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 Analysis of Evolutionary Algorithms — A Proof That Crossover Really Can Help

Thomas Jansen and Ingo Wegener

No. CI-51/98

Technical Report ISSN 1433-3325 December 1998

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 ANALYSIS OF EVOLUTIONARY ALGORITHMS — A PROOF THAT CROSSOVER REALLY CAN HELP

Thomas Jansen* and Ingo Wegener*

FB Informatik, LS 2, Univ. Dortmund, 44221 Dortmund, Germany
jansen, wegener@ls2.cs.uni-dortmund.de, fax: 0049-231-7552047

Abstract. There is a lot of experimental evidence that crossover is, for some functions, an essential operator of evolutionary algorithms. Nevertheless, it was an open problem to prove for some function that an evolutionary algorithm using crossover is essentially more efficient than evolutionary algorithms without crossover. In this paper, such an example is presented and its properties are proved.

1 Introduction

Stochastic search strategies have turned out to be efficient heuristic optimization techniques, in particular, if not much is known about the structure of the function and intense algorithmic investigations are not possible. The most popular among these algorithms are simulated annealing (van Laarhoven and Aarts 1987) and evolutionary algorithms, which come in great variety (evolutionary programming (Fogel, Owens, and Walsh 1966), genetic algorithms (Holland 1975, Goldberg 1989), evolution strategies (Schwefel 1995)). There is a lot of "experimental evidence" that these algorithms perform well in certain situations but there is a lack of theoretical analysis. It is still a central open problem whether, for some function, a simulated annealing algorithm with an appropriate cooling schedule is more efficient than the best Metropolis algorithm, i. e., a simulated annealing algorithm with fixed temperature (Jerrum and Sinclair 1997). Here, we solve a central open problem of similar flavor for genetic algorithms based on mutation, crossover, and fitness based selection. All these three modules are assumed to be essential but this has not been proved for crossover. Evolutionary algorithms without crossover are surprisingly efficient. Juels and Wattenberg (1994) report that even hill climbing (where the population size equals 1) outperforms genetic algorithms on nontrivial test functions. For a function called "long path", which was introduced by Horn, Goldberg, and Deb (1994), Rudolph (1997) has proved that a hill climber performs at least comparable to genetic algorithms. Indeed,

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

the following central problem considered by Mitchell, Holland, and Forrest (1994) is open:

- Define a family of functions and prove that genetic algorithms are essentially better than evolutionary algorithms without crossover.

One cannot really doubt that such examples exist. Several possible examples have been proposed. Forrest and Mitchell (1993) report for the well-known candidate called Royal Road function (Mitchell, Forrest, and Holland 1992) that some random mutation hill climber outperforms genetic algorithms. So the problem is still open (Mitchell and Forrest 1997). The problem is the difficulty to analyze the consequences of crossover, since crossover creates dependencies between the objects. Hence, the solution of the problem is a necessary step to understand the power of the different genetic operators and to build up a theory on evolutionary algorithms.

There are some papers dealing with the effect of crossover. Baum, Boneh, and Garrett (1995) use a very unusual crossover operator and a population of varying size which not really can be called genetic algorithm. Another approach is to try to understand crossover without fitness based selection. Rabinovich, Sinclair, and Wigderson (1992) model such genetic algorithms as quadratical dynamic systems, and Rabani, Rabinovich, and Sinclair (1998) investigate the isolated effects of crossover for populations. These are valuable fundamental studies. Here, we use a less general approach but we investigate a typical genetic algorithm based on mutation, uniform crossover, and fitness based selection. The algorithm is formally presented in Section 2. In Section 3, we prove, for the chosen functions, that algorithms without crossover necessarily are slow and, in Section 4, we prove that our genetic algorithm is much faster.

Definition 1. *The function* $\text{JUMP}_{m,n} : \{0,1\}^n \to \mathbb{R}$ *is defined by*

$$\text{JUMP}_{m,n}(x_1, \ldots, x_n) = \begin{cases} m + \|x\|_1 & \textit{if } \|x\|_1 \le n - m \textit{ or } \|x\|_1 = n \\ n - \|x\|_1 & \textit{otherwise} \end{cases}$$

where $\|x\|_1 = x_1 + \cdots + x_n$ *denotes the number of ones in* x*.*

The value of JUMP_m (the index n is usually omitted) grows linearly with the number of ones in the input but there is a gap between the levels $n - m$ and n. We try to maximize JUMP_m. Then, inputs in the gap are the worst ones. We expect that we have to create the optimal input $(1, 1, \ldots, 1)$ from inputs with $n - m$ ones. This "jump" is difficult for mutations but crossover can help. More precisely, we prove that time $\Omega(n^m)$ is necessary without crossover while a genetic algorithm can optimize JUMP_m with large probability in time $O(n^2 \log n + 2^{2m} n \log n)$ and the same bound holds for the expected time. The gap is polynomial for constant m and even superpolynomial for $m = \Theta(\log n)$.

