UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

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

On the Utility of Populations

Thomas Jansen

No. CI-102/00

Technical Report ISSN 1433-3325 November 2000

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

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

On the Utility of Populations

Thomas Jansen*
FB Informatik, LS 2, Univ. Dortmund, 44221 Dortmund, Germany
jansen@ls2.cs.uni-dortmund.de

Abstract

Evolutionary algorithms (EAs) are population-based search heuristics often used for function optimization. Typically they use selection, mutation, and crossover as search operators. On many test functions EAs are outperformed by simple hill-climbers. Therefore, it is investigated whether the use of a population and crossover is at all advantageous. In this paper it is rigorously proven that the use of a population instead of just a single individuum can be an advantage of its own even without making use of crossover. This establishes the advantage of EAs compared to (random) hill-climbers on appropriate objective functions.

1 Introduction

Evolutionary algorithms (EAs) are a broad class of different randomized search heuristics all stemming from natural evolution. The probably best-known examples are genetic algorithms (GAs) (Goldberg 1989), evolution strategies (Schwefel 1995), and evolutionary programming (Fogel 1995). Typically, a population of points in the search space is evolved in generations using selection, mutation, and crossover. However, it is reported (Juels and Wattenberg 1995) that simple random mutation hill-climbers are able to outperform much more sophisticated EAs. This raises the question for what problems EAs clearly outperform such hill-climbers. We concentrate on the optimization of discrete, static objective functions, here. For continuous search spaces, especially for the sphere functions, some results are known. Beyer (1993, 1995a, 1995b) shows that as far as local performance on the sphere function is concerned the use of populations cannot help. In the case of a noisy environment things are different. Arnold and Beyer (2000) argue how the use of populations can be helpful in a noisy environment and increase the efficiency of the search. Jansen and Wegener (1999) prove that a genetic algorithm can by far outperform a random mutation hill-climber. This result relies heavily on the use of uniform crossover. Therefore, it remains open whether the use of a population alone, even without crossover, can be advantageous. Obviously, if one takes only the number of generations needed before a global optimum is found into account, it is trivial to prove that this is the case. But this is in some sense cheating: at least using a non-parallel computing environment the number of generations alone is no appropriate measure for the computational effort. This is due to the fact that in each generation typically the whole population is used to guide the search. Therefore, the size of the population has to be taken into account, too. We apply as performance measure the number of function evaluations until a global

*supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531)

optimum is found for the first time. Thus, we assume that the computational effort of the whole EA is proportional to the number of function evaluations. This holds at least for typical EAs.

We compare the performance of a certain genetic algorithm that we will define precisely in the next section with the so-called (1+1) EA. This simple evolutionary algorithm employs a "population" of size 1, bit-wise mutations and the plus-selection, known from evolution strategies. Therefore, it can be taken to be a kind of random mutation hill-climber. On typical functions, the (1+1) EA outperforms other well-known hill-climbers as steepest ascent oder next ascent hill-climber (Mitchell 1995). This holds especially for the example function considered here.

In the next section we give formal definitions of the (1+1) EA and the genetic algorithm considered. We try to get an intuitive understanding of circumstances when the use of a population can be advantageous. Then we define a class of example functions that capture these circumstances. In Section 3 we derive the expected run time of the $(1 + 1)$ EA asymptotically exact and give the expected run time for the GA in a simple special case. We prove these results in simple, understandable way. In the following section, Section 4 we generalize the results for the GA and can thereby prove an enormous gap between the expected run times. Finally, in Section 5 we draw some conclusions and discuss what is still open.

2 Definitions

The (1+1) EA is perhaps the most simple EA, using a population size of just 1 and mutation, only. The selection mechanism used is the plus-selection known from evolution strategies. This simple algorithm is subject of intense research, see Mühlenbein (1992), Rudolph (1997), Droste, Jansen, and Wegener (1998), Garnier, Kallel, and Schoenauer (1999) for example. For the sake of clarity we give a formal definition. We assume that we want to maximize some objective function $f\colon \{0,1\}^n \to \mathbb{R}$.

Algorithm 1 (The $(1+1)$ EA).

1. Choose $x \in \{0,1\}^n$ uniformly at random.
2. Create y by flipping each bit in x independently with probability $1/n$.
3. If $f(y) \geq f(x)$, then set $x := y$.
4. Continue at 2.

Since the (1+1) EA uses a kind of degenerated population of size 1 and accepts only strings with at least equal function value it can be regarded as a kind of random mutation hill-climber. Note, however, that since in each step it generates any $x' \in \{0,1\}^n$ as child y with positive probability, it cannot get stuck in any local optima. This distinguishes the (1+1) EA from other hill-climbing algorithms.

We want to compare the efficiency of the (1+1) EA with that of an evolutionary algorithm that uses a real population but no crossover. We want to prove that such an EA can indeed be superior to the simple (1+1) EA. On the one hand, we would like to have an EA for this purpose that is a kind of "standard EA" except for neglecting crossover. On the other hand, it is immediately obvious that advantages due to the use of a population can only occur if the population does not converge too rapidly. Therefore, we would like to apply some mechanism that helps us to maintain at least some minimal degree of diversity. There are numerous more or less complicated mechanisms to do so, see Bäck, Fogel, and Michalewicz (1997) for an overview. A very simple mechanism is avoidance of duplicates as suggested and investigated by Ronald (1998). An even simpler and computationally less expensive mechanism is avoidance of replications as used by Jansen and Wegener (1999). Avoidance of replications means that it is ensured, that every child is subject to a mutation flipping at least 1 bit or to crossover. We will see that this weak and computationally cheap mechanism is sufficient to maintain an acceptable degree of diversity.

The EA we use is an elitist steady-state GA, using fitness-proportional selection for reproduction, bit-wise mutations with probability $1/n$ (just as the (1+1) EA), no crossover, reverse proportional selection for deletion, and deletion after insertion. We use population size N and leave the concrete choice of N open for the moment. Again, for the sake of clarity, we give a formal definition.

Algorithm 2 (The Genetic Algorithm).

