope that the last found probability distribution is focussed around points with a good objective function value, so that these are being sampled during the run of the EDA with high probability. Hence, EDAs can be interpreted as EAs whose population is replaced by a probability distribution resp. EAs where the genetic operators operate on the whole population (and not only on one or two search points like mutation and crossover normally do).

EDAs were introduced in the field of EAs for the first time by [9] and since then a lot of different EDAs and successful applications have been presented (see [6] for an overview). Naturally, we are also interested in theoretical results proving which type of problems can be solved with EDAs and which cannot. The convergence of some EDAs was analyzed theoretically (e. g., see [8] or [10]), giving us valuable insights in the behavior of EDAs without time limits. But a finite, yet exponential runtime until convergence is of no practical relevance for all problem dimensions not very small. To analyze the short time behavior we have to apply runtime analysis, estimating the number of steps until a sufficiently good probability distribution (resp. a sufficiently good search point for classical EAs) has been found.

Rigorous runtime analysis is well-established in the theory of EAs (see [2]). While experiments often give the initial ideas for a theoretical analysis, only a rigorous theoretical analysis estimating the probabilities of all errors introduced by assumptions can lead to results of mathematical certainty (see [13]). The rigorous mathematical analysis of EAs has started with simple mutation-based EAs and test problems (e. g., see [3] or [11]) and nowadays progresses towards more realistic EAs for combinatorial optimization problems (see [12]). Certainly, a similar development would be most beneficial for the field of EDAs.

The compact genetic algorithm (cGA) is an ideal starting point for the rigorous runtime analysis of EDAs: firstly presented in [4] its main purpose as a simplification of a GA was to investigate the effect of populations in GAs more

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

closely. Consider a GA using a population of K search points and a selection, where two elements of the population are randomly chosen and replaced by two copies of the better search point (with respect to the objective function $f : S^n \to \mathbb{R}$). Consequently, the proportion of the better individual increases by $1/K$. The cGA mimics this behavior while representing its probability distribution by n independent probability distributions for the n components of the search space S^n. Hence, it is not able to represent dependencies between the different components, intuitively making it a bad candidate for the optimization of all non-linear functions. But there is no rigorously proven knowledge about the runtime of the cGA on linear functions ([4] presented some experimental results) either. In this paper we will prove some runtime bounds with respect to K and n, following the widely accepted convention in the analysis of algorithms (see [7]) that the asymptotical dependence on K and n is more important than (usually small) constant factors.

In the next section we formally define the cGA. Then we introduce the surplus, a very helpful measure for the analysis of the cGA. This is used in Section 4 to prove a general lower bound of $\Omega(K\sqrt{n})$ on the expected runtime of the cGA (see [7] for the used notation of asymptotical growth rates). In Sections 5 resp. 6 we prove an upper resp. lower bound on the runtime for OneMax resp. BinVal, two linear functions well examined in the field of EAs. These bounds prove that the runtime of the cGA on linear functions can vary at least between $O(K\sqrt{n})$ and $\Omega(Kn)$. Furthermore, their proofs show how established methods in the analysis of population-based EAs can help to analyze EDAs like the cGA. The paper ends with some conclusions and open questions.

2. THE COMPACT GENETIC ALGORITHM AND LINEAR FUNCTIONS

The compact genetic algorithm (cGA) introduced in [4] is probably the most simple EDA possible. This simplicity makes it an obvious object for theoretical investigations: how can we hope to analyze more complex EDAs, if even the most structurally simple EDAs are not analyzed? Furthermore, we will see that simple algorithms and problems can give rise to highly non-trivial questions.

The cGA follows the general outline of EDAs by sampling search points according to a probability distribution, evaluating them, and updating the probability distribution with respect to the evaluation. One of the most important design decisions is the representation of the probability distribution. Obviously, we need exponential space in the problem dimension n to store a separate probability for every search point possible in a cartesian search space. Hence, we cannot model every possible probability distribution in polynomial space. Any designer of EDAs must therefore choose the right representation of the probability distribution that

1. can be stored in polynomial space in the problem dimension n and
2. can represent the probability distributions helping the EDA to find a sufficiently good one.

The cGA represents the probability distribution component-wise, i. e. for every dimension of the search space there is one probability distribution over the range of possible values of this component. We concentrate on pseudo-boolean objective functions $f : \{0,1\}^n \to \mathbb{R}$. In this case the cGA uses n probabilities $p_i \in [0,1]$ ($i \in \{1,\dots,n\}$), each one denoting the probability of a **1** in the ith dimension. Hence, a search point $x = (x_1,\dots,x_n) \in \{0,1\}^n$ has probability $P(x) := \left(\prod_{i \mid x_i=1} p_i\right) \cdot \left(\prod_{i \mid x_i=0}(1-p_i)\right)$. Hence, if the cGA is suited for some functions at all, it seems to be most suited to linear functions, because it cannot represent any dependencies between different dimensions.

Another detail to be specified is the mode of adjusting the probability distributions. The cGA samples two search points $x, y \in \{0,1\}^n$ are via the actual probability distribution, but otherwise independently. They are compared with respect to their objective function values $f(x)$ and $f(y)$ and for each component i the probability p_i is "pushed" towards the component of the better search point by a summand $1/K$. Hence, if the better search point has a **1** (resp. **0**) in the ith dimension and the worse one a **0** (resp. **1**), p_i is set to $p_i + 1/K$ (resp. $p_i - 1/K$). Notice, that the parameter K can depend on the problem dimension n, i. e. $K = K(n)$. In [4] the parameter K is fixed to n in order to simulate the simple GA with population size n. Since we also want to analyze the influence of K on the behavior of the cGA, we do not fix K. To prevent the p_is from getting smaller than 0 or larger than 1, K has to be an even integer. All in all, the cGA is defined as follows:

Algorithm 1. *The cGA with even $K \in \mathbb{N}^+$ for the maximization of $f : \{0,1\}^n \to \mathbb{R}$:*

```
1. Set $t := 1$ and $p_{1,t} := p_{2,t} := \cdots := p_{n,t} := 1/2$.
2. For all $i \in \{1,\dots,n\}$ set $x_i := 1$ with probability $p_{i,t}$
   and $x_i := 0$ otherwise.
3. For all $i \in \{1,\dots,n\}$ set $y_i := 1$ with probability $p_{i,t}$
   and $y_i := 0$ otherwise.
4. If $f(x) < f(y)$, swap $x$ and $y$.
5. For every $i \in \{1,\dots,n\}$:
   (a) If $x_i > y_i$, set $p_{i,t+1} := p_{i,t} + 1/K$.
   (b) If $x_i < y_i$, set $p_{i,t+1} := p_{i,t} - 1/K$.
   (c) If $x_i = y_i$, set $p_{i,t+1} := p_{i,t}$.
6. Set $t := t + 1$ and go to step 2.
```

The cGA is similar to the univariate marginal distribution algorithm (UMDA), see [8]. The UMDA samples in each step M search points and selects $N \leq M$ of these according to some selection method. The proportion of **1**s of all N search points at a position determines the probability of a **1** at this position in the next iteration. Hence, the UMDA is more closely related to classical population-based EAs than the cGA, which only samples two points in each iteration.

