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 Expected Runtime and the Success Probability of Evolutionary Algorithms

Ingo Wegener

No. CI-98/00

Technical Report ISSN 1433-3325 September 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 EXPECTED RUNTIME AND THE SUCCESS PROBABILITY OF EVOLUTIONARY ALGORITHMS

Ingo Wegener*

FB Informatik, LS2, Univ. Dortmund, 44221 Dortmund, Germany
wegener@ls2.cs.uni-dortmund.de

Abstract. Evolutionary algorithms are randomized search heuristics whose general variants have been successfully applied in black box optimization. In this scenario the function f to be optimized is not known in advance and knowledge on f can be obtained only by sampling search points a revealing the value of $f(a)$. In order to analyze the behavior of different variants of evolutionary algorithms on certain functions f, the expected runtime until some optimal search point is sampled and the success probability, i.e., the probability that an optimal search point is among the first sampled points, are of particular interest. Here a simple method for the analysis is discussed and applied to several functions. For specific situations more involved techniques are necessary. Two such results are presented. First, it is shown that the most simple evolutionary algorithm optimizes each pseudo-boolean linear function in an expected time of $O(n \log n)$. Second, an example is shown where crossover decreases the expected runtime from superpolynomial to polynomial.

1 INTRODUCTION

Evolutionary algorithm is the generic term for a class of randomized search heuristics. These search heuristics are investigated and applied since the mid-sixties and more intensively since the mid-eighties (Fogel (1985), Goldberg (1989), Holland (1975), Rechenberg (1994), or Schwefel (1995)). Despite many successes in applications people working in the area of efficient algorithms (including graph theoretic methods) have ignored evolutionary algorithms. The main reasons for this are the following ones. Many papers on evolutionary algorithms are quite imprecise and claim too general results which then are not proved rigorously. In particular, general evolutionary algorithms are mixed with hybrid algorithms for special problems which use ideas from efficient algorithms for the considered problem. Finally, an analysis of search heuristics for the most often considered problems is often too difficult. One of the few successful approaches is the analysis of the Metropolis algorithm for the graph bisection problem by Jerrum and Sorkin (1998). In order to understand the complicated stochastic process behind

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

evolutionary algorithms and related search heuristics like simulated annealing, Rabani, Rabinovich, and Sinclair (1998) and Rabinovich, Sinclair, and Wigderson (1992) have investigated isolated features of these algorithms. These results are fundamental for a well-founded theory but they do not contain any results on the behavior of evolutionary algorithms when applied to certain specific functions. These papers have been ignored in the community working on evolutionary algorithms.

We strongly believe that the analysis of randomized search heuristics is an interesting area which should be part of the area of efficient algorithms.

In order to distinguish general evolutionary algorithms from specialized ones, we introduce and discuss in Section 2 the scenario of black box optimization. In Section 3, we present those variants of evolutionary algorithms which are investigated in this paper. In Section 4, we discuss a general and simple method to analyze the behavior of evolutionary algorithms on certain functions. This method leads in surprisingly many situations to quite precise results. In order to prove that a variant of evolutionary algorithms may have a specific property, it is useful to look for example functions where this method of analysis works. However, this method has a limited power. For specific problems one has to develop specific tools which hopefully will turn out to be useful in further situations. In Section 5, we prove that the simplest evolutionary algorithm is efficient on all pseudo-boolean linear functions and, in Section 6, it is proved that crossover, although unnecessary for many problems, can decrease the expected runtime of an evolutionary algorithm from superpolynomial to polynomial.

2 RANDOMIZED SEARCH HEURISTICS IN BLACK BOX OPTIMIZATION

The scenario in the area of efficient algorithms is the following one. We are concerned with a specific and well-defined optimization problem. The structure behind this problem is investigated and this leads to a specialized algorithm which then is analyzed and implemented.

However, there are situations where this scenario is not adequate. If a problem does not belong to the class of problems investigated in textbooks or scientific papers, people may not have the time, skills or other resources to develop a specialized algorithm. Moreover, the functions f to be optimized may not be given in a clearly structured way. In order to optimize a technical system it may be the only possibility to perform or simulate an experiment to evaluate f on a specific input a. In these situations one needs randomized search heuristics with a good behavior on typical problems.

Such situations are captured by the scenario of black box optimization. The problem is to maximize an unknown function $f : S \rightarrow \mathbb{R}$. Here we only consider pseudo-boolean functions $f : \{0,1\}^n \rightarrow \mathbb{R}$. The algorithm can choose the first or more generally the t-th search point based only on the current knowledge namely the $t-1$ first search points together with their f-values.

