Natural Computing (2006) 5:257–283
DOI: 10.1007/s11047-006-9001-0

A rigorous analysis of the compact genetic algorithm for linear functions*

STEFAN DROSTE
LS Informatik 2, Universität Dortmund, 44221 Dortmund, Germany
(E-mail: stefan.droste@udo.edu)

Abstract. Estimation of distribution algorithms (EDAs) solve an optimization problem heuristically by finding a probability distribution focused around its optima. Starting with the uniform distribution, points are sampled with respect to this distribution and the distribution is changed according to the function values 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 (cGA), for linear pseudo-Boolean functions on n variables. We prove a general lower bound for all functions and a general upper bound for all linear functions. Simple test functions show that not all linear functions are optimized in the same runtime by the cGA.

Key words: compact genetic algorithm, runtime, theoretical analysis

1. Introduction

Estimation of distribution algorithms (EDAs) are a class of randomized search heuristics that rely on a probability distribution to represent the information gained so far about the optimization problem to be solved. Initially the probability distribution is usually uniform over the search space of the optimization problem, since there is no knowledge about the problem at hand. Then search points are sampled according to the actual probability distribution, these search points are evaluated with respect to the problem, and the probability distribution is adjusted towards giving good search points a higher

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

probability. These steps are iterated until a stopping criterion holds (e.g. that a predefined number of steps have been executed). One hopes that the found probability distribution is concentrated sufficiently well around the optima, such that sampling according to the found distribution results in optima with high probability or that optima have already been found during the run of the EDA (see Larranaga and Lozano, 2002 for a broad introduction to the field of EDAs).

Notice that EDAs follow roughly the same principle as evolutionary algorithms (EAs), where new search points are created via mutation and crossover and selection chooses the fittest search points. Hence, the genetic operators used to generate new search points are of great importance for an efficient optimization by the EA. Instead of genetic operators, EDAs use a probability distribution over the search space to sample new search points, hopefully of higher fitness than the old ones.

Hence, the representation of the probability distribution is one of the most important decisions during the design of an EDA. 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.

In order to investigate the influence of the representation of the probability distribution on the performance of an EDA, it is advisable to start with a simple EDA using a simple representation to develop methods needed for the analysis of more complicated EDAs. In this paper we analyze the compact genetic algorithm (cGA) introduced in Harik et al. (1998), because it uses the very simple representation by a factorial distribution: for every component one probability is stored and the probability of a search point is just the product of the components' probabilities. Therefore, the cGA cannot represent any dependencies between the variables making it intuitively inappropriate for non-linear functions, where the interaction between different variables influences the function values. Hence, we concentrate in this paper on the analysis of the cGA for linear functions: can the cGA optimize any linear function efficiently with respect to the problem's dimensionality? Are all linear functions optimized in the same runtime or are there ''difficult'' and ''easy'' linear functions for the cGA?

Our approach is a rigorous one, i.e. we want to prove properties of the cGA without any assumptions or simplifications, whose errors are not bounded. Rigorous runtime analysis is well established in the theory of EAs (see Beyer et al., 2002). 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 Wegener, 2002). The rigorous mathematical analysis of EAs has started with simple mutation-based EAs and test problems (e.g., see Rudolph, 1997 or Droste et al., 2002) and nowadays progresses towards more realistic EAs for combinatorial optimization problems (see Wegener, 2005 for an overview). Certainly, a similar development would be most beneficial for the field of EDAs.

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 Motwani and Raghavan, 1995 for the used notation of asymptotical growth rates), where K is an inverse measure of the maximal change of the probability distribution per iteration. In Section 5 it is shown that the expected runtime for ONEMAX is exactly of this order. Section 6 presents an upper bound of $O(KN)$ on the expected runtime for any linear function, while we prove in Section 7 that the runtime of the cGA on BINVAL is exactly of this order. These bounds prove that the runtime of the cGA on linear functions vary between $\Theta(K\sqrt{n})$ and $\Theta(Kn)$. The paper ends with some conclusions and open questions. Note, that this is an extended version of Droste (2005) with revised proofs and a new main result, the upper bound for all linear functions.

2. The compact genetic algorithm

The cGA introduced in Harik et al. (1998) 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.

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 $p_i \in [0,1]$ $(i \in \{1,\ldots,n\})$, each one denoting the probability of a **1** in the ith dimension. Thus, a search point $x = (x_1,\ldots,x_n) \in \{0,1\}^n$ has probability $P(x) := \left(\prod_{i|x_i=1} p_i\right) \cdot \left(\prod_{i|x_i=0}(1-p_i)\right)$ to be sampled. Hence, if the cGA is suited for some functions at all, it seems to be most suited for linear functions, because it cannot represent any dependencies between different dimensions.