Notice, that in practice this infinite sampling-evaluation-adjustment-cycle is stopped according to some stopping criterion. Since we are interested in the number of steps necessary to find a sufficiently good distribution, we omit it.

This raises the question of the definition of a "sufficiently good" probability distribution. In practice, it would be sufficient, if the final probability distribution P_t gives the set of all optima a probability of $1/p(n)$, where $p(n)$ is some polynomial in n. Then a polynomial number of samples according to P_t would result in at least one optimum with high probability (the exact number of samples is dependent on the degree of p and the probability we are looking for).

Here we restrict ourselves to exact optimization and only consider functions with exactly one optimum. Therefore, we will define the runtime T_f of the cGA on an objective function $f : \{0,1\}^n \to \mathbb{R}$ as the number of steps, until the probability distribution assigns probability one to the optimum and probability zero to all other search points:

$$T_f := \min\{t \in \mathbb{N}^+ \cup \{\infty\} \mid P_t(opt(f)) = 1\}$$

(where $opt(f)$ is the optimum of f). Notice, that the random variable T_f takes values from the set $\mathbb{N}^+ \cup \{\infty\}$. An infinite runtime happens exactly in the case that any probability p_i becomes zero during the run of the cGA, while the optimum has a $\mathbf{1}$ in the ith dimension (or vice versa). Once, the value of $p_{i,t}$ is one or zero, it can never change again, since the other bit value will never be sampled again.

Hence, the runtime T_f will be infinite for most functions with some probability only dependent on K and n. Therefore, also the expected runtime $E(T_f)$ will be infinite, making it useless as a performance measure. Thus, instead of analyzing $E(T_f)$ we often look at the expected runtime $E^*(T_f)$ under the condition of a finite runtime:

$$E^*(T_f) = E(T_f \mid T_f < \infty).$$

Additionally, we will try to bound the probability distribution of T_f, especially the probability $P_f^* := \text{Prob}(T_f < \infty)$ of a finite runtime of the cGA on f. Notice, that a lower bound on $E^*(T_f)$ can be useful without any bound on P_f^*, since it tells us how long we have to wait even when the runtime is finite. On the other hand, any upper bound on $E^*(T_f)$ should be accompanied by a lower bound on P_f^*. Otherwise, we do not know the likelihood that our (small) upper bound on the runtime holds.

The runtime of classical EAs is most often defined as the number of evaluations until one of the optima is sampled for the first time (the so called first-hitting time, e. g. see [3]). The runtime $T_f + 1$ is obviously an upper bound on the first-hitting time, so asymptotical upper bounds on T_f also hold for the first-hitting time. On the other hand, sampling an optimum in the cGA is very unlikely, if the distribution is far from optimal. Hence, we conjecture that the following results also hold for the first-hitting time, but have to leave exact proofs for future research.

The only parameter of the cGA is K, the inverse strength of changing the probability distribution. The effect of K on the optimization process is twofold: a small K implies a large change of the probability distribution, which speeds up optimization, if the distribution is changed "into the right direction". On the other hand, a small K makes it more likely that a probability $p_{i,t}$ becomes zero, although the ith component of the optimum is $\mathbf{1}$ (or vice versa). Hence, the choice of K is a critical one and we will also analyze the influence of K on the cGA for linear functions in the following. Before that, we introduce in the next section a general measure very helpful for the analysis of the cGA.

3. A GENERAL CHARACTERISTIC OF THE CGA: THE SURPLUS

In this section we define and analyze a general measure helping us to analyze the progress of the cGA: the so-called *surplus*. First of all, we make the assumption, that the only optimum of the objective function $f : \{0,1\}^n \to \mathbb{R}$ is the all-ones bit-string $(\mathbf{1},\dots,\mathbf{1})$. Notice, that this is no limitation on the set of all pseudo-boolean functions with exactly one optimum: since the cGA treats the bits $\mathbf{0}$ and $\mathbf{1}$ symmetrically, it will behave exactly the same for f and f_a ($a \in \{0,1\}^n$), where f_a results from f by flipping the ith bit in its argument, if and only if $a_i = 1$. Hence, considering $f_{\overline{opt(f)}}$ instead of f, where $opt(f)$ is the optimum of f and $\overline{a}$ the bitwise complement of a, does not change the runtime of the cGA. Because the optimum of $f_{\overline{opt(f)}}$ is $(\mathbf{1},\dots,\mathbf{1})$, any bound for $f_{\overline{opt(f)}}$ also holds for the original function f.

Hence, the cGA has found the optimal distribution if and only if $p_1 = \cdots = p_n = 1$. During each iteration of the cGA there are n single steps, the sub-steps of step 5 of Algorithm 1, possibly changing the probabilities $p_{i,t}$. A single step of the cGA increasing the probability $p_{i,t}$ by $1/K$ is called a *success*. Analogously, we call a single step a *failure*, if a probability $p_{i,t}$ is decreased by $1/K$. Hence, every iteration comprises n single steps and it is important to estimate how many of these steps are successes resp. failures.

Since the cGA starts with $p_1 = \cdots = p_n = 1/2$, a necessary and sufficient condition for the cGA to find the optimal distribution is that the accumulated number of successes is by $nK/2$ higher than the accumulated number of failures. To simplify our notation we will call the difference between the number of successes and failures the *surplus*. Obviously, the surplus depends strongly on f deciding whether the search point with more $\mathbf{1}$s is the fitter one. Consequently, we should speak of the surplus with respect to f, but often omit f when the context is clear.