1. `For` $i := 1$ `To` N `Do Choose` $x_i \in \{0,1\}^n$ `uniformly at random.`
2. `Repeat`
3. `Choose` $y \in \{x_1, x_2, \dots, x_N\}$ `such that Prob` $(y = x_i) = f(x_i) / \sum_{j=1}^{N} f(x_j)$.
4. `Create` x_{N+1} `by flipping each bit in` y `independently with probability` $1/n$.
5. `Until` $x_{N+1} \neq y$.
6. `Sort` $\{x_1, x_2, \dots, x_{N+1}\}$ `such that` $f(x_1) \geq f(x_2) \geq \cdots \geq f(x_{N+1})$ `holds.`
7. `Choose` $z \in \{x_2, x_3, \dots, x_{N+1}\}$ `such that`
   `Prob` $(z = x_i) = (f(x_1) + f(x_{N+1}) - f(x_i)) \,/\, \sum_{j=2}^{N+1} (f(x_1) + f(x_{N+1}) - f(x_j))$.
8. `Remove` z `from` $\{x_2, x_3, \dots, x_{N_1}\}$.
9. `Continue at 2.`

Our intuitive idea for the origin of an advantage for the GA compared to the (1+1) EA is the following (Rowe 2000). Assume we have a function that mainly consists of an easy to follow path to a local optimum that has second best function value. Then a direct mutation to a global optimum is needed. The expected waiting time for such a mutation can be quite large depending on the Hamming distance between the local and a global optimum. Since the GA uses a population and the relative weak proportional selection, it has a good chance to "diffuse" the population around the local optimum. Therefore, some members of the population have a smaller Hamming distance to a global optimum what can lead to a quicker finding of this global optimum. We now formally define a class of objective functions with such properties and prove for some members of this class that they have the desired properties such that the (1+1) EA is outperformed by the GA. The function class is a modified, in the first place scaled version of JUMP_k as used by Jansen and Wegener (1999) to exemplify the utility of uniform crossover.

Definition 3. *The function* $\text{SJUMP}_{k,s}\colon \{0,1\}^n \to \mathbb{R}$ *(short for* SCALEDJUMP*) is defined for any* $n \in \mathbb{N}$, $s \in \mathbb{N} \setminus \{1\}$, $k \in \{1, 2, \ldots, n\}$ *by*

$$\text{SJUMP}_{k,s}(x) := \begin{cases} s^{\|x\|_1} & \text{if } (\|x\|_1 \leq n-k) \vee (\|x\|_1 = n) \\ s^{n-k} + n - k - \|x\|_1 & \text{otherwise,} \end{cases}$$

for each $x \in \{0,1\}^n$ *where* $\|x\|_1$ *denotes the number of ones in* x.

3 First Results

The results for the (1+1) EA follow more or less directly from the investigations of the (1+1) EA on JUMP_k by Jansen and Wegener (1999). For the sake of completeness we present full proofs here, too.

Theorem 4. *The expected run time of the* $(1+1)$ *EA on* $\text{SJUMP}_{k,s}\colon \{0,1\}^n \to \mathbb{R}$ *for* $k \in \{2, 3, \ldots, n\}$ *and* $s \in \mathbb{N} \setminus \{1\}$ *is* $\Theta\left(n^k\right)$.

Proof. We begin with an upper bound on the expected run time. Consider levels of strings

$$l_i := \{x \in \{0,1\}^n \mid \|x\|_1 = i\}$$

for $i \in \{0, 1, \ldots, n\}$. Note, that for all i and all $x, y \in l_i$ we have $\text{SJUMP}_{k,s}(x) = \text{SJUMP}_{k,s}(y)$ since $\text{SJUMP}_{k,s}$ is a symmetric function. Furthermore, for all $i \neq j$ and all $x \in l_i$ and all $y \in l_j$ we have $\text{SJUMP}_{k,s}(x) \neq \text{SJUMP}_{k,s}(y)$. Thus, once the $(1+1)$ EA leaves some level l_i (that is we have $x \in l_i$, $y \notin l_i$ and $\text{SJUMP}_{k,s}(x) > \text{SJUMP}_{k,2}(y)$) no string from l_i can ever become current string. Let p_i denote the probability that in one generation the $(1+1)$ EA accepts a string within level l_j with $j \neq i$ as new current string given that the current string belongs to l_i. Clearly, then

$$\sum_{i=0}^{n-1} \frac{1}{p_i}$$

is an upper bound on the expected run time. For $i < n-k$ it is sufficient to mutate exactly one of the $n-i$ bits with value 0. For $i > n-k$ it is sufficient to mutate exactly one of the i bits with value 1. This yields $p_i \geq \min\{i, n-i\} \frac{1s}{n}(1-1/n)^{n-1}$ for $i \neq n-k$. For $i = n-k$ we have $p_i = 1/n^k(1-1/n)^{n-k}$, since the only way to increase the function value is to mutate exactly the k bits with value 0 leading to the unique global optimum. Thus, we have

$$\left(1 - \frac{1}{n}\right)^{1-n} \cdot \left(\left(\sum_{\substack{0 \leq i \leq n/2 \\ i \neq n-k}} \frac{n}{n-i} \right) + n^k \right) = \mathrm{O}\left(n^k\right)$$

as upper bound on the expected run time.

Now, we prove a lower bound on the expected run time. We simplify the analysis a bit by considering the expected number of steps until we have $x \in l_{n-1} \cup l_n$ for the first time instead of the expected run time. After random initialization this is the case with probability $1 - (n+1)/2^n$.

Assume we have $x \in \bigcup_{n-1<i\leq n-k} l_i$. Then, we have $\text{SJ}\textsc{ump}_{k,s}(x) > \text{SJ}\textsc{ump}_{k,s}(y)$ for all $y \in l_{n-1}$. Thus, after some steps of the (1 + 1) EA we have $x \in l_{n-k} \cup l_n$. The probability to have $y \in l_n$ is bounded above by $1/n^2$. We know from the proof of the upper bound that the expected number of steps needed to reach $x \in l_{n-k}$ is bounded above by $\mathrm{O}(n \log n)$, given that we do not encounter $y \in l_n$. By Markov's inequality with probability $1 - 1/\log n$ we reach $x \in l_{n-k}$ within $\mathrm{O}\left(n \log^2 n\right)$ steps. The probability to reach $y \in l_n$ within $\mathrm{O}\left(n \log^2 n\right)$ steps is bounded above by $\mathrm{O}\left(\left(\log^2 n\right)/n\right)$. Thus, with probability

$$\left(1 - \frac{1}{\log n}\right) \cdot \left(1 - \mathrm{O}\left(\frac{\log^2 n}{n}\right)\right) = 1 - \mathrm{O}\left(\frac{1}{\log n}\right)$$

we reach some $x \in l_{n-k}$ in this case.