In every iteration the cGA samples two search points $x,y \in \{0,1\}^n$ 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 Harik et al. (1998) 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 positive 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 $i \in \{1,\ldots,n\}$ set x_i:= 1 with probability $p_{i,t}$ and x_i:= 0 otherwise.
3. For all $i \in \{1,\ldots,n\}$ 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 $i \in \{1,...,n\}$:
   (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 Mühlenbein and Mahnig (1999). 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 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 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:

Definition 1. The runtime of the cGA on a function $f: \{0,1\}^n \to \mathbb{R}$ is defined as

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

where $\text{opt}(f)$ is the optimum of f and P_t is the probability distribution of the tth step of the cGA on 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 **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:

$$E^*(T_f) = \mathrm{E}(T_f | T_f < \infty).$$

Additionally, we will try to bound the probability distribution of T_f, especially the probability $P_f^* := \mathrm{Prob}(T_f < \infty)$ 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 Droste et al., 2002). 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 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 **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

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 (**1**,...,**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 **0** and **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 $f_{\overline{\text{opt}(f)}}$ instead of f, where opt(f) is the optimum of f and $\bar{a}$ the bitwise complement of a, does not change the runtime of the cGA. Because the optimum of $f_{\overline{\text{opt}(f)}}$ is (**1**,...,**1**), any bound for $f_{\overline{\text{opt}(f)}}$ 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 2, 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 call the difference $S_f(p_1,...,p_n)$ between the number of successes and failures the *surplus*. Obviously, the surplus depends strongly on f deciding whether the search point with more **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. Since (**1**,...,**1**) is the only optimum, the surplus in one iteration is the number of indices $i \in \{1,...,n\}$ where the fitter sampled search point has a **1** and the worse one a **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". Intuitively, a large positive surplus in every step guarantees a small runtime. By analyzing the distribution of the surplus, we can bound 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* $p_1, \ldots, p_n \in [0,1]$ *and the random variables* $X_1, \ldots, X_n, Y_1, \ldots, Y_n \in \{0,1\}$ *be defined by*:

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

Then $|X_1 + \cdots + X_n - (Y_1 + \cdots + Y_n)|$ *is called the optimal surplus* $S_{\text{opt}}(p_1,...,p_n)$.

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 $p_1,...,p_n$ is at most $S_{\text{opt}}(p_1,...,p_n)$:

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

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* $p_1,...,p_n \in [0,1]$ *and* $p := p_1(1-p_1) + \cdots + p_n(1-p_n)$. *The expected optimal surplus* $E(S_{\text{opt}}(p_1,...,p_n))$ *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 variance of $|Z|$, which is at least zero:

$$\begin{aligned} \text{Var}(|Z|) &= E(|Z|^2) - |E(|Z|)|^2 \\ &= E(Z^2) - E(|Z|)^2 \geq 0 \end{aligned}$$

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

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

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

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

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 $p_1,...,p_n$ the best possible lower bound is zero (all p_i can be either zero or one). But often we are 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. $p_1,...,p_n \in [a,1]$. 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 $p_1, \ldots, p_n \in [a,1]$ with $p_s := p_1 + \cdots + p_n \leq n - k(1-a)$ for $k \in \mathbb{N}^+$. Then the expected optimal surplus $S_{\text{opt}}(p_1, \ldots, p_n)$ is $\Omega\big(\sqrt{ka(1-a)}\big)$.*

Proof. To prove the desired lower bound on the expectation of the optimal surplus $S_{\text{opt}}(p_1, \ldots, p_n)$, we show that $\text{Prob}\big(S_{\text{opt}}(p_1, \ldots, p_n) \geq \sqrt{ka(1-a)}/2\big)$ 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 $i \in \{1, \ldots, n\}$ 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 $S(p_1,...,p_n)$.