In order to prove a general lower bound on the expected runtime of the cGA, we have to find a measure upper bounding the surplus for any f. The surplus in one iteration is the number of successes minus the number of failures in this iteration. Since we assume that $(\mathbf{1},\dots,\mathbf{1})$ is the only optimum, the surplus in one iteration is the number of indices $i \in \{1,\dots,n\}$ where the fitter sampled search point has a $\mathbf{1}$ and the worse one a $\mathbf{0}$ (the number of successes) minus the number of indices, where it is contrary (the number of failures). In other words, the surplus is the number of probabilities p_i "going in the right direction" minus the number of p_i, that "go in the wrong direction". Obviously, a large positive surplus in every step guarantees a small runtime. By analyzing the distribution of the surplus, we can get bounds on the runtime of the cGA.

Let the random variable $X_i \in \{0,1\}$ (resp. $Y_i \in \{0,1\}$) denote the outcome of sampling the ith component of the first (resp. second) search point. Therefore, in case that the first (resp. second) sample is the fitter one, the surplus equals $X_1 + \cdots + X_n - (Y_1 + \cdots + Y_n)$ (resp. $Y_1 + \cdots + Y_n - (X_1 + \cdots + X_n)$). To be independent of the actual f, we take the maximum of these two values as an upper bound (which is just the absolute value):

DEFINITION 2. *Let $p_1,\dots,p_n \in [0,1]$ and the random variables $X_1,\dots,X_n$, $Y_1,\dots,Y_n \in \{0,1\}$ be defined by:*

$$Prob(X_i = a) := Prob(Y_i = a) := \begin{cases} p_i & \text{, if } a = 1, \\ 1 - p_i & \text{, if } a = 0. \end{cases}$$

Then $|X_1 + \cdots + X_n - (Y_1 + \dots Y_n)|$ is called the optimal surplus $S(p_1,\dots,p_n)$.

Because of the above argumentation we know that, regardless of f and the random samples x and y, the surplus of the cGA with probabilities $p_1,\dots,p_n$ is at most $S(p_1,\dots,p_n)$:

Lemma 3. *Let $p_1, \dots, p_n \in [0,1]$. The optimal surplus $S(p_1, \dots, p_n)$ stochastically dominates the surplus of the cGA with probabilities $p_1, \dots, p_n$ regardless of f, i. e. for all $t \in \{-n, \dots, n\}$ the probability that the surplus of the cGA in one iteration is at least t is at most $Prob(S(p_1, \dots, p_n) \geq t)$.*

Hence, it is of obvious interest to upper bound the optimal surplus in order to upper bound the surplus of the cGA:

Lemma 4. *Let $p_1, \dots, p_n \in [0,1]$ and $p := p_1(1-p_1) + \cdots + p_n(1-p_n)$. The expected optimal surplus $S(p_1, \dots, p_n)$ is at most $\sqrt{2p}$.*

Proof. Analyzing the expectation of the optimal surplus seems to be complicated, as it is defined as the absolute value of $Z := X_1 + \cdots + X_n - (Y_1 + \cdots + Y_n)$. But notice, that $E(|Z|)$ is at most $\sqrt{E(Z^2)}$. This follows from the convexity of the function x^2, i. e. for all $a_1, a_2 \in \mathbb{R}$ and $\alpha \in [0,1]$:

$$(\alpha \cdot a_1 + (1-\alpha) \cdot a_2)^2 \leq \alpha \cdot a_1^2 + (1-\alpha) \cdot a_2^2.$$

By iteration the above inequality also holds for $n \geq 2$ arguments $a_1, \dots, a_n \in \mathbb{R}$ with coefficients $\alpha_1, \dots, \alpha_n \in [0,1]$ with $\alpha_1 + \cdots + \alpha_n = 1$ (widely known as Jensen's inequality):

$$\left(\sum_{i=1}^{n} \alpha_i a_i\right)^2 \leq \sum_{i=1}^{n} \alpha_i \cdot a_i^2.$$

Defining $a_1, \dots, a_n$ as the different outcomes of $|Z|$ and α_i as $\text{Prob}(|Z| = a_i)$, the above inequality is equivalent to $E(|Z|)^2 \leq E(Z^2)$, since $|Z|^2 = Z^2$.

We can figure out $E(Z^2)$ easily by linearity of expectation:

$$\begin{aligned} E(Z^2) &= E\left(((X_1 - Y_1) + \cdots + (X_n - Y_n))^2\right) \\ &= E\left(\sum_{i,j=1}^{n} (X_i - Y_i) \cdot (X_j - Y_j)\right) \\ &= \sum_{i,j=1}^{n} E\left((X_i - Y_i) \cdot (X_j - Y_j)\right). \end{aligned}$$

Notice that for $i \neq j$ the random variable $(X_i - Y_i) \cdot (X_j - Y_j)$ is 1 resp. -1 with the same probability $2 \cdot p_i(1-p_i) \cdot p_j(1-p_j)$ and 0 otherwise. Hence, the expected value of $(X_i - Y_i) \cdot (X_j - Y_j)$ is zero and $E(Z^2)$ can be simplified to

$$E(Z^2) = \sum_{i=1}^{n} E\left((X_i - Y_i)^2\right) = \sum_{i=1}^{n} 2p_i(1-p_i) = 2p.$$

Thus, the expected value of $|Z|$ is at most $\sqrt{2p}$. □

Although the optimal surplus is an upper bound on the surplus of the cGA, we will see that there are objective functions, where the surplus of the cGA in one iteration equals the optimal surplus, i. e. the bound is tight. Hence, we also want to find a lower bound on the optimal surplus.

If there are no restrictions on the values of $p_1, \dots, p_n$ the best possible lower bound is zero (all p_i can be either zero or one). But often we will be in a situation, where we know the accumulated surplus so far, i. e. the value of $p_s := p_1 + \cdots + p_n$ and also have some constant lower bound $a > 0$ on the values of the p_i, i. e. $p_1, \dots, p_n \in [a, 1]$. In this situation we can state a lower bound on the minimal optimal surplus:

Lemma 5. *Let $a \in [0,1[$ be a constant and $p_1, \dots, p_n \in [a,1]$ with $p_s := p_1 + \cdots + p_n \leq n - k(1-a)$ for $k \in \mathbb{N}^+$. Then the expected optimal surplus is $\Omega\left(\sqrt{ka(1-a)}\right)$.*

Proof. To prove the desired lower bound on the expectation of the optimal surplus $S(p_1, \dots, p_n)$, we show that $\text{Prob}\left(S(p_1, \dots, p_n) \geq \sqrt{ka(1-a)}/2\right)$ is at least a constant $c > 0$. In order to do this, we describe the optimal surplus by an equivalent, but easier to analyze generating process: In the beginning all random variables X_i and Y_i are 0. In the first phase we choose each $i \in \{1, \dots, n\}$ independently with probability $2p_i(1-p_i)$. In the second phase we randomly choose for every index i chosen in the first phase with probability $1/2$, whether we set $X_i = 1$ or $Y_i = 1$. Hence, we have $\text{Prob}(X_i + Y_i = 1) = 2p_i(1-p_i)$ and the probability distribution of $|Z| := |X_1 + \cdots + X_n - (Y_1 + \cdots + Y_n)|$ generated by this process equals the probability distribution of the optimal surplus $S(p_1, \dots, p_n)$.