2 Evolutionary Algorithms

We discuss the main operators of evolutionary algorithms working on the state space $S = \{0,1\}^n$ where we maximize a fitness function $f\colon S \to \mathbb{R}$. We use the operators initialization, mutation, crossover, and selection.

$X :=$ **initialize**(S, s). Choose randomly and independently s objects from S to form the population X.

$(y, b) :=$ **mutate**(X, p). Choose randomly an object $x \in X$ and, independently for all positions $i \in \{1, \ldots, n\}$, set $y_i := 1 - x_i$ with probability p and $y_i := x_i$ otherwise. Set $b := 0$, if $x = y$, and $b := 1$ otherwise.

The most common choice is $p = 1/n$ ensuring that, on average, one bit of x is flipped. The optimality of this choice has been proved for linear functions by Droste, Jansen, and Wegener (1998b). For some evolutionary algorithms it is not unusual to abstain from crossover. Evolutionary programming (Fogel, Owen, and Walsh 1966) and evolution strategies (Schwefel 1995) are examples. The so-called $(\mu + \lambda)$-evolution strategy works with a population of size μ. Then λ children are created independently by mutation and the best μ objects among the parents and children are chosen as next population. Ties are broken arbitrarily.

Genetic algorithms typically use crossover. For the function at hand, uniform crossover is appropriate.

$(y, b) :=$ **uniform-crossover-and-mutate**(X, p). Choose randomly and independently $x', x'' \in X$ and, independently for all positions $i \in \{1, \ldots, n\}$, set $z_i := x'_i$ with probability $1/2$ and $z_i := x''_i$ otherwise. Then $y := \text{mutate}(\{z\}, p)$. Set $b := 0$, if $y \in \{x', x''\}$, and $b := 1$ otherwise.

Ronald (1998) suggests to avoid duplicates in the population in order to prevent populations with many indistinguishable objects. We adopt this idea and only prevent replications (see below). Moreover, we use a variant of genetic algorithms known as steady state (Sarma and De Jong 1997). This simplifies the analysis, since, in one step, only one new object is created. Now we are able to describe our algorithm.

Algorithm 1.

1. $X :=$ **initialize**$(\{0,1\}^n, n)$.
2. Let r be a random number from $[0, 1]$ (uniform distribution).
3. If $r \leq 1/(n \log n)$,
   $(y, b) :=$ **uniform-crossover-and-mutate**$(X, 1/n)$.
4. If $r > 1/(n \log n)$,
   $(y, b) :=$ **mutate**$(X, 1/n)$.
5. Choose randomly one of the objects $x \in X$ with smallest f-value.
6. If $b = 1$ and $f(y) \geq f(x)$,
   $X := (X - \{x\}) \cup \{y\}$.
7. Return to Step 2.

Steps 5 and 6 are called steady state selection preventing replications. We do not care about the choice of an appropriate stopping rule by using as most of the authors the following complexity measure. We count the number of evaluations of f on created objects until an optimal object is created. In the following we discuss in detail the evolution strategies described above and the genetic algorithm described in Algorithm 1. We are able to obtain similar results for the following variants of the algorithm (details can be found in the full version).

1. Evolutionary algorithms without crossover may use subpopulations which work independently for some time and may exchange information sometimes.
2. Evolutionary algorithms without crossover as well as the genetic algorithm may choose objects based on their fitness (for mutation, crossover, and/or selection) as long as objects with higher fitness get a better chance to be chosen and objects with the same fitness get the same chance to be chosen.
3. The genetic algorithm may refuse to include any duplicate into the population.
4. The genetic algorithm may accept replactions as well as duplicates. In this case, all our results qualitatively still hold. The actual size of the upper bounds for the genetic algorithm changes in this case, though. At the end of Section 4 we discuss this in more detail.
5. The genetic algorithm may replace the chosen object by y even if $f(y) < f(x)$.