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 $S(p_1,...,p_n)$ stochastically dominates $S(p'_1,...,p'_n)$, 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 $p_1,...,p_n \in [0,1]$ with fixed $p_s := p_1+\cdots+p_n$, if $(p_1,...,p_n)$ is of the form $(a,...,a,1,...,1,x)$ 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 $p_1,...,p_n$ 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

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

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 Motwani and Raghavan, 1995), 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 also 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 He and Yao, 2004) 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 $p_1,...,p_n$, a run of the cGA can be represented by a Markoff chain $S_1,...,S_T$ 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 $S_{T^*} = (1,\ldots,1) := x_{\text{opt}}$, the sole optimum of the random walk. The drift method (see He and Yao, 2004) estimates the runtime of a random walk via a function V mapping states of the random walk to real numbers with $V(x_{\text{opt}}) = 0$. 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 $V(S_1)/c_{\text{low}}$ (resp. at least $V(S_1)/c_{\text{up}}$) 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 $S=(p_1,...,p_n)$ on $V(S) = n-(p_1+\cdots+p_n)$. Since the initial state is $(1/2,...,1/2)$, 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

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

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 $p_1,...,p_n$ 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. $1-o(1/q(n))$ 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. Thereby 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 $K=\text{poly}(n)$ 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 $S_{\text{opt}}(p_1,...,p_n)$ 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 $S_{\text{opt}}(p_1,\ldots,p_n) = |X_1 + \cdots + X_n - (Y_1 + \cdots + Y_n)| > \log(n)\sqrt{n}$. 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 $\text{Prob}\big(S_{\text{opt}}(p_1,\ldots,p_n) > \log(n)\sqrt{n}\big)$ by the probability that the number $\tilde{X}$ of $i \in \{1,...,n\}$ with $X_i - Y_i = 1$ or the number $\hat{X}$ of $i \in \{1,...,n\}$ 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 $\text{Prob}\big(S_{\text{opt}}(p_1,\ldots,p_n) > \log(n)\sqrt{n}\big)$ by:

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

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 Motwani and Raghavan, 1995) 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 $\varepsilon \le 1$). Thus, we can upper bound $\text{Prob}\big(S_{\text{opt}}(p_1,\ldots,p_n) > \log(n)\sqrt{n}\big)$ 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.

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 $S_{\text{opt}}(p_1,...,p_n)$. 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.

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 (remember, that w.l.o.g. the all-ones bit-string (**1**,...,**1**) is the optimum of f). 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

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

is the most simple non-trivial linear function and has been the starting point for many theoretical investigations of EAs (see Rudolph, 1997 or Droste et al., 2002). 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 of the cGA on OneMax equals the optimal surplus $S_{\text{opt}}(p_1,...,p_n)$ (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 8. *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 $p_s := p_1 + \cdots + p_n$ 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 $p_s := p_1 + \cdots + p_n \leq n - k(1-a)$. Let us now divide the run of the cGA on OneMax into n phases (a technique often used in the analysis of EAs, see Wegener, 2002), where the ith phase ($i \in \{1,...,n\}$) starts with $p_s = n/3 + 2(i-1)/3$ and is finished, when $p_s \geq n/3 + 2i/3$. 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 $n/3 + 2i/3$. Therefore, we can apply Lemma 5 with $k = (n-i)$, because

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

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

$$\frac{2/3}{\Omega(\sqrt{n-i}/K)} = O\left(\frac{K}{\sqrt{n-i}}\right).$$

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

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

We have to analyze the nth phase in more detail by subdividing it into $2K/3$ subphases, where the jth subphase ($j \in \{0,...,2K/3-1\}$) starts with $p_s = (n-2/3)+j/K$ and ends with $p_s \geq (n-2/3)+(j+1)/K$. 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 $2(1/3+j/K)(2/3-j/K)$. Therefore, the expected length of the jth subphase is $1/(2(1/3+j/K)(2/3-j/K))$, i.e. we can upper bound the length of the nth phase by

$$\sum_{j=0}^{2K/3-1} \frac{1}{2\left(\frac{1}{3}+\frac{j}{K}\right)\left(\frac{2}{3}-\frac{j}{K}\right)} \leq \sum_{j=0}^{2K/3-1} \frac{3/2}{\frac{2}{3}-\frac{j}{K}} = \sum_{j=1}^{2K/3} \frac{3/2}{j/K} = O(K\ln(K)).$$

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 Motwani and Raghavan, 1995) 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 **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 $1/2 - n^{1/2+3\varepsilon/4}/K = 1/2 - n^{-\varepsilon/4}$ during all N iterations with probability super-polynomially close to one. For n large enough we can lower bound $1/2 - n^{-\varepsilon/4}$ by $1/3$.