Now, assume that we have $x \in \bigcup_{0\leq i<n-k}$. If we encounter some $y \in l_j$ with $n - k < j < n - 1$, we get this y as new current string x. Then, we reach some $x \in l_{n-k}$ with probability $1 - \mathrm{O}(1/\log n)$ as we just saw above. The probability to reach $y \in l_n$ in one step is bounded above by $1/n^{n-i} < 1/n^k$. Thus, we concentrate on the probability to encounter $y \in l_{n-1}$. In each generation this probability is upper bounded by

$$\binom{n-i}{1}\frac{1}{n^{n-i-1}} + \binom{i}{1}\frac{1}{n^{n-i+1}} = \mathrm{O}\left(\frac{n-i}{n^{n-i-1}}\right).$$

Note that $n - i - 1 \geq k$ holds. If we have $i \leq n - 4$, we have

$$\mathrm{O}\left(\frac{n-i}{n^{n-i-1}}\right) = \mathrm{O}\left(\frac{1}{n^{n-i-2}}\right) = \mathrm{O}\left(\frac{1}{n^2}\right)$$

as upper bound. Otherwise we have

$$\mathrm{O}\left(\frac{n-i}{n^{n-i-1}}\right) = \mathrm{O}\left(\frac{1}{n^{n-i-1}}\right) = \mathrm{O}\left(\frac{1}{n^k}\right)$$

as upper bound. Therefore, the probability to encounter some $y \in l_{n-1}$ within $\mathrm{O}\left(n \log^2 n\right)$ steps is bounded above by $\mathrm{O}\left(\left(\log^2 n\right)/n\right)$ in any case.

We conclude that with probability $1 - \mathrm{O}(1/\log n)$ some $x \in l_{n-k}$ becomes current string of the (1 + 1) EA. If this happens, only a direct mutation to $y \in l_n$ leads to a number current string x. Such a mutation occurs with probability $\mathrm{O}\left(1/n^k\right)$ Thus, we have $\Omega\left(n^k\right)$ as lower bound on the expected run time. □

Obviously, the simple (1+1) EA is able to optimize $\text{SJ}\textsc{ump}_{k,s}$ quite efficiently, at least for small k. Note, that the size of s does not matter for this algorithm. This is different for the GA, which uses a fitness-sensitive selection mechanism, namely proportional selection. Since the (1+1) EA is already quite fast on $\mathrm{SJ{\scriptsize UMP}}_{k,s}$ for small values of k, it is astonishing that the GA can outperform it even for $k = 2$. We concentrate on the special case $\mathrm{SJ{\scriptsize UMP}}_{2,n^2}$, first. This is useful, since in this simpler case it is easier to develop the methods and insights that will yield the result in more general cases, too. Here, however, no too big advantage is possible. But we will prove a smaller asymptotic bound on the expected run time given an appropriate choice of the population size.

Theorem 5. *The expected run time of Algorithm 2 with population size $N = \lfloor\sqrt{n}\rfloor$ on the function $\mathrm{SJ{\scriptsize UMP}}_{2,n^2} \colon \{0,1\}^n \to \mathbb{R}$ is bounded above by $O\left(n^{3/2}\right)$.*

Before we prove the result on the GA we consider two events that are of special interest and both have to do with selection. First, we are interested in a lower bound on the probability to select some member of the population with a certain function value as parent.

Lemma 6. *Consider Algorithm 2 on the function $\mathrm{SJ{\scriptsize UMP}}_{k,s}$. Assume that x is a member of the current population. Let f_1 and f_2 denote the function value of the current best and current second best members of the population. Consider the selection in line 3 of the algorithm.*

a) The probability to select some member y of the current population with

$$\mathrm{SJ{\scriptsize UMP}}_{k,s}(y) = \mathrm{SJ{\scriptsize UMP}}_{k,s}(x)$$

is bounded below by $\mathrm{SJ{\scriptsize UMP}}_{k,s}(x)/(Nf_1)$.

b) The probability to select some member y of the current population with

$$\mathrm{SJ{\scriptsize UMP}}_{k,s}(y) = f_1$$

is bounded below by $1 - (f_2 N)/f_1$.

Proof. a) Let S denote the sum of the function values of the current population, i. e.

$$S := \sum_{i=1}^{N} \mathrm{SJ{\scriptsize UMP}}_{k,s}(x_i).$$

Let r denote the number of members of the current population with function value $\mathrm{SJ{\scriptsize UMP}}_{k,s}(x)$, i. e.

$$r := \left|\{x_i \mid (1 \leq i \leq N) \wedge (\mathrm{SJ{\scriptsize UMP}}_{k,s}(x_i) = \mathrm{SJ{\scriptsize UMP}}_{k,s}(x))\}\right|.$$

Since we are interested in lower bounds, we may assume that we have $r = 1$. This can be seen as follows. The probability that we select an individuum with function value $\mathrm{SJ{\scriptsize UMP}}_{k,s}(x)$ equals

$$\frac{r\mathrm{SJ{\scriptsize UMP}}_{k,s}(x)}{S}.$$

If we remove some string y with $\text{SJUMP}_{k,s}(y) = \text{SJUMP}_{k,s}(x)$ from the population and replace it with some string z with $\text{SJUMP}_{k,s}(z) \neq \text{SJUMP}_{k,s}(x)$, then the probability to select a string with function value $\text{SJUMP}_{k,s}(x)$ equals

$$\frac{(r-1)\text{SJUMP}_{k,s}(x)}{S - \text{SJUMP}_{k,s}(x) + \text{SJUMP}_{k,s}(z)}.$$

We have

$$\begin{aligned} & \frac{r\text{SJUMP}_{k,s}(x)}{S} - \frac{(r-1)\text{SJUMP}_{k,s}(x)}{S - \text{SJUMP}_{k,s}(x) + \text{SJUMP}_{k,s}(z)} \\ = \quad & \frac{\text{SJUMP}_{k,s}(x) \cdot (S - r\text{SJUMP}_{k,s}(x) + r\text{SJUMP}_{k,s}(z))}{S\,(S - \text{SJUMP}_{k,s}(x) + \text{SJUMP}_{k,s}(z))} \geq 0 \end{aligned}$$

since obviously $S \geq r\text{SJUMP}_{k,s}(x)$ holds and we have $\text{SJUMP}_{k,s}(a) \geq 0$ for all $a \in \{0,1\}^n$. Therefore, it follows via induction that we can assume $r = 1$ without loss of generality.