3 Evolutionary Algorithms without Crossover on $\mathbf{JUMP}_m$

Evolutionary algorithms without crossover create new objects by mutations only. If x contains i zeros, the probability of creating the optimal object equals $p^i(1-p)^{n-i}$. Let $m \leq (\frac{1}{2}-\varepsilon)n$. It follows by Chernoff's inequality that, for populations of polynomial size, the probability to have an object x, where $||x||_1 > n-m$, in the first population is exponentially small. Then, the expected time to reach the optimum is bounded below by $t_{m,n} = \min\{p^{-i}(1-p)^{i-n} \mid 0 \leq i \leq n-m\}$. This holds since we do not select objects x where $n-m < ||x||_1 < n$. It is obvious that $t_{m,n} = \Theta(n^m)$, if $p = 1/n$. The following result follows by easy calculations.

Proposition 1. *Let $m \leq (\frac{1}{2}-\varepsilon)n$ for some constant $\varepsilon > 0$. Evolutionary algorithms without crossover need expected time $\Omega(n^m)$ to optimize $JUMP_m$ if mutations flip bits with probability $1/n$. For each mutation probability p, the expected time is $\Omega(n^{m-c})$ for each constant $c > 0$.*

Droste, Jansen, and Wegener (1998a) have proved that, for population size 1 and $p = 1/n$, the expected time of an evolutionary algorithm on JUMP_m, $m > 1$, equals $\Theta(n^m)$.

4 The Genetic Algorithm as Optimizer of $\mathbf{JUMP}_m$

The main result of this paper is the following theorem.

Theorem 1. *Let m be a constant. With probability $1-e^{-\Omega(n)}$ the genetic algorithm creates an optimal object for* JUMP_m *within $O(n^2 \log n)$ steps.*

Proof. Our proof strategy is the following. We consider different phases of the algorithm and "expect" in each phase a certain behavior. If a phase does not fulfill our expectation, we estimate the probability of such a "failure" and may start the next phase under the assumption that no failure has occurred. We also assume to have not found an optimal object, since otherwise we are done. Finally, the failure probability can be estimated by the sum of the individual failure probabilities. The constants $c_1, c_2, c_3 > 0$ will be chosen appropriately.
Phase 0: Initialization. We expect to obtain only objects x where $\|x\|_1 \leq n-m$ or $\|x\|_1 = n$.
Phase 1: This phase has length $c_1 n^2 \log n$. We expect to create an optimal object or to finish with n objects with $n-m$ ones.
Remark: If Phase 1 is successful, the definition of the genetic algorithm ensures that the property is maintained forever.
Phase 2: This phase has length $c_2 n^2 \log n$. We expect to create an optimal object or to finish with objects with $n-m$ ones where the zeros are not too concentrated. More precisely, for each bit position i, there are at most $\frac{1}{4m}n$ of the n objects with a zero at position i.
Phase 3: This phase has length $c_3 n^2 \log n$. We expect that, as long as no optimal object is created, the n objects contain at each position $i \in \{1, \ldots, n\}$ altogether at most $\frac{1}{2m}n$ zeros. Moreover, we expect to create an optimal object.

Analysis of Phase 1. We apply results on the coupon collector's problem (see Motwani and Raghavan 1995). There are n empty buckets and balls are thrown randomly and independently into the buckets. Then the probability that, after $2n \ln n$ throws, there is still an empty bucket is $O(e^{-n})$. This result remains true if, between the throws, we may rename the buckets.

We consider the n^2 bit positions of the n objects as buckets. Buckets corresponding to zeros are called empty. The genetic algorithm never increases the number of zeros. Hence, we slow down the process by ignoring the effect of new objects created by crossover or by a mutation flipping more than one bit. We further slow down the process by changing the fitness to $\|x\|_1$ and waiting for ones at all positions. If a mutation flips a single bit from 0 to 1, we obtain a better object which is chosen. The number of empty buckets decreases at least by 1. If a single bit flips from 1 to 0, we ignore possible positive effects (perhaps we replace a much worse object). Hence, by the result on the coupon collector's problem, the failure probability is bounded by $e^{-\Omega(n^2)}$ after $4n^2 \ln n$ good steps.

A step is not good if we choose crossover (probability $1/n \log n$) or we flip not exactly one bit. The probability of the last event equals $1 - n \cdot \frac{1}{n} \cdot (1-\frac{1}{n})^{n-1}$ and is bounded by a constant $a < 1$. Hence, by Chernoff's bound, we can bound the probability of having enough good steps among $c_1 n^2 \ln n$ steps by $e^{-\Omega(n^2)}$, if c_1 is large enough.