Using this two-phase process, it is obvious that the probability of $|Z|$ being at least some $z \geq 1$ only depends on $2p := 2p_1(1-p_1) + \cdots + 2p_n(1-p_n)$, the sum of the probabilities of choosing an index in the first phase. A larger value of $2p$ implies a higher probability of choosing at least z' (for any $z' \geq 1$) of the n indices in the first phase. The surplus generated in the second phase increases monotonously with the number of indices chosen in the first phase. Hence, the surplus $S(p_1, \dots, p_n)$ stochastically dominates $S(p'_1, \dots, p'_n)$, if $p_1(1-p_1) + \cdots + p_n(1-p_n) > p'_1(1-p'_1) + \cdots + p'_n(1-p'_n)$. Therefore, a lower bound on p allows to lower bound the optimal surplus. We claim that p becomes minimal for all $p_1, \dots, p_n \in [0,1]$ with fixed $p_s := p_1 + \cdots + p_n$, if $(p_1, \dots, p_n)$ is of the form $(a, \dots, a, 1, \dots, 1, x)$ with $x \in [a, 1]$ (where the order of the values is not important).

Assume that this does not hold, i. e. the term p becomes minimal for some $p_1, \dots, p_n$ with two values neither a nor 1. W. l. o. g. these two values are $p_1 < 1$ and $p_2 > a$ with $p_1 > p_2$. We want to show that increasing p_1 by an arbitrary $\varepsilon > 0$ and decreasing p_2 by the same ε decreases the value of p in contrast to our assumption that p is already minimal. The contribution of p_1 and p_2 in p is $p_1(1-p_1) + p_2(1-p_2) =: p^*$. Changing p_1 to $p_1 + \varepsilon$ and p_2 to $p_2 - \varepsilon$, the contribution of p_1 and p_2 in p changes to

$$\begin{aligned} &(p_1 + \varepsilon)(1 - p_1 - \varepsilon) + (p_2 - \varepsilon)(1 - p_2 + \varepsilon) \\ = \; &p_1(1-p_1) + [\varepsilon(1 - p_1 - \varepsilon) - p_1\varepsilon] \\ &+ p_2(1-p_2) + [-\varepsilon(1 - p_2 + \varepsilon) + p_2\varepsilon] \\ = \; &p^* + [\varepsilon(2p_2 - 2p_1 - 2\varepsilon)] < p^*. \end{aligned}$$

Hence, we can assume w. l. o. g., that $p_1 = \cdots = p_l = a$, $p_{l+1} = \cdots = p_{n-1} = 1$, and $p_n \in [a,1]$ imply a minimal value of $\text{Prob}(|Z| \geq z)$. Since $p_s \leq n - k(1-a)$, we know that $l \geq k$. Therefore, the expected number of chosen indices in the first phase is at least $2ka(1-a)$. Using Chernoff bounds (see [7]), it follows that the number of chosen indices in the first phase is at least $ka(1-a)$ with constant probability.

The second phase partitions the set of chosen indices randomly uniformly into two sets. If the number of chosen indices in the first phase is N, it is well-known that with constant probability the larger set has at least $\sqrt{N}/2$ more elements than the smaller set (because the largest binomial coefficient $\binom{N}{N/2}$ is at most $\varepsilon \cdot 2^N/\sqrt{N}$ for some constant $\varepsilon < 1$). Hence, there is a constant probability that the surplus in the second phase is at least $\sqrt{ka(1-a)}/2$, if the number of chosen indices in the first phase is at least $ka(1-a)$. As the latter has constant probability, the probability of the surplus being $\sqrt{ka(1-a)}/2$ is at least a constant. □

4. A GENERAL LOWER BOUND OF THE RUNTIME

Using the results of the last section we show in this section a general lower bound $K\sqrt{n/2}$ on the expected number $E^*(T_f)$ of steps the cGA needs to optimize any function f. The main idea of the proof is to show, that the expected optimal surplus cannot be larger than $\sqrt{n/2}$, while a surplus of $Kn/2$ is necessary to reach the optimum. Using a drift argument (see [5]) it follows that the expected number of iterations is at least $K\sqrt{n/2}$.

Theorem 6. *The expected runtime $E^*(T_f)$ of the cGA for any function $f : \{0,1\}^n \to \mathbb{R}$ is at least $K\sqrt{n/2}$.*

Proof. According to Lemma 4 the expected optimal surplus in one iteration is at most $\sqrt{2p}$ with $p := p_1(1-p_1) + \cdots + p_n(1-p_n)$. Because p is at most $n/4$ (for $p_1 = \cdots = p_n = 1/2$), the expected optimal surplus in one iteration of the cGA is at most $\sqrt{n/2}$.

Now we model a run of the cGA as follows: since the state of the cGA is completely determined by the probabilities $p_1, \ldots, p_n$, a run of the cGA can be represented by a Markoff chain $S_1, \ldots, S_T$ on the state set

$$\left\{0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{K-1}{K}, 1\right\}^n.$$

The runtime of the cGA is the smallest T^*, such that $S_{T^*} = (1,\ldots,1) := S_{opt}$, the sole optimum of the random walk. The drift method (see [5]) estimates the runtime of a random walk via a function V mapping states of the random walk to real numbers with $V(S_{opt}) = 0$. If c_{low} (resp. c_{up}) is a lower bound (resp. an upper bound) on $E(V(S_t) - V(S_{t+1}))$ for all $t \geq 0$, the expected runtime is at most $V(S_1)/c_{low}$ (resp. at least $V(S_1)/c_{up}$) for the initial state S_1.