Now, we see that the probability to select some member of the population with function value $\text{SJUMP}_{k,s}(x)$ is bounded below by

$$\frac{\text{SJUMP}_{k,s}(x)}{\text{SJUMP}_{k,s}(x) + (N-1)f_1} \geq \frac{\text{SJUMP}_{k,s}(x)/f_1}{N} = \frac{\text{SJUMP}_{k,s}(x)}{Nf_1}$$

as claimed.

b) As for a) we assume that there is exactly one string x with $\text{SJUMP}_{k,s}(x) = f_1$ in the current population. Then, we have $S \leq f_1 + (N-1)f_2$ so that we have

$$\frac{f_1}{f_1 + (N-1)f_2} \geq \frac{f_1/f_2}{f_1/f_2 + N} = 1 - \frac{1}{f_1/(Nf_2) + 1} \geq 1 - \frac{f_2 N}{f_1}$$

as a lower bound as claimed.

□

The other aspect of selection is selection for deletion in line 7. Here, we are interested in an upper bound on the probability to select some member of the extended population with a certain function value for deletion.

Lemma 7. *Consider Algorithm 2 on the function* $\text{SJUMP}_{k,s}$*. We make the following assumptions about the current population:*

- *The population contains $N+1$ strings.*
- *The current best member of the population contains exactly $n-k$ ones.*
- *There is exactly one member x of the population which contains exactly j ones, where we have $n-k < j < n$.*

Consider the selection in line 7 of the algorithm.

a) The probability to select x is bounded above by $2/N$.

b) If there is at least one member of the population with less than $n-k$ ones, then the probability to select x is bounded above by $1/(s+N)$.

Proof. a) The probability to select x for deletion is obviously maximal, if on the one hand the number of ones in x, j, equals $n-1$, since then $\text{SJUMP}_{k,s}(x)$ is as small as possible, and on the other hand the sum of all function values is as large as possible. Therefore, we can concentrate on the case that x contains exactly $n-1$ ones and all other members of the population contain exactly $n-k$ ones. In this case the probability to select x equals

$$\frac{s^{n-k}}{(N-1)s^{n-k}+s^{n-k}} = \frac{1}{N-(N-1)(k-1)/s^{n-k}} \leq \frac{2}{N}$$

as claimed.

b) The population contains x, at least one string with less then $n-k$ ones and at least one string with $n-k$ ones. The probability to select x is maximal, if there is exactly one string with $n-k-1$ ones in the population, if all other strings contain exactly $n-k$ ones and, as above, x contains exactly $j=n-1$ ones. Therefore,

$$\frac{s^{n-k-1}+k-1}{s^{n-k-1}+k-1+s^{n-k}+(N-2)s^{n-k-1}} = \frac{1+(k-1)/s^{n-k-1}}{s+N-1+(k-1)/s^{n-k-1}} \leq \frac{1}{s+N}$$

is an upper bound on the probability to select x.

□

Proof of Theorem 5. We use a population size of $N=\lfloor\sqrt{n}\rfloor$ and assume for the sake of notational simplicity that we have $N=\lfloor\sqrt{n}\rfloor=\sqrt{n}$. The generalization to the other cases is trivial. We describe a kind of typical run that ends when the unique global optimum becomes member of the current population for the first time. We divide that run into two disjoint phases and estimate the expected length of each phase. The first phase begins after random initialization and ends when a string with $n-2$ ones or the unique global optimum becomes member of the current population for the first time. The second phase begins after the first phase and ends when the unique global optimum becomes member of the current population for the first time. Note, that in the second phase we always have at least one string with $n-2$ ones in the population. This is due to the elitism employed in the GA.

The expected length of the first phase is $O(n\log n)$. This is easy to see. We estimate the probability to select the current best member of the population as parent in line 3. Either the whole population consists of N strings that all have current best function value, or we have $f_2/f_1 = \mathrm{O}(1/n^2)$. In the first case we obviously select such a parent with probability 1. In the latter case, we apply Lemma 6 b). Thus, we have

$$1 - \mathrm{O}\left(\frac{N}{n^2}\right) = 1 - \mathrm{O}\left(\frac{1}{n^{3/2}}\right)$$

in any case as a lower bound for selecting the current best member of the current population for reproduction. Due to the elitistic replacement strategy the current best member of the population cannot be replaced. Therefore, we can pessimistically ignore steps where some other member of the population is chosen as parent. Now, consider the case that the chosen parent is the current best member of the population and contains i bits with value 1. Then, we have a probability of at least $(n-i)/n(1-1/n)^{n-1}$ to create a child y with larger function value. Thus, the expected waiting time for improving the current best member of the population is bounded above by

$$\frac{n^{3/2}}{n^{3/2}-1} \cdot \frac{en}{n-i} = \mathrm{O}\left(\frac{n}{n-i}\right)$$

in this situation. We sum up these expected waiting times and get

$$\sum_{i=0}^{n-1} \mathrm{O}\left(\frac{n}{n-i}\right) = \mathrm{O}(n \log n)$$

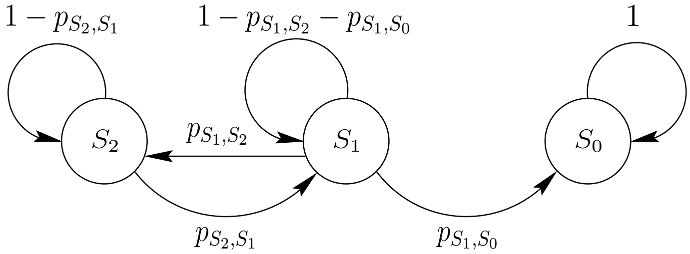
as upper bound on the expected length of the first phase. Note, that these considerations do not depend on the initial population.

In the second phase we are especially interested in three special events. First, we investigate the probability that we introduce a string with exactly $n-1$ ones in the population when there is no such string present in one generation. Second, we are interested in the probability to remove the only occurrence of a string with exactly $n-1$ ones without introducing the global optimum instead in one generation. Finally, we investigate the probability to create the global optimum in one generation given that at least one string with exactly $n-1$ ones is present in the current population.