Analysis of Phase 2. We have n objects with m zeros each. We cannot prove that the mn zeros are somehow nicely distributed among the positions. Good objects tend to create similar good objects, at least in the first phase. The population may be "quite concentrated" at the end of the first phase. Then, crossover cannot help. We prove that mutations ensure in the second phase that the zeros become "somehow distributed".

We only investigate the first position and later multiply the failure probability by n to obtain a common result for all positions. Let z be the number of zeros in the first position of the objects. Then $z \leq n$ in the beginning and we claim that, with high probability, $z \leq \frac{1}{4m}n$ at the end. We look for an upper bound $p^+(z)$ on the probability to increase the number of zeros in one round and for a lower bound $p^-(z)$ on the probability to decrease the number of zeros in one round. The number of zeros at a fixed position can change at most by 1 in one round. As long as we do not create an optimal object, all objects contain $n-m$ ones.

Let A^+ resp. A^- be the event that (given z) the number of zeros (at position 1) increases resp. decreases in one round. Then

$$A^+ \subseteq B \cup \left(\overline{B} \cap C \cap \left[\left(D \cap E \cap \bigcup_{1 \leq i \leq m-1} F_i^+ \right) \cup \left(\overline{D} \cap \overline{E} \cap \bigcup_{1 \leq i \leq m} G_i^+ \right) \right] \right)$$

with the following meaning of the events:

- B: crossover is chosen as operator.
- C: an object with a one at position 1 is chosen for replacement.
- D: an object with a zero at position 1 is chosen for mutation.
- E: the bit at position 1 does not flip.
- F_i^+: there are exactly i positions among the $(m-1)$ 0-positions $j \neq 1$ which flip and exactly i positions among the $(n-m)$ 1-positions which flip. We can exclude the case $i = 0$ which leads to a replication.
- G_i^+: there are exactly i positions among the m 0-positions which flip and exactly $i-1$ positions among the $(n-m-1)$ 1-positions $j \neq 1$ which flip.

Hence,

$$\begin{aligned}
p^+(z) \leq \frac{1}{n \log n} + &\left(1 - \frac{1}{n \log n}\right) \frac{n-z}{n} \\
&\left[\frac{z}{n} \sum_{i=1}^{m-1} \binom{m-1}{i} \binom{n-m}{i} \left(\frac{1}{n}\right)^{2i} \left(1 - \frac{1}{n}\right)^{n-2i} \right. \\
&\left. + \frac{n-z}{n} \sum_{i=1}^{m} \binom{m}{i} \binom{n-m-1}{i-1} \left(\frac{1}{n}\right)^{2i} \left(1 - \frac{1}{n}\right)^{n-2i} \right] \\
\leq \frac{1}{n \log n} + &\left(1 - \frac{1}{n \log n}\right) \frac{n-z}{n} \\
&\left[\frac{z}{n} (m-1)(n-m) \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{n-2} + O\left(\frac{m^2}{n^2}\right) \right. \\
&\left. + \frac{n-z}{n} \left(m \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{n-2} + O\left(\frac{m^2}{n^3}\right) \right) \right].
\end{aligned}$$

Similarly, we get

$$A^- \supseteq \overline{B} \cap \overline{C} \cap \left[\left(D \cap \overline{E} \cap \bigcup_{1 \le i \le m} F_i^- \right) \cup \left(\overline{D} \cap E \cap \bigcup_{1 \le i \le m} G_i^- \right) \right]$$

where

- F_i^-: there are exactly $i-1$ positions among the $(m-1)$ 0-positions $j \neq 1$ which flip and exactly i positions among the $(n-m)$ 1-positions which flip.
- G_i^-: there are exactly i positions among the m 0-positions which flip and exactly i positions among the $(n-m-1)$ 1-positions $j \neq 1$ which flip (if $i = 0$, we get the case of a replication).