Now we use the "missing probability to the optimum" as the distance measure, i. e. V maps a state $S = (p_1, \ldots, p_n)$ on $V(S) = n - (p_1 + \cdots + p_n)$. Since the initial state is $(1/2, \ldots, 1/2)$, we have $V(S_1) = n/2$. As the expected surplus in every iteration is at most $\sqrt{n/2}$ and every success increases the probability by $1/K$, the expected decrease $E(V(S_t) - V(S_{t+1}))$ is at most $\sqrt{n/2}/K$. Hence, the drift method implies that the expected runtime is at least

$$\frac{n/2}{\sqrt{n/2}/K} = K\sqrt{n/2}.$$

□

The upper bound on the optimal surplus of $\sqrt{n/2}$ seems to be unnecessarily rough, since it is only tight for $p_1 = \cdots = p_n = 1/2$. During the run of the cGA the probabilities $p_1, \ldots, p_n$ will on average increase and the optimal surplus of $\sqrt{2p}$ will diminish. This observation can be used to get a smaller upper bound on the surplus and therefore a larger lower bound on the expected runtime. Since it is of the same asymptotical order $\Omega(K\sqrt{n})$ (with the constant factor 1 instead of $\sqrt{1/2}$), we omit its proof for the sake of brevity.

The main reason for the lower bound of $K\sqrt{n/2}$ is the fact, that in any binomial distribution with constant success probabilities the number of successes varies from its expectation by $O(\sqrt{n})$ in the expected case. This also holds with probability super-polynomially close to one (i. e. $1 - o\left(\frac{1}{q(n)}\right)$ for any polynomial q), if we increase the range of variation by a factor of $\log(n)$.

In other words, we can show that in every iteration of the cGA the surplus is not larger than $\log(n)\sqrt{n}$ with probability super-polynomially close to one. Therby we can rule out polynomially many steps having a larger surplus and still have a super-polynomially small error probability. Since the surplus has to be larger than $nK/2$, this allows us to lower bound the number of steps by $nK/(2\log(n)\sqrt{n})$ with high probability, as long as K is polynomial:

Theorem 7. *With probability super-polynomially close to one the number T_f of steps of the cGA with parameter $K = poly(n)$ to optimize an arbitrary function $f : \{0,1\}^n \to \mathbb{R}$ is at least $K\sqrt{n}/(2\log(n))$.*

Proof. We show that the optimal surplus $S(p_1, \ldots, p_n)$ is at most $\log(n)\sqrt{n}$ with probability super-polynomially close to one. Using the above argumentation it follows that the cGA needs at least $\frac{nK/2}{\log(n)\sqrt{n}} = K\sqrt{n}/(2\log(n))$ steps with probability super-polynomially close to one for all K polynomial in n.

Assume that $S(p_1, \ldots, p_n) = |X_1 + \cdots + X_n - (Y_1 + \cdots + Y_n)| > \log(n)\sqrt{n}$. This implies that not both the number of positive $X_i - Y_i$ and the number of negative $X_i - Y_i$ can be in the interval $[p - \log(n)\sqrt{n}/2, p + \log(n)\sqrt{n}/2]$ (for $p := p_1(1-p_1) + \cdots + p_n(1-p_n)$). Hence, we can upper bound $\text{Prob}\left(S(p_1, \ldots, p_n) > \log(n)\sqrt{n}\right)$ by the probability that the number $\tilde{X}$ of $i \in \{1, \ldots, n\}$ with $X_i - Y_i = 1$ or the number $\hat{X}$ of of $i \in \{1, \ldots, n\}$ with $X_i - Y_i = -1$ is outside the interval $[p - \log(n)\sqrt{n}/2, p + \log(n)\sqrt{n}/2]$. Since $\tilde{X}$ and $\hat{X}$ are equally distributed we can upper bound $\text{Prob}\left(S(p_1, \ldots, p_n) > \log(n)\sqrt{n}\right)$ by:

$$\begin{aligned} & 2 \cdot \text{Prob}\left(\tilde{X} \notin \left[p - \frac{\log(n)\sqrt{n}}{2}, p + \frac{\log(n)\sqrt{n}}{2}\right]\right) \\ = \; & 2 \cdot \left(\text{Prob}\left(\tilde{X} < p\left(1 - \frac{\log(n)\sqrt{n}}{2p}\right)\right) + \right. \\ & \left. \text{Prob}\left(\tilde{X} > p\left(1 + \frac{\log(n)\sqrt{n}}{2p}\right)\right)\right) \; (*) \end{aligned}$$

If p is at most $\log(n)\sqrt{n}/2$, the last sum (*) is equal to $2 \cdot \text{Prob}\left(\tilde{X} > p\left(1 + \frac{\log(n)\sqrt{n}}{2p}\right)\right)$. Since the probability of $X_i - Y_i = 1$ is $p_i(1-p_i)$ the expected number $E(\tilde{X})$ of such indices i is p and we can use Chernoff bounds (see [7]) to upper bound the last probability by

$$2 \cdot \exp\left(-\frac{\log(n)\sqrt{n}}{2p} \cdot \frac{p}{3}\right) = 2 \cdot \exp\left(-\frac{\log(n)\sqrt{n}}{6}\right),$$

which is super-polynomially small.

If p is larger than $\log(n)\sqrt{n}/2$, then $\log(n)\sqrt{n}/2p$ is smaller than 1. Hence, we have to upper bound both terms of (*). Since they are of the form $\text{Prob}(\tilde{X} < p(1-\varepsilon))$ resp. $\text{Prob}(\tilde{X} > p(1+\varepsilon))$, we can upper bound each of them by $\exp(-\varepsilon^2 p/2)$ via Chernoff bounds (because $\varepsilon \leq 1$). Thus, we can upper bound $\text{Prob}\left(S(p_1, \ldots, p_n) > \log(n)\sqrt{n}\right)$ by

$$4\exp\left(-\left(\frac{\log(n)\sqrt{n}}{2p}\right)^2 \cdot \frac{p}{2}\right) = 4\exp\left(-\frac{\log(n)^2 n}{8p}\right).$$

Since p is at most $n/4$, we can upper bound this last expression by $4\exp\left(-\log(n)^2/2\right)$. Therefore, the probability of the surplus in one iteration being larger than $\log(n)\sqrt{n}$ is super-polynomially small for all p. □

Altogether, we now know that the runtime of the cGA is at least of order $K\sqrt{n}$ and that with high probability it can only be by a factor $\log(n)$ smaller. Nevertheless, this does not imply that a small K is always advantageous, since a small K increases the probability of an infinite runtime.

In the next section we analyze, if this bound is asymptotically tight, i. e. if there is a function f, so that $E^*(T_f)$ is of exact asymptotical order $K\sqrt{n}$.