We consider the current population and distinguish three different "states" it can be in. Note, that each of these three states does not represent one special population but a large number of different populations that all have some property in common. If no string in the current population has more than $n-2$ ones we say the GA is in state A. If the string with the maximal number of ones in the current population contains $n-1$ ones, we say the GA is in state B. Finally, if the global optimum is member of the current population we speak of state C. We want to derive an upper bound on the expected number of generations the GA needs to "reach state C", i. e. to have a population that satisfies the defining condition of state C.

Let us consider a Markov chain with three states S_0, S_1, and S_2. We are interested in the first hitting time of S_0 starting in S_2. For $X, Y \in \{S_0, S_1, S_2\}$ we denote by $p_{X,Y}$ the transition probability for a state transition from X to Y.

We assume pessimistically that we have $p_{S_2,S_0} = 0$. This can obviously only enlarge the first hitting time of S_0. Since we are interested in the first hitting time of S_0, only, we can assume $p_{S_0,S_0} = 1$, so $p_{S_0,S_2} = p_{S_0,S_1} = 0$ follows. Due to our assumptions we have $p_{S_2,S_1} + p_{S_2,S_2} = 1$, so we can replace p_{S_2,S_2} by $1 - p_{S_2,S_1}$. In the same way we can replace p_{S_1,S_1} by $1 - p_{S_1,S_2} - p_{S_1,S_0}$. The resulting Markov chain can be visualized as follows.



Let the random number of transitions starting in S_2 leading to S_0 be denoted by T_{S_2}. Let the random number of transitions starting in S_1 leading to S_0 be denoted by T_{S_1}. Then we have

$$\mathrm{E}\left(T_{S_2}\right) = \left(1 - p_{S_2,S_1}\right)\left(1 + \mathrm{E}\left(T_{S_2}\right)\right) + p_{S_2,S_1}\left(1 + \mathrm{E}\left(T_{S_1}\right)\right)$$

which leads to

$$\mathrm{E}\left(T_{S_2}\right) = \frac{1}{p_{S_2,S_1}} + \mathrm{E}\left(T_{S_1}\right).$$

For $\mathrm{E}\left(T_{S_1}\right)$ we have

$$\mathrm{E}\left(T_{S_1}\right) = \left(1 - p_{S_1,S_0} - p_{S_1,S_2}\right)\left(1 + \mathrm{E}\left(T_{S_1}\right)\right) + p_{S_1,S_0} + p_{S_1,S_2}\left(1 + \mathrm{E}\left(T_{S_2}\right)\right)$$

leading to

$$\mathrm{E}\left(T_{S_1}\right) = \frac{1}{p_{S_1,S_0}} + \frac{p_{S_1,S_2}}{p_{S_2,S_1} \cdot p_{S_1,S_0}}.$$

We look for an upper bound on the expected absorption time and thus need an upper bound on p_{S_1,S_2} and lower bounds on p_{S_2,S_1} and p_{S_1,S_0}. In order to obtain meaningful results, we connect the described Markov chain to the genetic algorithm we consider. First of all, the defining conditions of the three different "states" of the GA are obviously not sufficient to describe unambiguously exactly one population. In fact, there are for each such "state" a lot of different populations that satisfy the according defining condition. The probability to reach some population that meets condition B in one generation starting from a population that meets condition A depends on the specific population the GA starts in. The same holds for all other "transition probabilities.

If we associate A with S_2, B with S_1, and C with S_0 we have the following. If the minimal number of zeros a member of the current population has equals i, then the population satisfies a condition that is associated with S_i. We choose values for p_{S_2,S_1}, p_{S_1,S_0}, and p_{S_1,S_2} in the following way. Let P_A denote the set of all populations that satisfy condition A. Let P_B and P_C denote the according sets for conditions B and C. We denote the change from a population P_1 to another population P_2 in one generation by $P_1 \rightarrow P_2$. We want p_{S_2,S_1} to be a lower bound on the probability to come from a population satisfying condition A to a population satisfying condition B, i. e.

$$p_{S_2,S_1} \leq \min \left\{ \sum_{P_2 \in P_B} \text{Prob}\,(P_1 \rightarrow P_2) \mid P_1 \in P_A \right\}.$$

We want p_{S_1,S_0} to be a lower bound on the probability to come from a population satisfying condition B to a population satisfying condition C, thus we want to have

$$p_{S_1,S_0} \leq \min \left\{ \sum_{P_2 \in P_C} \text{Prob}\,(P_1 \rightarrow P_2) \mid P_1 \in P_B \right\}.$$

Finally, we want p_{S_1,S_2} to be an upper bound on the probability to come from a population satisfying condition B to a population satisfying condition A, that is

$$p_{S_1,S_2} \geq \max \left\{ \sum_{P_2 \in P_A} \text{Prob}\,(P_1 \rightarrow P_2) \mid P_1 \in P_B \right\}.$$

Such lower bounds for p_{S_2,S_1} and p_{S_1,S_0} are easy to find. To introduce a string with exactly $n-1$ ones into the population can be achieved in the following way. First, one selects the current best member of the population, which is a string with exactly $n-2$ ones. As in the first phase, we either have the the whole population consists of strings with $n-2$ ones and have current best function value. In this case we obviously select such a string with probability 1. Otherwise, we apply Lemma 6 b). We have $f_1 = f_2 = n^{2(n-2)}$ or $f_1 = n^{2(n-2)}$ and $f_2 \leq n^{2(n-3)}$. Thus, the probability for this event is bounded below by $1 - N/n^2$ in any case. Then exactly one of the two bits with value zero is mutated. This event has probability $(2/n)(1-1/n)^{n-1}$. Finally, this child is not deleted from the (at this moment enlarged) population. We apply Lemma 7 a) and have $1 - 2/N$ as a lower bound. Therefore, we have

$$\left(1 - \frac{1}{n^{3/2}}\right) \cdot \frac{1}{2en} \cdot \left(1 - \frac{2}{\sqrt{n}}\right) = \Omega\left(\frac{1}{n}\right)$$

as a lower bound for p_{S_2,S_1}.