Hence,

$$\begin{aligned} p^-(z) \ge & \left(1 - \frac{1}{n \log n}\right) \frac{z}{n} \left[\frac{z}{n} \sum_{i=1}^{m} \binom{m-1}{i-1} \binom{n-m}{i} \left(\frac{1}{n}\right)^{2i} \left(1 - \frac{1}{n}\right)^{n-2i} + \right. \\ & \left. \frac{n-z}{n} \sum_{i=1}^{m} \binom{m}{i} \binom{n-m-1}{i} \left(\frac{1}{n}\right)^{2i} \left(1 - \frac{1}{n}\right)^{n-2i} \right] \\ \ge & \left(1 - \frac{1}{n \log n}\right) \frac{z}{n} \left[\frac{z}{n}(n-m) \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{n-2} \right. \\ & \left. + \frac{n-z}{n} m(n-m-1) \frac{1}{n^2} \left(1 - \frac{1}{n}\right)^{n-2} \right]. \end{aligned}$$

Since m is a constant, we obtain, if $z \ge \frac{1}{8m}n$, that $p^-(z) \ge p^-(z) - p^+(z) = \Omega(\frac{1}{n})$ and it is also easy to see that $p^-(z) = O(\frac{1}{n})$. We call a step essential, if the number of zeros in the first position changes. The length of Phase 2 is $c_2 n^2 \log n$. The following considerations work under the assumption $z \ge \frac{1}{8m}n$. The probability that a step is essential is $\Omega(\frac{1}{n})$. Hence, for come $c_2' > 0$, the probability of having less than $c_2' n \log n$ essential steps, is bounded by Chernoff's bound by $e^{-\Omega(n)}$. We assume that this failure does not occur. Let $q^+(z)$ resp. $q^-(z)$ be the conditional probability of increasing resp. decreasing the number of zeros in essential steps. Then $q^+(z) = p^+(z)/(p^+(z) + p^-(z))$, $q^-(z) = p^-(z)/(p^+(z) + p^-(z))$, and $q^-(z) - q^+(z) = (p^-(z) - p^+(z))/(p^+(z) + p^-(z)) = \Omega(1)$. Hence, for some $c_2'' > 0$, the probability of decreasing the number of zeros by less than $c_2'' n$ is bounded by Chernoff's bound by $e^{-\Omega(n)}$. We obtain $c_2'' = 1$ by choosing c_2 large enough. But this implies that we have at some point of time less than $z^* = \frac{1}{8m}n$ zeros at position 1 and our estimations on $p^+(z)$ and $p^-(z)$ do not hold. We investigate the last point of time with z^* zeros at position 1. Then there are t essential steps left. If $t \le \frac{1}{8m}n$, it is sure that we stop with at most $\frac{1}{4m}n$ zeros at position 1. If $t > \frac{1}{8m}n$, we can apply Chernoff's bound and obtain a failure probability of $e^{-\Omega(n)}$. Altogether the failure probability is $ne^{-\Omega(n)} = e^{-\Omega(n)}$.

Analysis of Phase 3. First, we investigate the probability that the number of zeros at position 1 reaches $\frac{1}{2m}n$. For this purpose, we consider subphases starting at points of time where the number of zeros equals $\frac{1}{4m}n$. A subphase where the number of zeros is less than $\frac{1}{4m}n$ cannot cause a failure. The same holds for subphases whose length is bounded by $\frac{1}{4m}n$. In all other cases we can apply Chernoff's bound and the assumption $z \geq \frac{1}{4m}n$. Hence, the failure probability for each subphase is bounded by $e^{-\Omega(n)}$ and the same holds for all subphases and positions altogether. We create an optimal object if we perform crossover (probability $\frac{1}{n \log n}$) if the following mutation does not flip any bit (probability $(1-\frac{1}{n})^n$), if the chosen bit strings do not share a zero at some position (probability at least $\frac{1}{2}$, see below) and the crossover chooses at each of the $2m$ positions, where the objects differ, the object with the one at this position (probability $(\frac{1}{2})^{2m}$). We prove the open claim. We fix one object of the population. The m zeros are w.l.o.g. at the positions $1, \ldots, m$. There are at most $\frac{1}{2m}n$ objects with a zero at some fixed position $j \in \{1, \ldots, n\}$, altogether at most $\frac{1}{2}n$ colliding objects. Hence, the probability of choosing a second object without collision with the first one is at least $\frac{1}{2}$. Hence, the success probability is at least $\frac{1}{n \log n}(1-\frac{1}{n})^n 2^{-2m-1}$ and the failure probability for $c_3 n^2 \log n$ steps is bounded by $e^{-\Omega(n)}$.