5. ANALYSIS OF THE CGA ON ONEMAX

The lower bound on the runtime of the cGA in the last section depends heavily on the optimal surplus $S(p_1, \dots, p_n)$. To answer the question, if the lower bound of $K\sqrt{n}$ is asymptotically tight, we have to find a function whose surplus is close to the optimal surplus. Therefore, it is advisable to describe the surplus by an easier, yet equivalent process.

Let us consider linear functions $f(x_1, \dots, x_n) = \sum_{i=1}^{n} w_i x_i$ with coefficients $w_1, \dots, w_n \in \mathbb{R}$. Since the order of the bits is not relevant for the cGA, we can assume w. l. o. g. $w_1 \geq \dots w_n > 0$. Thus, the order of the search points x and y after step 4 of the cGA only depends on the bits, which are set differently in x and y. The surplus is exactly the number of positions, where the fitter search point has a **1** and the other one a **0** (the successes), minus the number of positions, where it is the other way round (the failures). Hence, we can describe the surplus of the cGA with respect to a linear f by a more structured process:

DEFINITION 8. *Let $p_1, \dots, p_n \in [0, 1]$ and $f : \{0, 1\}^n \to \mathbb{R}$ a linear function with coefficients $w_1 \geq \dots \geq w_n > 0$. Consider the following random process generating two subsets $A, B \subseteq \{1, \dots, n\}$ with $A \cap B = \emptyset$:*

1. *Choose each $i \in \{1, \dots, n\}$ with probability $2p_i(1-p_i)$.*
2. *Partition the set of chosen indices uniformly randomly into A and B.*
3. *If $\sum_{i\in A} w_i < \sum_{i\in B} w_i$, swap A and B.*

The surplus $S_f(p_1, \dots, p_n)$ of the cGA with respect to f and $p_1, \dots, p_n$ is defined as $|A| - |B|$.

The surplus in one iteration of the cGA can be negative, if the fitter of the two search points x and y contains more **0**s than **1**s. To exclude such a situation, which intuitively makes the optimization process slower, we choose an objective function counting the **1**s in its argument. This function, OneMax : $\{0, 1\}^n \to \mathbb{R}$ defined by

$$\text{OneMax}(x_1, \dots, x_n) := \sum_{i=1}^{n} x_i,$$

is the most simple non-trivial linear function and has been the starting point for many theoretical investigations of EAs (see [3]). When optimizing OneMax the number of successes is in every iteration of the cGA at least as high as the number of failures, since OneMax favors search points with more **1**s than **0**s. Moreover, the surplus $S_{\text{OneMax}}(p_1, \dots, p_n)$ of the cGA on OneMax equals the optimal surplus $S(p_1, \dots, p_n)$ (i. e. the probability distributions of both random variables are the same): the optimal surplus exactly counts the number of positions where the fitter search point with respect to OneMax has a **1**, but the less fit search point a **0**. Hence, the function OneMax is an obvious candidate for a function with expected runtime $O(\sqrt{n}K)$:

THEOREM 9. *The expected runtime $E^*(T_f)$ of the cGA with $K = n^{1/2+\varepsilon}$ ($\varepsilon > 0$ constant) for $f =$ OneMax is $O(\sqrt{n}K)$ and the probability of the runtime being $O(\sqrt{n}K)$ is at least 1/2, implying $P_f^* \geq 1/2$.*

PROOF. To upper bound the runtime we have to lower bound the surplus in each iteration. Lemma 5 can be helpful, if we can upper bound $p_s := p_1 + \dots + p_n$ and lower bound the values of the p_i. To get an upper bound on p_s we divide the run of the cGA into different phases. The ith phase ends as soon as p_s exceeds some bound p_i^* (see below). This upper bounds the value of p_s in the ith phase by p_i^*.

We assume that all p_i are at least 1/3 during the course of optimization (we will see later on how likely this assumption is). In this situation we can apply Lemma 5 with $a = 1/3$ stating that the expected optimal surplus is $\Omega(\sqrt{ka(1-a)})$ if $p_s := p_1 + \dots + p_n \leq n - k(1-a)$. Let us now divide the run of the cGA on OneMax into n phases (a technique often used in the analysis of EAs, see [13]), where the ith phase ($i \in \{1, \dots, n\}$) starts with $p_s = n/2 + (i-1)/2$ and is finished, when $p_s \geq n/2 + i/2$. Since the nth phase ends with the optimal distribution, the runtime of the cGA is the sum of the lengths of the n phases.

In the ith phase p_s is at most $n/2 + i/2$. Therefore, we can apply Lemma 5 with $k = \lfloor 3(n-i)/4 \rfloor$, because

$$\frac{n}{2} + \frac{i}{2} = n - \left(\frac{n}{2} - \frac{i}{2}\right) = n - \frac{2}{3} \cdot \left(\frac{3}{4}(n-i)\right).$$

Hence, the expected optimal surplus in the ith phase is $\Omega(\sqrt{n-i})$, i. e. the value of p_s increases in one iteration of the cGA in expectation by $\Omega(\sqrt{n-i}/K)$. Now we apply the drift method analogously to the proof of Theorem 6, implying that the expected length of the ith phase is

$$\frac{1/2}{\Omega(\sqrt{n/2 - i/2}/K)} = O\left(\frac{K}{\sqrt{n-i}}\right).$$

Hence, the sum of the lengths of the phases 1 to $n-1$ is

$$\begin{aligned}\sum_{i=1}^{n-1} O\left(\frac{K}{\sqrt{n-i}}\right) &= O(K) \cdot \left(\sum_{i=1}^{n-1} \frac{1}{\sqrt{i}}\right) \\ \leq \quad O(K) \int_0^{n-1} x^{-1/2} dx &= O(K) \frac{\sqrt{n-1}}{2} = O(K\sqrt{n}).\end{aligned}$$

We have to analyze the nth phase in more detail by subdividing it into $K/2$ subphases, where the jth subphase ($j \in \{0, \dots, K/2-1\}$) starts with $p_s = (n-1/2) + j/K$ and ends with $p_s \geq (n-1/2) + (j+1)/K$. Hence, each subphase consists of the iterations until the surplus increases by one. In the worst case only one p_i, w. l. o. g. p_n is smaller than one. In this case the probability of an success in the jth subphase is $2(1/2 + j/K)(1/2 - j/K)$. Therefore, the expected length of the jth subphase is $1/(2(1/2+j/K)(1/2-j/K))$, i. e. we can upper bound the length of the nth phase by

$$\begin{aligned}\sum_{j=0}^{K/2-1} \frac{1}{2(\frac{1}{2} + \frac{j}{K})(\frac{1}{2} - \frac{j}{K})} &\leq \sum_{j=0}^{K/2-1} \frac{1}{\frac{1}{2} - \frac{j}{K}} \\ = \quad \sum_{j=1}^{K/2} \frac{1}{j/K} &= O(K\ln(K)).\end{aligned}$$

Since $\ln(K) = O(\sqrt{n})$, the total sum of the lengths of all subphases is $O(K\sqrt{n})$. Hence, the Markoff bound (see [7]) states that there is a constant $c > 0$, such that the probability of the runtime being at most $cK\sqrt{n}$ is at least $2/3$, if all p_i are at least $1/3$ during the whole run.