Therefore, all probabilities $p_1,\ldots,p_n$ are at least $1/3$ for the first $cK\sqrt{n}$ iterations with probability super-polynomially close to one. Consequently, the cGA finds the optimal distribution in the first $cK\sqrt{n}$ iterations with constant probability $1/2$ for n large enough. This does not hold only for the first $cK\sqrt{n}$ iterations (a so-called *super-phase*): since all probabilities decrease only by $n^{-\varepsilon/4}$ in one super-phase with probability super-polynomially close to one, it holds for the first $n^{\varepsilon/4-\delta}$ super-phases for any constant $\delta > 0$, e.g. for $n^{\varepsilon/5}$ super-phases.

We now know that in each of the first $n^{\varepsilon/5}$ super-phases the cGA finds the optimal distribution with probability at least $1/2$. Although we cannot lower bound the success probabilities of the remaining super-phases in the same way, the probability of all the first $n^{\varepsilon/5}$ super-phases not to be successful is at most $(1/2)^{n^{\varepsilon/5}}$, i.e. exponentially small in n. The expected runtime of the cGA on OneMax is regardless of the starting conditions polynomially in n (if K is polynomial in n): this holds, since any run of the cGA can be seen as a random walk on a state space of size nK, where the probability of a "good" step is at least as high as that of a "bad" step. As the probability of a step in an iteration is at least $\Omega(1/K)$, the expected runtime is polynomial in n.

Hence, assuming that every super-phase has a success probability of at least $1/2$ introduces only an error super-polynomially small in the estimation of the expected number of super-phases. Therefore, we can assume that 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 steps until the optimal distribution is found is $2cK\sqrt{n}$. □

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.

6. A general upper bound for linear functions

We have shown that the expected runtime of the cGA on any function is of order $\Omega(K\sqrt{n})$ and that there is a simple linear function, OneMax, whose expected runtime is exactly of this order. Intuitively, linear functions are the simplest functions for the cGA, since they cannot contain any dependencies between variables, the cGA is not capable to represent anyway. Hence, the runtime of the cGA on linear functions is of special interest: if linear functions cannot be optimized efficiently, we cannot expect other, more complicated functions to be optimized by the cGA efficiently.

Linear pseudo-Boolean functions f are of the form $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_i x_i$ with coefficients $w_1, \ldots, w_n \in \mathbb{R}$. Since the order of the bits is not relevant for the cGA and **0**s and **1**s can be swapped, we can assume w.l.o.g. $w_1 \geq \cdots \geq w_n > 0$. 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 9. *Let $p_1,\ldots,p_n \in [0,1]$ and $f : \{0,1\}^n \to \mathbb{R}$ a linear function with coefficients $w_1 \geq \cdots \geq w_n > 0$. Consider the following random process generating two subsets $A, B \subseteq \{1, \ldots, n\}$:*

1. *Choose each $i \in \{1,\ldots,n\}$ with probability $2p_i(1-p_i)$.*
2. *Partition the chosen set uniformly randomly into A and B. The surplus $S_f(p_1,\ldots,p_n)$ of the cGA with respect to f and $p_1,\ldots,p_n$ is defined as $|A|-|B|$ resp. $|B|-|A|$, if $w(A) \geq w(B)$ resp. $w(A) < w(B)$, where the weight of $M \subseteq \{1, \ldots, n\}$ is defined by $w(M) := \sum_{i \in M} w_i$.*

Hence, A resp. B contains the indices i with $x_i = 1$ and $y_i = 0$ resp. $x_i = 0$ and $y_i = 1$. Often we will shorten the description of the surplus to S_f, if the parameters $p_1,\ldots,p_n$ are obvious from the context.

We will show in the following that the expected surplus of the cGA for any linear function is at least 1, if we only consider the steps of the cGA with $w(A) \neq w(B)$. Since the probability of $w(A) \neq w(B)$ is constant, if the probabilities p_i are not too close to 0 or 1, we can show the expected surplus to be at least a constant for every linear function.

The essential idea of the proof is to show that the set with the larger weight (of the random process defined in Definition 9) contains at

least one more element than the smaller set in the expected case. This is trivial for $n=1$ and can be shown by induction to hold for $n>1$:

Lemma 10. Let $a \in]0,1[$, $p_1,...,p_n \in [a,1[$, $c \in \mathbb{R}$ and $f: \{0,1\}^n \to \mathbb{R}$ a linear function with coefficients $w_1 \geq w_2 \geq \cdots \geq w_n > 0$. If $k \in \mathbb{N}$ is the smallest number, such that there exists a subset $\{j_1, \ldots, j_k\} \subseteq \{1, \ldots, n\}$ with $\sum_{i=1}^{k} w_{j_i} \geq c$, then

$$E(S_f(p_1, \ldots, p_n) | w(A) \geq w(B) + c) \geq k.$$

If there is no k fulfilling the condition, i.e. $\sum_{i=1}^{n} w_i < c$, the conditional expected value is zero.