For p_{S_1,S_0} we consider the case that a string with $n-1$ ones is selected as parent and the only bit with value 0 is mutated, only. The probability to select such a string is bounded below by

$$\frac{n^{2(n-2)} - 1}{Nn^{2(n-2)}} = \frac{1}{\sqrt{n}} - \frac{1}{n^{2(n-2)+1/2}} \geq \frac{1}{2\sqrt{n}}$$

due to Lemma 6 a). Thus, we have

$$\frac{1}{2\sqrt{n}} \cdot \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} = \Omega\left(\frac{1}{n^{3/2}}\right)$$

as a lower bound on p_{S_1,S_0}.

Finally, we need an upper bound for p_{S_1,S_2}. Obviously, a necessary condition for the event we consider is that the only string with $n-1$ ones that is in the population is selected for deletion. We distinguish two different cases. First, assume that at least one of the strings in the set $\{x_2, x_3, \ldots, x_{N+1}\}$ (in line 7 of Algorithm 2) contains less than $n-2$ ones. This can be due to the fact that such a string was already member of the population or due to the fact that such a child was created. In this case the probability to select the string with $n-1$ ones for deletion is bounded above by $1/n^2$ due to Lemma 7 b). Thus, in the case that a string with less than $n-2$ ones is created by mutation we have $1/n^2$ as an upper bound on p_{S_1,S_2}.

Now, we consider the second case, where except for the one string x with $n-1$ ones all other members of the population contain exactly $n-2$ ones. First of all, the probability to select this string x for deletion is bounded above by $2/N$ due to Lemma 7 a). We consider two sub-cases with respect to the parent of the newly created child. If the parent is a string with $n-2$ ones, then the probability to create a child with $n-2$ ones is bounded above by $2/n$, since at least one of the two bits with value zero has to flip. Otherwise, either a child with a number of ones that is different from $n-2$ is created or we have the case of a replication that is not allowed due to the definition of Algorithm 2. If, on the other hand, the parent is the only string with $n-1$ ones, it is obviously a necessary condition that this string is selected for reproduction. The probability of such a selection is bounded above by

$$\frac{n^{2(n-2)}-1}{Nn^{2(n-2)}} \leq \frac{1}{\sqrt{n}}$$

due to Lemma 6 a). Therefore, we have

$$\max\left\{\frac{1}{n^2}, \frac{2}{\sqrt{n}} \cdot \frac{2}{n}, \frac{2}{\sqrt{n}} \cdot \frac{1}{\sqrt{n}}\right\} = \frac{2}{n}$$

as an upper bound on p_{S_1,S_2} in any case.

We combine what we have and get

$$\mathrm{E}\left(T_{S_2}\right) = \mathrm{O}\left(n^{3/2}\right)$$

leading to

$$\mathrm{E}\left(T_{S_1}\right) = \mathrm{O}\left(n\right) + \mathrm{O}\left(n^{3/2}\right) = \mathrm{O}\left(n^{3/2}\right)$$

which is also an upper bound on the expected length of the second phase. Together with the upper bound on the expected length of the first phase we obtain the desired result. □

We see, that the GA optimizes SJUMP_{2,n^2} on average $\Omega\left(\sqrt{n}\right)$-times faster than the (1+1) EA. Note, that this result depends on the appropriate choice of the population size N. For $N = \Theta\left(\sqrt{n}\right)$ similar results can be proven. Note, however, that already for $N \geq n$ we cannot prove any better bound then $\Omega\left(n^2\right)$. We will focus on this problem in the next section. By doing so, we will also be able to prove stronger results.

4 Generalization of the Results

There are two main problems with the results from the previous section. First, we proved a gap in the order of $\Omega\left(\sqrt{n}\right)$. Since evolutionary algorithms are general, robust search heuristics and not specialized optimization algorithms, we do not really expect optimal or nearly optimal expected run times. Therefore, being by a factor of $\sqrt{n}$ slower might still be acceptable. Second, and more important, the result relies on the population size. It is known that parameterization of evolutionary algorithms is a difficult task. Nevertheless, we prefer results not showing a too big dependence on the parameter setting. Thus, we want to strengthen the result with respect to two goals. First, we want to increase the size of the gap between the $(1+1)$ EA and the GA. Second, we want to decrease the dependence on the population size. We will achieve both, at least to some degree, by considering $\text{SJ}\textsc{ump}_{k,s}$ for other values of k and s. For $\text{SJ}\textsc{ump}_{k,s}$ with $k > 2$, things change a little. The expected run time of the $(1+1)$ EA grows to $\Theta\left(n^k\right)$. Thus, the GA "gets more time" to outperform this simple EA. Now, in the following we prove a statement for the GA depending on the choice of the population size N. Note, that for a wide range of possible settings the GA clearly outperforms the $(1+1)$ EA on $\text{SJ}\textsc{ump}_{k,s}$ on average.

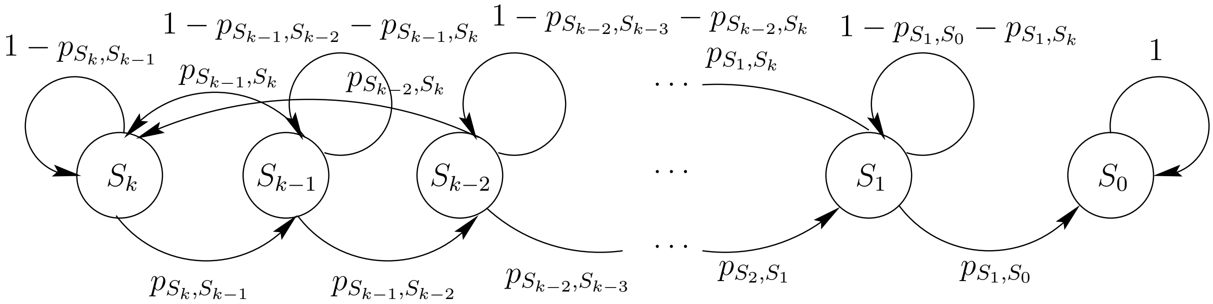
Theorem 8. *Consider Algorithm 2 on the function* $\text{SJ}\textsc{ump}_{k,s}\colon \{0,1\}^n \to \mathbb{R}$ *with* $k > 1$, $k = O((\log n) / \log\log n)$, *and* $s \in \mathbb{N} \setminus \{1\}$ *with* $s \geq n^2$. *The expected run time of the GA with population size* N, *where* $n \leq N \leq \sqrt{s}$ *holds, is* $O\left(nN(c \cdot k)^{k+1}\right) = n^{O(1)}$, *where* c *is some positive constant.*

Proof. This time we divide a run of the GA into three disjoint phases. As in the proof of Theorem 5 the first phase starts after random initialization and ends when an optimal string or some string with exactly $n-k$ ones becomes member of the current population for the first time. As in the proof of Theorem 5 we see that the first phase has expected length $\mathrm{O}\left(n \log n\right)$.