In order to prove that this assumption is very likely, we argue as follows: If step 3 of the cGA would be omitted, the probabilities of a success and a failure at the ith position would be both $p_i(1-p_i)$. As step 3 favors the search point with more $\mathbf{1}$s (since the objective function is OneMax, of course), the probability of a success is for every position at least as high as the probability of a failure.

To make this argumentation formally correct, we notice that $K = n^{1/2+\varepsilon}$. Let us consider $N := cK\sqrt{n} = cn^{1+\varepsilon}$ iterations and one of the n positions, w. l. o. g. the first position. Since in every iteration a success in the first position is at least as likely as a failure, we can only overestimate the number of failures if we assume that both have the same probability. Using Chernoff bounds analogously to the proof of Theorem 7 it follows that in N iterations with probability super-polynomially close to one the number of failures can only be by at most $\log(N)\sqrt{N}$ larger than the number of successes. Since

$$\log(N)\sqrt{N} \leq n^{1/2+3\varepsilon/4}$$

for n large enough, the probability p_1 is at least $1/2 - n^{1/2+3\varepsilon/4}/K$ during all N iterations with probability super-polynomially close to one. For n large enough we can lower bound $1/2 - n^{1/2+3\varepsilon/4}/K$ by $1/3$. Therefore, all probabilities $p_1, \ldots, p_n$ are at least $1/3$ for $cK\sqrt{n}$ iterations with probability super-polynomially close to one. Consequently, the cGA finds the optimal distribution in $cK\sqrt{n}$ steps with constant probability $1/2$ for n large enough.

Considering $cK\sqrt{n}$ iterations of the cGA as a super-phase, we now know that in each super-phase the cGA finds the optimal distribution with probability at least $1/2$. Hence, the number of super-phases until the optimal distribution is found is geometrically distributed with success probability $1/2$, i. e. the expected number of super-phases until the optimal distribution is found is $2cK\sqrt{n}$ (because of lack of space we have to omit the proof that after each super-phase all p_i are at least $1/2$ with constant probability). $\square$

Hence, OneMax is one of the easiest functions for the cGA, if K is not too small. If K is too small, we cannot rule out the possibility that one of the p_i gets zero by unlucky chance. In the next section we prove that not all linear functions can be optimized by the cGA in expected time $O(K\sqrt{n})$.

6. A DIFFICULT LINEAR FUNCTION FOR THE CGA: BINVAL

Intuitively, OneMax is the easiest linear function possible. Nevertheless, the (1+1) EA optimizes every non-trivial linear function asymptotically as fast as OneMax ([3]). Hence, one could guess that the cGA optimizes every non-trivial linear function $f : \{0,1\} \to \mathbb{R}$ in expected time $O(K\sqrt{n})$. But this guess does not hold: the main reason for the fast optimization of OneMax by the cGA is the fact, that its surplus is equal to the optimal surplus. This is caused by every bit having the same influence on the function value, which guarantees that in every iteration the search point closer to the optimum $(\mathbf{1}, \ldots, \mathbf{1})$ is the fitter one.

Obviously, this does not hold for every linear function. One well-known counterexample is the so-called BinVal-function, mapping a bit-string $(x_1, \ldots, x_n)$ on the integer having the binary representation $(x_1, \ldots, x_n)$:

$$\text{BinVal}(x_1, \ldots, x_n) := \sum_{i=1}^{n} x_i \cdot 2^{n-i}.$$

Intuitively, BinVal is quite opposite to OneMax, since the coefficient 2^{n-i} of x_i is higher than the sum $\sum_{j=i+1}^{n} 2^{n-j} = 2^{n-i} - 1$ of all coefficients of the bits $x_{i+1}, \ldots, x_n$ “to the right of” x_i. Therefore, a search point $(x_1, \ldots, x_n)$ has a higher BinVal-value than a search point $(y_1, \ldots, y_n)$, if and only if for the smallest index i with $x_i \neq y_i$ we have $x_i = 1$.

Assume, that $\text{BinVal}(x) > \text{BinVal}(y)$, i. e. there is one i with $x_i > y_i$ and $x_j = y_j$ for all $j < i$, while the remaining bits $x_{i+1}, \ldots, x_n$ and $y_{i+1}, \ldots, y_n$ cannot influence the order of $\text{BinVal}(x)$ and $\text{BinVal}(y)$. Then the surplus on the “unimportant bits” $x_{i+1}, \ldots, x_n$ and $y_{i+1}, \ldots, y_n$ is in expectation zero, i. e. the surplus in one iteration cannot be much larger than one with high probability. We make this consideration formally exact:

Theorem 10. *For all constant $\varepsilon > 0$ the probability of the runtime of the cGA on* BinVal *being larger than $nK/(1+\varepsilon)$ is exponentially close (in K) to 1:*

$$\textit{Prob}\left(T_{\text{BinVal}} > \frac{nK}{1+\varepsilon}\right) \geq 1 - \exp\left(-\frac{\varepsilon^2}{4(1+\varepsilon)} \cdot K\right).$$

Proof. Assume, the runtime T_{BinVal} is at most $nK/(1+\varepsilon)$, i. e. $T := T_{\text{BinVal}} \leq nK/(1+\varepsilon)$ steps have led to an accumulated surplus of $nK/2$. According to Definition 8 the surplus $S_{\text{BinVal}}(p_1, \ldots, p_n)$ in one step is the absolute difference of the sizes of two sets A and B, where $\sum_{i \in A} 2^{n-i}$ is larger than $\sum_{i \in B} 2^{n-i}$ and every i has the same probability $p_i(1-p_i)$ of being in A resp. B.