Proof. The condition of the lemma implies that A contains at least k elements (w.l.o.g. $\{1,...,k\}$), because otherwise $w(A)$ cannot be at least $w(B)+c \geq c$. If there is no subset of $\{w_1,...,w_n\}$ whose sum is at least c, the condition $w(A) \geq w(B)+c$ has probability zero. Hence, the conditional expected value is zero in this case.

We prove the statement of the lemma by induction over $n \geq 1$. For $n=1$ we distinguish two cases: if $c \leq 0$ the smallest subset fulfilling the requirement is empty, i.e. $k=0$. Hence, the lemma states in this case that the expected conditional surplus is non-negative. This is true, since the unconditional expected surplus is zero and lower bounding the weight of A can only increase the surplus, since such a condition rules out more cases where $|A| < |B|$ than vice versa.

If $0 < c \leq w_1$ the condition $w(A) \geq w(B)+c$ enforces $A=\{1\}$ and $B=\emptyset$. Since this event has positive probability for $p_1 \in]0,1[$, the conditional expected value is $k=1$.

Let us now assume that the induction statement holds for n to prove it for $n+1$. We want to show that

$$E(S_f(p_1, \ldots, p_{n+1}) | w(A) \geq w(B) + c) \geq k$$

for the smallest k with $w_1 + \cdots + w_k \geq c$. We distinguish three cases with respect to the choice of index 1. Let A' and B' be the sets consisting of the remaining elements of $\{2,...,n+1\}$, i.e. $A' = A \cap \{2,...,n+1\}$ and $B' = B \cap \{2,...,n+1\}$. If element 1 is chosen to be in the set A, we have $A = A' \cup \{1\}$ and $B = B'$. This implies

$$w(A) \geq w(B) + c \Leftrightarrow w(A') \geq w(B') + (c - w_1).$$

In order to compensate the summand $c-w_1$ at least $k-1$ elements are necessary. Hence, the induction hypothesis implies that A' contains at least $k-1$ elements more than B', i.e. A contains at least k elements more than B in expectation.

The same argumentation for the case $1 \in B$ leads to

$$w(A) \geq w(B) + c \Leftrightarrow w(A') \geq w(B') + (c + w_1).$$

In this case we can use the induction hypothesis with $k+1$, since at least $k+1$ elements are needed to form the sum $c+w_1$. Therefore, A' has to contain at least $k+1$ elements more than B' in expectation. Together with the first element 1, A has to contain at least k elements more than B in expectation.

If 1 is neither in A nor in B we can use the induction hypothesis directly, stating that A' must contain at least k elements more than B' in expectation.

Combining the three cases, we get:

$$\begin{aligned}
&\mathrm{E}(S_f(p_1, \ldots, p_{n+1}) | w(A) \geq w(B) + c) \\
=&\mathrm{Prob}(1 \in A | w(A) \geq w(B) + c) \cdot \\
&\quad E(S_f(p_2, \ldots, p_{n+1}) + 1 | w(A) \geq w(B) + c, 1 \in A) \\
&+ \mathrm{Prob}(1 \in B | w(A) \geq w(B) + c) \cdot \\
&\quad E(S_f(p_2, \ldots, p_{n+1}) - 1 | w(A) \geq w(B) + c, 1 \in B) \\
&+ \mathrm{Prob}(1 \notin A \cup B | w(A) \geq w(B) + c) \cdot \\
&\quad E(S_f(p_2, \ldots, p_{n+1}) | w(A) \geq w(B)) \\
\geq&\mathrm{Prob}(1 \in A | w(A) \geq w(B) + c) \cdot ((k-1) + 1) \\
&+ \mathrm{Prob}(1 \in B | w(A) \geq w(B) + c) \cdot ((k+1) - 1) \\
&+ \mathrm{Prob}(1 \notin A \cup B | w(A) \geq w(B) + c) \cdot k = k
\end{aligned}$$ □

Applying Lemma 10 for $c = 1$ results in

$$\mathrm{Prob}(S_f | w(A) > w(B)) = \mathrm{Prob}(S_f | w(A) \geq w(B) + 1) \geq 1.$$