Combining all estimations we have proved the theorem. □

Corollary 1. *Let m be a constant. The expected time of the genetic algorithm on $JUMP_m$ is bounded by $O(n^2 \log n)$.*

Proof. We remark that Theorem 1 can be proved for arbitrary starting populations instead of random ones. Then we add one phase of length $\Theta(n^2 \log n)$ at the beginning. With the analysis of Phase 1, it follows that the probability that Phase 0 ends with a population containing at least one object with i ones, where $n-m<i<n$, is $e^{-\Omega(n)}$. The expected number of superrounds consisting of the four phases is $1+e^{-\Omega(n)}$. □

In the following, we generalize our results to the case $m = O(\log n)$ where we reduce the crossover probability to $\frac{1}{n \log^3 n}$. Nothing has to be changed for Phase 1 (and Phase 0). In Phase 2, we obtain $p^-(z) - p^+(z) = \Omega(\frac{1}{n \log^2 n})$, $p^-(z) = O(\frac{\log n}{n})$, and $q^-(z) - q^+(z) = \Omega(\frac{1}{\log^3 n})$, if $z \geq \frac{1}{8m}n$. Then $O(n^2 \log^5 n)$ steps are enough to obtain the desired properties with a probability bounded by $e^{-\Omega(n^\delta)}$ for each $\delta < 1$. The same arguments work for the first property of Phase 3 as long as the length is polynomially bounded. In order to have an exponentially small failure probability for the event to create an optimal object, we increase the number of steps to $\Theta(n^2 2^{2m})$. If we are satisfied with a constant success probability, $\Theta(n(\log^3 n)2^{2m})$ steps are sufficient. We summarize these considerations.

Theorem 2. *Let $m = O(\log n)$. For each constant $\delta < 1$, with probability $1-e^{-\Omega(n^\delta)}$, the genetic algorithm creates an optimal object for $JUMP_m$ within $O(n^2(\log^5 n + 2^{2m}))$ steps. The expected run time is bounded by*

$$O(n \log^3 n(n \log^2 n + 2^{2m})).$$

If we allow replications as well as duplicates things change a little. We concentrate on the case where m is a constant. Nothing changes for phase 0. Our analysis of phase 1 remains valid, too. We remark that replications occur with probability at least $(1-\frac{1}{n\log n})(1-\frac{1}{n})^n$, so the tendency of good objects to create similar or equal objects in the first phase is enlarged. We change the length of phase 2 to $a(n)\cdot c_2n^3\log n$ and discuss the role of $a(n)$ later. We have to adapt our considerations to the circumstance that replications are allowed. For A^+ we have to include the event F_0^+, for A^- we include G_0^-. We still have $p^-(z)\geq p^-(z)-p^+(z)=\Omega(\frac{1}{nm^2})=\Omega(\frac{1}{n})$, but $p^-(z)=O(\frac{1}{n})$ does not hold, now. We consider only essential steps that still occur with probability $\Omega(\frac{1}{n})$. Let again $q^-(z)$ resp. $q^+(z)$ be the conditional probabilities for decreasing resp. increasing the number of zeros in one essential step. Now, we have $q^-(z)-q^+(z)=\Omega(\frac{1}{n})$. If in exactly d of $a(n)\cdot c_2'n^2\log n$ essential steps the number of zeros is decreased, we end up with $z+a(n)\cdot c_2'n^2\log n-2d$ zeros. Applying Chernoff's inequality yields that the probability not to decrease the number of zeros to at most $\frac{1}{4m}n$ in $a(n)\cdot c_2'n^2\log n$ essential steps is $e^{-\Omega(a(n)\log n)}$. Choosing $a(n)=\frac{n}{\log n}$ yields that with probability $1-e^{-\Omega(n)}$ the number of zeros is at most $\frac{1}{4m}n$ at all positions after phase 2, which now has a length of c_2n^4 steps. With $a(n)=\log n$, phase 2 needs only $c_2n^3\log^2 n$ steps, but the probability of a failure is increased to $e^{-\Omega(\log^2 n)}$, which is still subpolynomial. In Phase 3 replications change the probabilities for changing the number of zeros in the same way as in phase 2. Therefore, we can adapt our proof to the modified algorithm the same way as we did for phase 2. Since we are satisfied, if the number of zeros is not increasing too much, compared to phase 2, where we need a decreasement, it is not necessary to adjust the length of phase 3. We conclude that a genetic algorithm that allows replications finds a optimum of JUMP_m, for constant m, in $O(a(n)n^3\log n)$ steps with probability $e^{-\Omega(a(n)\log n)}$. We remark that the analysis of this variant of a genetic algorithm can be adapted to $m=O(\log n)$, too.