This scenario has some unusual features. The randomized search strategy is not developed for a specific function. Hence, there will be no strategy which is superior to all the other ones. The aim is to develop a strategy with a good behavior on many problems, in particular, those problems expected to be considered in applications.

Moreover, it should be obvious that algorithms designed for a special class of functions should be better than a search heuristic for black box optimization when applied to the same class of functions. Since black box optimization is a typical scenario in applications, the design and analysis of randomized search heuristics is well motivated. The results reveal some knowledge on the type of problems where the given heuristic has good properties.

In black box optimization, the algorithm does not "know" if the best point ever seen is optimal. Hence, a stopping criterion is applied. After the algorithm has been stopped, the best of the sampled search points is presented as a solution. We abstract from this problem and consider search heuristics as infinite stochastic processes. We are interested in the random variable $T_{A,f}$ describing for the algorithm A and the function f the first point of time when an optimal input is sampled. The expected runtime $R_{A,f}$ of A for f is the expected value of $T_{A,f}$. Another important parameter is the success probability $S_{A,f,t}$ that A applied to f has sampled an optimal input within the first t steps. It is possible that $R_{A,f}$ increases exponentially while the success probability for a polynomial number of steps is not too small (constant or $1/poly(n)$). Then a multistart variant of A is efficient for f.

3 SOME VARIANTS OF EVOLUTIONARY ALGORITHMS

The most simple variant of an evolutionary algorithm is the so-called (1+1)EA working with a population size of 1 and, therefore, without crossover.

The first search point x is chosen randomly in $\{0,1\}^n$. This search point x is also the actual one. Later, a new search point x' is obtained from the actual one by mutation, i.e., the bits x'_i are considered independently and x'_i differs from x_i with the mutation probability $p_m(n)$. The default value of $p_m(n)$ equals $1/n$ ensuring that the expected number of flipping bits equals 1. The new point x' becomes the actual one iff $f(x') \geq f(x)$. Otherwise, x remains actual. The notation (1+1) describes that we select among one parent and one child. It will turn out that $p_m(n) = 1/n$ often is a good choice but sometimes other values are better. There are methods to find a good value of $p_m(n)$ during the search process (Bäck (1993, 1998)). However, a quite simple dynamic version called dynamic (1+1)EA turns out to have some good features (Jansen and Wegener (2000)). The first mutation step uses the parameter $p_m(n) = 1/n$. Before each new step $p_m(n)$ gets doubled and a value larger than 1/2 is replaced immediately by $1/n$. This leads to phases of approximately $\log n$ steps where for each meaningful mutation probability p one step uses a mutation probability which is close to p.

The analysis of evolutionary algorithms without crossover is much easier than the analysis in the presence of the crossover operator. Although many people have claimed that crossover is a useful operator, Jansen and Wegener (1999) were the first to prove this for some well-chosen function (see Section 6). They work with a population size of n, i.e., the algorithm stores n search points and their f-values. The first n search points are chosen randomly and independently. With probability $p_c(n)$, the algorithm chooses randomly and independently two parents x' and x'' from the current population. Uniform crossover leads to a random child z where the bits z_i are chosen independently and $z_i = x'_i$ if $x'_i = x''_i$. Otherwise, $z_i = x'_i$ with probability $1/2$ and $z_i = x''_i$ otherwise. Then y is the result of mutation with parameter $1/n$ applied to z. With probability $1 - p_c(n)$, y is the result of mutation with parameter $1/n$ applied to a random parent from the current population. Let x be a random element of those elements of the current population which have the smallest f-value. The new element y replaces x iff $f(y) \geq f(x)$ and y is not a replication of (one of) its parent(s).

4 A SIMPLE METHOD FOR THE ANALYSIS OF MUTATION BASED EVOLUTIONARY ALGORITHMS AND SOME APPLICATIONS

Let $A, B \subseteq \{0,1\}^n$. Then $A <_f B$ if $f(x) <_f (y)$ for all $x \in A$ and $y \in B$. Let $A_1, \ldots, A_p$ be a partition of the search space $\{0,1\}^n$ such that $A_1 <_f A_2 <_f \ldots <_f A_p$ and A_p is the set of all optimal search points.