Let A w. l. o. g. be the set with the larger sum. Hence, A contains the element with the smallest index in $A \cup B$, while all elements with larger indices are with the same probability in A resp. B. Let A_i $(i \in \{1, \ldots, T\})$ be the set A in the ith step, where we have deleted the element of $A \cup B$ with the smallest index. Analogously, let B_i be the set B in the ith step (where no element is deleted). Therefore, the surplus in the ith step is exactly $1 + |A_i| - |B_i|$.

In order to find the optimal distribution in $T \leq nK/(1+\varepsilon)$ steps, the set $A_1 \cup \cdots \cup A_T$ must contain $nK/2 - nK/(1+\varepsilon) = \varepsilon nK/(1+\varepsilon)$ more elements than $B_1 \cup \cdots \cup B_T$. We want to upper bound the probability of this event. By assuming that every element is chosen with probability 1 to be in A or B (according to Definition 8) and that $|A_1 \cup \cdots \cup A_t \cup B_1 \cup \cdots \cup B_T| = Tn$, we can only overestimate this probability.

These assumptions imply that $A_1 \cup \cdots \cup A_T$ is a uniformly chosen subset of a set of $Tn \leq n^2K/(1+\varepsilon)$ elements and $B_1 \cup \cdots \cup B_T$ is its complement. Again, we can only overestimate the probability by assuming that T equals $nK/(1+\varepsilon)$. Hence, $A_1 \cup \cdots \cup A_T$ contains $\varepsilon nK/(1+\varepsilon)$ more elements than $B_1 \cup \cdots \cup B_T$ if and only if

$$|A_1 \cup \cdots \cup A_T| \geq \frac{n^2K}{2(1+\varepsilon)} + \frac{\varepsilon nK}{2(1+\varepsilon)}.$$

Since the expected size of $A_1 \cup \cdots \cup A_T$ is $n^2K/(2(1+\varepsilon))$ and $|A_1 \cup \cdots \cup A_T|$ is the sum of $\{0,1\}$-valued random variables, we can use Chernoff bounds to compute:

$$\begin{aligned}
&\text{Prob}\left(|A_1 \cup \cdots \cup A_T| \geq \frac{n^2K}{2(1+\varepsilon)} + \frac{\varepsilon n K}{2(1+\varepsilon)}\right) \\
= \; &\text{Prob}\left(|A_1 \cup \cdots \cup A_T| \geq \frac{n^2K}{2(1+\varepsilon)} \cdot \left(1+\frac{\varepsilon}{n}\right)\right) \\
\leq \; &\exp\left(-\frac{\varepsilon^2}{n^2}\cdot\frac{n^2K}{2(1+\varepsilon)}\cdot\frac{1}{2}\right) = \exp\left(-\frac{\varepsilon^2}{4(1+\varepsilon)}\cdot K\right).
\end{aligned}$$

Hence, the runtime is larger than $nK/(1+\varepsilon)$ with probability exponentially close (in K) to 1. □

So, the runtime of the cGA on BinVal is of order $\Omega(nK)$ with high probability. For $K = n^{1+\varepsilon}$ with a constant $\varepsilon > 0$ we can show analogously to the proof of Theorem 9 that the expected runtime $E^*(T_{\text{BinVal}})$ is $O(nK)$ and that the runtime is of this growth order with constant probability. We have to omit this proof because of lack of space.

7. CONCLUSION

The cGA is a very simple EDA using probability distributions without dependencies between different components. Hence, its behavior on linear functions is of major interest, since we expect those to be the most appropriate functions for the cGA. In this paper we have presented a number of theoretical analyses of the cGA on linear functions besides a general lower bound of $\Omega(K\sqrt{n})$ on its expected runtime, which holds up to a factor of $\log(n)$ with high probability.

Our main result is the proof that some linear functions like OneMax are optimized by the cGA in the optimal expected runtime $O(K\sqrt{n})$, while others like BinVal need at least expected runtime $\Omega(Kn)$. Notice, that this is in compliance with experimental results in [4], who observed empirically that for fixed n both the average runtimes of the cGA on OneMax and BinVal seem to increase linearly in K, while the latter grows more rapidly. Here we have rigorously proven that these differences must occur and have also presented an easily comprehendible measure, the surplus, explaining why these differences occur. Furthermore, we have seen that methods used in the analysis of common population-based EAs, like the drift method or the partitioning into phases, can be applied to analyze the cGA.

While these results are interesting by their own, they can also serve as a first step towards the analysis of more complicated EDAs. Naturally, there is no easily applicable procedure how to generalize these results. Every EDA must be analyzed separately, but the presented analyses can give hints how to do this. The surplus, i. e. a very local progress towards the optimum, can probably be transferred to other EDAs. E. g., analyzing the UMDA would almost inevitably call for an analysis of the influence of the selection method on the surplus.

Nevertheless, there are a lot of open questions for the cGA: first of all, we are interested in a general upper bound on the expected runtime of the cGA for all linear functions. Knowing that BinVal is optimized in expected time $O(Kn)$, we strongly conjecture that this bound holds for all linear functions. Furthermore, the influence of K on the probability P_f^* of a finite runtime has to be investigated more closely: although we can guarantee a constant lower bound on P_{OneMax}^* for $K = n^{1/2+\varepsilon}$, we lack a proof, that an infinite runtime gets much more likely for smaller K.

Acknowledgements

I thank Jens Jägersküpper, Tobias Storch, Ingo Wegener, and Carsten Witt for valuable advice and discussions. Furthermore, I thank the anonymous referees for some useful remarks helping me to improve the paper.

8. REFERENCES

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, New York, NY, 1997.

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

[3] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1 + 1) EA. *Theoretical Computer Science*, (276):51–81, 2002.

[4] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 523 – 528, 1998.

[5] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, (3):21 – 35, 2004.

[6] P. Larranaga and J. A. Lozano. *Estimation of Distribution Algorithms*. Kluwer Academic Publisher, 2002.

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

[8] H. Mühlenbein and T. Mahnig. Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology*, (7):19 – 32, 1999.

[9] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In *Proceedings of PPSN IV*, pages 178 – 187, 1996.

[10] M. Pelikan, D. E. Goldberg, and E. Cantù-Paz. Bayesian optimization algorithm, population sizing, and time to convergence. In *Proceedings of GECCO 2000*, pages 275 – 282, 2000.

[11] G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač, 1997.

[12] I. Wegener. *Computational Complexity and EC*. Tutorial at GECCO 2003, 2004, and 2005.

[13] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In R.Sarker, X.Yao, and M.Mohammadian, editors, *Evolutionary Optimization*, pages 349 – 369. Kluwer, 2002.