Since the expected surplus is zero, if we only consider the cases with $w(A) = w(B)$, we have because of symmetry:

$$\begin{aligned}E(S_f) &= \text{Prob}(w(A) > w(B)) \cdot E(|A| - |B||w(A) > w(B)) \\ &\quad + \text{Prob}(w(B) > w(A)) \cdot E(|B| - |A||w(B) > w(A)) \\ &\quad + \text{Prob}(w(A) = w(B)) \cdot E(|A| - |B||w(A) = w(B)) \\ &= 2 \cdot \text{Prob}(w(A) > w(B)) \cdot E(|A| - |B||w(A) > w(B)) \\ &\geq 2 \cdot \text{Prob}(w(A) > w(B)).\end{aligned}$$

Therefore, all that is left to do is to lower bound the probability of $w(A) > w(B)$ by a constant. In order to do this, it is sufficient to lower bound the probability of $A \cup B \neq \emptyset$ by a constant:

Notice that two sets A and B with $w(A) = w(B)$ can be converted to sets $\tilde{A}$ and $\tilde{B}$ with $w(\tilde{A}) \neq w(\tilde{B})$ by swapping the smallest index in $A \cup B$ to the other set. This mapping is injective and the pair $(\tilde{A}, \tilde{B})$ has the same probability as (A,B). Hence, the probability of $w(A) \neq w(B)$ is at least the probability of $w(A) = w(B)$. Because of symmetry $w(A) > w(B)$ has the same probability as $w(A) < w(B)$. All in all, $\text{Prob}(A \cup B \neq \emptyset)$ is

$$\begin{aligned}&\text{Prob}(w(A) \neq w(B), A \cup B \neq \emptyset) + \text{Prob}(w(A) = w(B), A \cup B \neq \emptyset) \\ &\leq 2\text{Prob}(w(A) \neq w(B), A \cup B \neq \emptyset) \leq 4\text{Prob}(w(A) > w(B))\end{aligned}$$

Therefore, a constant lower bound on $\text{Prob}(A \cup B \neq \emptyset)$ implies a constant lower bound on $\text{Prob}(w(A) > w(B))$.

In general, a constant lower bound for $A \cup B \neq \emptyset$, i. e. the event that the cGA samples two different search points x and y, is hard to achieve, since all probabilities p_i can be close to zero or one. Thus, we divide the run of the cGA into phases similar to the proof of Theorem 8 to prove the upper bound of $O(nK)$ on the expected runtime of the cGA:

Theorem 11. *Let $f: \{0,1\}^n \to \mathbb{R}$ be a linear function. The expected runtime $E^*(T_f)$ of the cGA with $K = n^{1+\varepsilon}(\varepsilon > 0$ constant) for f is $O(nK)$ and the probability of the runtime being $O(nK)$ is at least $1/2$, implying $P^*_f \geq 1/2$.*

Proof. Following the argumentation made in the proof of Theorem 8 we assume that all probabilities $p_1,...,p_n$ are at least $1/3$. We divide the run of the cGA in n phases, where the ith phase ends, if $p_1 + \cdots + p_n \geq n/3 + 2i/3$ (notice that some phases may be initially empty, since the cGA starts with $p_1 + \cdots + p_n = n/2$). Hence, the

expected runtime of the cGA is the sum of the expected lengths of the n phases.

In the ith phase ($i \in \{1,...,n\}$) the sum $p_1 + \cdots + p_n$ is at least $n/3 + 2(i-1)/3$. Given some probabilities $p_1,...,p_n \in [1/3,1]$ with fixed sum $p_s = p_1 + \cdots + p_n$, the probability of $x \neq y$ becomes minimal, if all p_i are either 1/3 or 1. This follows from the fact, that the probability

$$2p_i(1 - p_i) = 2p_i - 2p_i^2$$

of $x_i \neq y_i$ is maximal for $p_i = 1/2$ and decreases monotonously when increasing resp. decreasing p_i. Since the probability of $x \neq y$ is

$$1 - \prod_{i=1}^{n} \text{Prob}(x_i = y_i) = 1 - \prod_{i=1}^{n} (1 - \text{Prob}(x_i \neq y_i)),$$

and $\text{Prob}(x_i \neq y_i)$ decreases increasingly strongly with an increasing distance of p_i to 1/2 (as the second derivative of $\text{Prob}(x_i \neq y_i)$ is -4), $\text{Prob}(x \neq y)$ becomes minimal for $(p_1,...,p_n)$, where at most one probability p_i is neither 1/3 nor 1.