5 Conclusion

Evolutionary and genetic algorithms are often used in applications but the theory on these algorithms is in its infancy. In order to obtain a theory on evolutionary and genetic algorithms, one has to understand the main operators. This paper contains the first proof that, for some function, genetic algorithms with crossover can be much more efficient (polynomial versus superpolynomial) than all types of evolutionary algorithms without crossover. The specific bounds are less important than the fact that we have analytical tools to prove such a result. The difference in the behavior of the algorithms can be recognized in experiments already for small parameters, e. g. $n=50$ and $m=3$.

References

Baum, E. B., Boneh, D., and Garret, C.: On genetic algorithms. In Proceedings of the 8th Conference on Computational Learning Theory (COLT '95), (1995) 230–239.

Droste, S., Jansen, Th., and Wegener, I.: On the Analysis of the $(1+1)$ Evolutionary Algorithm. Tech. Report CI-21/98. Collaborative Research Center 531, Reihe Computational Intelligence, Univ. of Dortmund, Germany, (1998).

Droste, S., Jansen, Th., Wegener, I.: A rigorous complexity analysis of the $(1+1)$ evolutionary algorithm for separable functions with Boolean inputs. Evolutionary Algorithms **6(2)** (1998) 185–196.

Fogel, L. J., Owens, A. J., and Walsh, M. J.: Artificial Intelligence Through Simulated Evolutions. (1966) Wiley, New York.

Forrest, S. and Mitchell, M.: Relative building block fitness and the building block hypothesis. In D. Whitley (Ed.): Foundations of Genetic Algorithms 2, (1993) 198–226, Morgan Kaufmann, San Mateo, CA.

Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning. (1989) Addison Wesley, Reading, Mass.

Holland, J. H.: Adaption in Natural and Artificial Systems. (1975) Univ. of Michigan.

Horn, J., Goldberg, D. E., and Deb, K.: Long Path problems. In Y. Davidor, H.-P. Schwefel, and R. Männer (Eds.): Parallel Problem Solving from Nature (PPSN III), (1994) 149–158, Springer, Berlin, Germany.

Jerrum, M. and Sinclair, A.: The Markov Chain Monte Carlo method: An approach to approximate counting and integration. In D. S. Hochbaum (Ed.): Approximation Algorithms for NP-hard Problems. (1997) 482–520, PWS Publishers, Boston, MA.

Juels, A. and Wattenberg, M.: Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms, Tech. Report CSD-94-834, (1994), Univ. of California.

van Laarhoven, P. J. M. and Aarts, E. H. L.: Simulated Annealing. Theory and Applications, (1987), Reidel, Dordrecht, The Netherlands.

Mitchell, M. and Forrest, S.: Royal Road functions. In T. Bäck, D. B. Fogel and Z. Michalewicz (Eds.): Handbook of Evolutionary Computation, (1997) B2.7:20–B2.7:25, Oxford University Press, Oxford UK.

Mitchell, M., Forrest, S., and Holland, J. H.: The Royal Road function for genetic algorithms: Fitness landscapes and GA performance. In F. J. Varela and P. Bourgine (Eds.): Proceedings of the First European Conference on Artificial Life, (1992) 245–254, MIT Press, Cambridge, MA.

Mitchell, M., Holland, J. H., and Forrest, S.: When will a genetic algorithm outperform hill climbing? In J. Cowan, G. Tesauro, and J. Alspector (Eds.): Advances in Neural Information Processing Systems, (1994), Morgan Kaufman, San Francisco, CA.

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

Rabani, Y., Rabinovich, Y., and Sinclair, A.: A computational view of population genetics. Random Structures and Algorithms **12(4)** (1998) 314–334.

Rabinovich, Y., Sinclair, A., and Wigderson, A.: Quadratical dynamical systems (preliminary version). In Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science (FOCS '92), (1992) 304–313, IEEE Press Piscataway, NJ.

Ronald, S.: Duplicate genotypes in a genetic algorithm. In Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98), (1998) 793–798, IEEE Press Piscataway, NJ.

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

Sarma J. and De Jong, K.: Generation gap methods. In T. Bäck, D. B. Fogel and Z. Michalewicz (Eds.): Handbook of Evolutionary Computation, (1997) C2.7, Oxford University Press, UK.

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