The second phase begins after the first phase and ends when we have at least $N/2$ strings in the current population each with at least $n-k$ ones, or the global optimum. The expected length of the second phase is $\mathrm{O}\left(Nn\right)$. We always have at least one string in the population with exactly $n-k$ bits with value one. Assume that there are at most $3N/4$ strings with at least $n-k$ ones. The probability to select the current best member of the population, which is a string with exactly $n-k$ ones, is bounded below by $1-N/s \geq 1-1/n$. The probability to get a child y different from the parent that contains exactly $n-k$ bits with value one, too, is bounded below by $\Omega(1/n)$. The probability to select one of the at least $N/4$ string with more than k bits with value zero is lower below by $1-1/(s+N)$. Thus, the probability to decrease the number of strings with more than k zeros is bounded below by $\Omega(1/n)$. On the other hand, the probability to increase the number of strings such strings is bounded above by $(1/n) \cdot 1/(s+N) \leq 1/n^3$. Therefore, with probability exponentially close to 1, after $\mathrm{O}\left(Nn\right)$ generations there at least $N/2$ strings with at least $n-k$ ones in the population.

The third (and final) phase begins after the second phase and ends when an optimal string becomes member of the current population for the first time. As in the proof of Theorem 5 we consider a special Markov chain and derive an upper bound on the first hitting time of an absorbing state. Then, we show that the expected first hitting time is an upper bound on the expected length of the second phase of the GA.

The Markov chain has $k+1$ states, namely $S_0, S_1, \ldots, S_k$. We use the same notations as in the proof of Theorem 5 and consider the following Markov chain. Transitions that are not shown in the figure have probability 0.



We use a simpler estimation than in the proof of Theorem 5. We assume that we start in S_k. The main idea is to give a lower bound on the probability for a sequence of transitions that ends in S_0 via S_{k-1}, S_{k-2}, ... , S_1 in that ordering without ever making a transition from S_i to S_k with $i < k$. Note, that all the only other transitions that have none zero probability are of the form $S_i \to S_i$ and have probability $1 - p_{S_i,S_{i-1}} - p_{S_i,S_k}$.

Assume that p is such a lower bound. Assume that t is an upper bound on the expected number of steps to come from S_k to S_0 in such a run. Then, t/p is an upper bound on the expected absorption time of the described Markov chain.

We claim that

$$\prod_{0<i<k} \frac{p_{S_i,S_{i-1}}}{p_{S_i,S_{i-1}} + p_{S_i,S_k}}$$

is such a lower bound p. That is easy to see. Assume that the current state is S_i with $i > 0$ and $i < k$. Since there are only three different transitions with non-zero probability, the next state that is different from S_i is either S_{i-1} or S_k. The probability, that the next state is S_{i-1} equals

$$\frac{p_{S_i,S_{i-1}}}{p_{S_i,S_{i-1}} + p_{S_i,S_k}}$$

since this is the probability to reach S_{i-1} under the condition that either S_i or S_k are reached. Due to the independence of the random choices made in each state the claim follows.

We claim that

$$\sum_{0<i\leq k} \frac{1}{p_{S_i,S_{i-1}}}$$

is such an upper bound on t. This is obvious, since under the condition that the state S_{i-1} is reached from S_i the expected waiting is upper bounded by $1/p_{S_i,S_{i-1}}$.

We conclude that the first hitting time of S_0 is bounded above by

$$\left(\prod_{0<i<k} \frac{p_{S_i,S_{i-1}}}{p_{S_i,S_{i-1}} + p_{S_i,S_k}}\right)^{-1} \cdot \left(\sum_{0<i\leq k} \frac{1}{p_{S_i,S_{i-1}}}\right).$$

Now, we have to define a connection between this Markov chain and the GA we are interested in. In order to do so, we say that the current population of the GA satisfies condition C_i, if the population contains at least one string with exactly i zeros. Assume that $p_{S_i,S_{i-1}}$ is a lower bound on the probability to get in one generation from a population P_1 satisfying condition C_i to a population P_2 satisfying condition C_{i-1} (nor containing the global optimum), where we sum over all such P_2 and minimize over all such P_1. Assume that p_{S_i,S_k} is an upper bound on the probability to get in one generation from a population P_1' satisfying condition C_i to a population P_2' neither satisfying C_i nor C_{i-1}, where we sum over all such P_2' and maximize over all such P_1'. Then, the expected length of the second phase is bounded above by the expected first hitting time of S_0 in the Markov chain considered above.

We claim that with

$$p_{S_i,S_{i-1}} = \frac{1}{6Nn} = \Omega\left(\frac{1}{Nn}\right)$$

we have such a lower bound. We assume that the current population satisfies condition C_i. Obviously, the following sequence of three events is sufficient to get from the current population to some population satisfying condition C_{i-1}. We assume that there is exactly one string with exactly i zeros. Otherwise condition C_i remains satisfied, anyway.

1. A string with exactly i zeros is selected in line 3.
2. Exactly one of these zero-valued bits is flipped in line 4.
3. The string with i zeros is not selected for replacement in line 7.

The first event has a probability that is lower bounded by

$$\frac{s^{n-k} + n - k - (n-i)}{s^{n-k} + n - k - (n-i) + (N-1)s^{n-k}} = \frac{sn - k + i - k}{Ns^{n-k} + i - k} > \frac{1}{2N}.$$

The second event has probability

$$\binom{i}{1}\frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} > \frac{i}{en}.$$

In order to lower bound the probability of the last event, we can assume that each string in the current population contains at least $n-k$ ones. Then the probability of the third event is lower bounded by

$$1 - \frac{2}{N}.$$

Thus, we see that

$$\frac{1}{2N} \cdot \frac{i}{en} \cdot \left(1 - \frac{2}{N}\right) > \frac{i}{6Nn}$$

is an appropriate lower bound as claimed.