Hence, we can assume pessimistically that in the ith phase $p_1 = \cdots = p_{i-1} = 1$, $p_i \in [1/3,1]$, and $p_{i+1} = \cdots = p_n = 1/3$. The probability of $x \neq y$ is

$$1 - \prod_{i=1}^{n} \text{Prob}(x_i = y_i) \geq 1 - \left(\frac{1}{3} \cdot \frac{1}{3} + \frac{2}{3} \cdot \frac{2}{3}\right)^{n-i} = 1 - (5/9)^{n-i},$$

which is at least 4/9 for all $i < n$.

Therefore, the expected surplus per iteration in all phases except the last phase is constant. Applying the drift method analogously to the proof of Theorem 6 we can upper bound the expected length of each of these phases by $O(K)$.

The nth phase has to be analyzed in more detail by subdividing it into $2K/3$ subphases, where the jth subphase ($j \in \{0,...,2K/3-1\}$) starts with $p_s = (n-2/3) + j/K$ and ends with $p_s \geq (n-2/3) + (j+1)/K$. 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 $2(1/3 + j/K)(2/3 - j/K)$. Therefore, the expected length of the jth subphase is $1/(2(1/3 + j/K)(2/3 - j/K))$, i.e. we can upper bound the length of the nth phase by

$$\sum_{j=0}^{2K/3-1} \frac{1}{2\left(\frac{1}{3}+\frac{j}{K}\right)\left(\frac{2}{3}-\frac{j}{K}\right)} \leq \sum_{j=0}^{2K/3-1} \frac{3/2}{\frac{2}{3}-\frac{j}{K}} = \sum_{j=1}^{2K/3} \frac{3/2}{j/K} = O(K\ln(K)).$$

Since $\ln(K) = o(n)$, the total sum of the lengths of all subphases is $O(Kn)$. Hence, the Markoff bound (see Motwani and Raghavan, 1995) states that there is a constant $c > 0$, such that the probability of the runtime being at most cKn is at least 2/3, if all p_i are at least 1/3 during the whole run.

Now we can argue in the same way as in the proof of Theorem 8, that the probabilities p_i are at least 1/3 for a sufficient number of steps: the essential difference to the proof of Theorem 8 is that K equals $n^{1+\varepsilon}$ for a constant $\varepsilon > 0$. Hence, a super-phase now consists of $N := Kn = n^{2+\varepsilon}$ steps. Again, Chernoff bounds imply that the number of failures can only be by at most $\sqrt{N}\log(N) \leq n^{1+3\varepsilon/4}$ larger than the number of successes in a super-phase with probability super-polynomially close to one. Since each failure decreases a probability p_i by $1/K$, the sum p_s of probabilities decreases by at most $n^{-\varepsilon/4}$ in one super-phase with high probability (notice that this would not hold for $K = n^{1/2+\varepsilon}$ as it was assumed in the upper bound for OneMax). Therefore, we are in the same situation as in the proof of Theorem 8 and can follow the same arguments to show that with probability at least 1/2 all p_i are at least 1/3 in the first $n^{\varepsilon/5}$ super-phases. This implies that the cGA finds the optimal distribution in cKn steps with probability at least 1/2 and that the expected number of steps is $O(Kn)$, completing the proof. □

Now we know that all linear functions are optimized by the cGA in $O(Kn)$ expected steps, if $K = n^{1+\varepsilon}$. Note, that this condition is needed to rule out, that some p_i could become zero by unlucky chance during the $O(Kn)$ steps.

7. A difficult linear function for the cGA: BinVal

The previous sections tell us, that the expected runtime of the cGA on every linear function is $O(nK)$, while the runtime for OneMax is only $\Theta(K\sqrt{n})$. Intuitively, OneMax is the easiest linear function possible, opening the possibility that there are some linear functions essentially more difficult for the cGA.

Nevertheless, the (1 + 1) EA optimizes every non-trivial linear function asymptotically as fast as OneMax (see Droste et al., 2002).

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 (**1**,...,**1**) is the fitter one.

The proof of the expected runtime $O(nK)$ bases on lower bounding the expected surplus by a constant. Hence, a linear function whose expected surplus can be upper bounded by a constant, would be an ideal candidate for a difficult function for the cGA.

We will prove in the following that the so-called BinVal-function, mapping a bit-string $(x_1,...,x_n)$ on the integer having the binary representation $(x_1,...,x_n)$, is such a difficult function:

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

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 $x_{i+1},...,x_n$ "to the right of" x_i. Therefore, a search point $(x_1,...,x_n)$ has a higher BinVal-value than a search point $(y_1,...,y_n)$, if and only if for the smallest index i with $x_i \neq y_i$ we have $x_i = 1$.

Assume, that BinVal (x) > 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 $x_{i+1},...,x_n$ and $y_{i+1},...,y_n$ cannot influence the order of BinVal(x) and BinVal (y). Then the surplus on the "unimportant bits" $x_{i+1},...,x_n$ and $y_{i+1},...,y_n$ 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 12. *For all constant $\varepsilon > 1$ the probability of the runtime of the cGA on* BinVal *being larger than $nK/(1+\varepsilon)$ is exponentially close (in K) to* 1:

$$\text{Prob}\left(T_{\text{BinVal}} > \frac{nK}{1+\varepsilon}\right) \geq 1 - \exp\left(-\frac{(\varepsilon-1)^2}{16(1+\varepsilon)} \cdot K\right).$$

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 9 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 ($i \in \{1,...,T\}$) 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 $nK/2 - nK/(1+\varepsilon) = (\varepsilon - 1)nK/(2+ 2\varepsilon)$ 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 9) 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 $(\varepsilon - 1)nK/(2 + 2\varepsilon)$ more elements than $B_1 \cup \cdots \cup B_T$ if and only if

$$|A_1 \cup \cdots \cup A_T| \geq \frac{n^2K}{2(1+\varepsilon)} + \frac{(\varepsilon - 1)nK}{4(1+\varepsilon)}.$$

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:

$$\begin{aligned}
&\text{Prob}\left(|A_1 \cup \cdots \cup A_T| \geq \frac{n^2K}{2(1+\varepsilon)} + \frac{(\varepsilon - 1)nK}{4(1+\varepsilon)}\right) \\
=&\text{Prob}\left(|A_1 \cup \cdots \cup A_T| \geq \frac{n^2K}{2(1+\varepsilon)} \cdot \left(1 + \frac{\varepsilon - 1}{2n}\right)\right) \\
\leq&\exp\left(-\frac{(\varepsilon - 1)^2}{4n^2} \cdot \frac{n^2K}{4(1+\varepsilon)}\right) = \exp\left(-\frac{(\varepsilon - 1)^2}{16(1+\varepsilon)} \cdot K\right).
\end{aligned}$$

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

Analogously to the proof of Theorem 11 we can show that the expected runtime of the cGA on BINVAL is polynomial in n and K regardless of any preconditions. Combining this with the just proven fact, that the runtime is $\Omega(Kn)$ with probability super-polynomially close to one, we get:

Corollary 13. *The expected runtime* $E^*(T_{\text{BINVAL}})$ *of the cGA for* BINVAL *is* $\Theta(Kn)$.

8. 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 lower bound of $\Omega(K\sqrt{n})$ on its expected runtime for any function and an upper bound of $O(Kn)$ on its expected runtime for any linear function.

One main result is the proof that some linear functions like ONEMAX are optimized by the cGA in the optimal expected runtime $\Theta(K\sqrt{n})$, while others like BINVAL have expected runtime $\Theta(Kn)$. Notice that this is in compliance with experimental results in (Harik et al., 1998), 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 the behavior of the cGA for non-linear functions. How does the runtime for quadratic functions or other

well-known test functions, like LeadingOnes or Trap, grow with n and K? Results of this kind can help to better understand the differences between the cGA and simple EAs, like the $(1+1)$ EA.

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. The same holds for the probability P^*_{BinVal} for $K=n$: we cannot prove a constant lower bound on the probability of a finite runtime, but guess that the probability is constant.

Acknowledgements

I thank Jens Jägersküpper, Tobias Storch, Ingo Wegener, and Carsten Witt for valuable advice and discussions.

References

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

Droste S (2005) Not all linear functions are equally difficult for the compact genetic algorithms. In: Beyer H-G et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, pp. 679–686. ACM Press, Washington, DC

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

Harik GR, Lobo FG and Goldberg DE (1998) The compact genetic algorithm. In: Proceedings of the IEEE Conference on Evolutionary Computation, pp. 523–528. IEEE Press, Anchorage, AK

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

Larranaga P and Lozano JA (2002) Estimation of Distribution Algorithms. Kluwer Academic Publisher

Motwani R and Raghavan P (1995) Randomized Algorithms. Cambridge University Press, Cambridge

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

Rudolph G (1997) Convergence Properties of Evolutionary Algorithms. Verlag Dr., Kovač

Wegener I (2002) Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In: Sarker R, Yao X and Mohammadian M (eds) Evolutionary Optimization, pp. 349 – 369. Kluwer

Wegener I (2005) Computational Complexity and EC. Tutorial at GECCO 2003, 2004, and 2005