Our next claim is that with

$$p_{S_i,S_k} = \mathrm{O}\left(\frac{1}{N^2} + \frac{k}{Nn} + \frac{1}{s+N}\right)$$

we have such an upper bound. Obviously, we get an upper bound if we consider the event to get from a population satisfying condition C_i to any population neither satisfying condition C_i nor condition C_{i-1} (and not containing the global optimum, of course). Just as above we assume that there is exactly one string x with i zeros in the current population. We consider three disjoint events A, B, and C that all result in such a population. It will be obvious from the description of the events that no other events can lead to such a population.

A: In line 4, x is selected, we get a child y with j zeros, where $j > i$ holds, and in line 7 x is selected for replacement. The probability of the selection for reproduction is bounded above by $\mathrm{O}(1/N)$. The probability for the mutation can be upper above by 1, of course. The probability of the selection for replacement is bounded above by $\mathrm{O}(1/N)$. Therefore, we have $\mathrm{O}(1/N^2)$ as upper bound in this case.

B: We do not select x in line 4, get some child x_{N+1} with j zeros, where $k \geq j > i$ holds, and select x for replacement in line 7. We have $\mathrm{O}(k/n)$ as upper bound on the probability for such a mutation and $\mathrm{O}(1/N)$ as upper bound for the final selection. This yields $\mathrm{O}(k/(Nn))$ as upper bound in this case.

C: We do not select x in line 4 and get some child x_{N+1} with more than k bits with value 0. We use 1 as upper bound on the probability of these two events. Then, x is selected for replacement in line 7. This event has probability $\mathrm{O}(1/(s+N))$, which serves as upper bound for event C, as well.

We see that we have

$$\mathrm{O}\left(\frac{1}{N^2} + \frac{k}{Nn} + \frac{1}{s+N}\right)$$

as upper bound as claimed.

We assume that $\sqrt{s} \geq N \geq n$ holds. Thus, we have $s \geq N^2 \geq Nn$, so that $\mathrm{O}(k/(Nn))$ is an upper bound on p_{S_i,S_k}. This yields

$$\frac{p_{S_i,S_{i-1}}}{p_{S_i,S_{i-1}} + p_{S_i,S_k}} = \mathrm{O}\left(\frac{1}{k}\right).$$

Thus, there exists some constant $c > 0$, such that

$$(c \cdot k)^k \cdot (k+1) \cdot Nn = \mathrm{O}\left((ck)^{k+1} Nn\right)$$

is an upper bound on the expected run time. □

Let $k = \Theta((\log n)/\log\log n)$ and $s = \Omega\left(n^{\log n}\right)$. Then we have the following result. Using any polynomial population size $N \geq n$, the GA has polynomial expected run time on $\text{SJUMP}_{s,k}$, whereas the $(1+1)$ EA has expected run time $\Theta\left(n^{(\log n)/\log\log n}\right)$, which is super-polynomial. Thus, there is a whole family of objective functions, where the expected run time can be tremendously reduced by employing a population instead of just one single point in the search space.

5 Conclusions

We investigated the question whether the use of a population can by itself be advantageous even without using crossover. We measured the computational cost by the number of function evaluations. We considered the maximization of pseudo-Boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$ and proved for one example that the appropriate use of a population can speed-up optimization by a factor of $\Omega(\sqrt{n})$. We could even strengthen this result, by proving, that the use of not too small population of polynomial size can reduce the expected run time from super-polynomial to polynomial. This is the first such result that has been rigorously proven.

It remains open whether functions can be found where the advantage due to the use of a population (without employing crossover) is even exponential. It would be interesting to find out Note, that in order to do a fair comparison one has to allow the (1+1) EA to make use of restart mechanisms.

Acknowledgments

This work is based upon an idea of Jonathan Rowe. He described a first intuition under what circumstances a carefully chosen EA may have an advantage compared to a random mutation hill-climber This intuition is the basis of the paper.

References

D. Arnold and H.-G. Beyer (2000). Local performance of the $(\mu/\mu\mathrm{I}, \lambda)$-ES in a noisy environment. In W. Spears and D. Whitley (Eds.), *FOGA*. to appear.

T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds.) (1997). *Handbook of Evolutionary Computation*. Institute of Physics.

H.-G. Beyer (1993). Some asymptotical results from the $(1, +\lambda)$-theory. *Evolutionary Computation 1*(2), 165–188.

H.-G. Beyer (1995a). Toward a theory of evolution strategies: On the benefit of sex — the $(\mu/\mu, \lambda)$-theory. *Evolutionary Computation 3*(1), 81–111.

H.-G. Beyer (1995b). Toward a theory of evolution strategies: The (μ, λ)-theory. *Evolutionary Computation 2*(4), 381–407.

S. Droste, T. Jansen, and I. Wegener (1998). A rigorous complexity analysis of the (1+1) evolutionary algorithm for linear functions with Boolean inputs. In D. B. Fogel, H.-P. Schwefel, T. Bäck, and X. Yao (Eds.), *Proceedings of the IEEE International Conference on Evolutionary Computation* (*ICEC '98*), 499–504. IEEE Press.

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

J. Garnier, L. Kallel, and M. Schoenauer (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation 7*(2), 173–203.

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimzation, and Machine Learning*. Addision-Wesley, Reading, MA.

T. Jansen and I. Wegener (1999). On the analysis of evolutionary algorithms — a proof that crossover really can help. In J. Nešetřil (Ed.), *Proceedings of the 7th Annual European Symposium on Algorithms* (*ESA '99*), Volume 1643 of *Lecture Notes in Computer Science*, 184–193. Springer, Berlin.

A. Juels and M. Wattenberg (1995). Hillclimbing as a baseline method for the evaluation of stochastic optimization algorithms. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 8*, 430–436. MIT Press.

M. Mitchell (1995). *An Introduction to Genetic Algorithms*. MIT Press.

H. Mühlenbein (1992). How genetic algorithms really work. Mutation and hillclimbing. In R. Männer and R. Manderick (Eds.), *Proceedings of the 2nd Parallel Problem Solving from Nature* (*PPSN II*), 15–25. North-Holland, Amsterdam, Niederlande.

S. Ronald (1998). Duplicate genotypes in a genetic algorithm. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (*ICEC '98*). IEEE Press, Piscataway, NJ.

J. Rowe (2000). Personal communication.

G. Rudolph (1997). How mutation and selection solve long path-problems in polynomial expected time. *Evolutionary Computation 4*(2), 195–205.

H.-P. Schwefel (1995). *Evolution and Optimum Seeking*. Wiley, New-York, NY.