For $x \in A_i$ let $s(x)$ be the probability that mutation changes x into some $y \in A_j$ where $j > i$. and let $s(i) = \min\{s(x) | x \in A_i\}$. The expected time to leave A_i has an upper bound of $r(i) = 1/s(i)$. Hence, the expected runtime of the (1+1)EA on f is bounded above by the sum of all $r(i)$, $1 \leq i \leq p-1$. The bound can be improved by taking into account the probabilities $\pi(i)$ that the first search point lies in A_i. It is essential to choose a "good" partition of $\{0,1\}^n$. The number of sets should not be too large but, nevertheless, the probabilities $s(i)$ should not be too small. More involved applications of this technique consider also the probabilities that the search may omit some sets of the partition by jumping from A_i to A_{i+d} for some $d \geq 2$.

We look for an upper bound on the success probability $S_{f,t}$ of the (1+1)EA which implies a lower bound of $(1 - S_{f,t}) \cdot t$ on the expected runtime. We have no success within t steps if

- we start with a search point in $A_1 \cup \cdots \cup A_i$,
- no jump from some A_j to $A_{j+d} \cup \cdots \cup A_p$ takes place,
- and less than $(p-i)/(d-1)$ times a change of the A-set takes place.

Hence, we have to prove that long jumps have a very small probability and that $s^*(i) = \max\{s(x) | x \in A_i\}$ is small. (It is easy to generalize the upper and lower bound technique to other mutation based evolutionary algorithms.)

In the following, we analyze the (1+1)EA on several functions. A function $f : \{0,1\}^n \to \mathbb{R}$ is called unimodal if each non-optimal search point x has a neighbor y ($H(x,y) = 1$ for the Hamming distance H) such that $f(y) > f(x)$.

Theorem 1. *Let $f : \{0,1\}^n \to \mathbb{R}$ be unimodal and let $I(f)$ be the size of the set image$(f) = \{f(x)|x \in \{0,1\}^n\}$. The expected runtime of the (1+1)EA is bounded above by $e \cdot n \cdot (I(f) - 1)$.*

Proof. We partition $\{0,1\}^n$ into the sets of inputs with the same f-value. Because of the unimodality of f there is for each non-optimal x a mutation changing one bit of x and leading to an improvement. Hence, $s(x) \geq \frac{1}{n}(1 - \frac{1}{n})^{n-1} \geq \frac{1}{e \cdot n}$ and $r(i) \leq e \cdot n$ for all non-optimal A-sets. □

The function LO_n (LEADING ONES) counts the length of the longest prefix of x consisting of ones only. LO_n is unimodal and $I(\mathrm{LO}_n) = n + 1$. The upper bound of Theorem 1 for LO_n equals $e \cdot n^2$.

Theorem 2. *The expected runtime of the (1+1)EA on LO_n equals $\Theta(n^2)$.*

Sketch of Proof. (Droste, Jansen, and Wegener (2000)).
With a probability of at least $1 - \frac{1}{n}$ the initial search point has at most $\log n$ leading ones. The probability that the f-value is increased during one step equals $1/n$, since one specific bit has to flip. The suffix of the actual point x behind the prefix 1^i0 is a random string. Hence, the f-value increases in a successful step with probability $(1/2)^j$ by $j+1$. The result follows by an appropriate application of Chernoff bounds (Hagerup and Rüb (1989)). □

A function $f : \{0,1\}^n \to \mathbb{R}$ is called linear if $f(x) = w_1x_1 + \cdots + w_nx_n$ for some weights. The linear function ONEMAX_n is defined by $w_1 = \cdots = w_n = 1$ and the linear function BV_n (BINARY VALUE) by $w_i = 2^{n-i}$. All linear functions are unimodal. The following results have been proved by Mühlenbein (1992), Rudolph (1997b), and Droste, Jansen, and Wegener (1998).

Theorem 3. *The expected runtime of the (1+1)EA on a linear function f where $w_i \neq 0$ for all i is bounded below by $\Omega(n \log n)$. It is bounded above for all linear functions by $O(n^2)$ and by $O(n \log n)$ for $ONEMAX_n$.*

Sketch of Proof. The lower bound follows by an application of the coupon collector's theorem (see Motwani and Raghaven (1995)). For the general upper bound we assume w.l.o.g. $w_1 \geq \cdots \geq w_n \geq 0$. Let $A_i = \{x | w_1 + \cdots + w_{i-1} \leq f(x) < w_1 + \cdots + w_i\}$, $1 \leq i \leq n+1$. Then $s(i) \geq \frac{1}{e \cdot n}$ for $i \leq n$. For ONEMAX_n even $s(i) \geq \frac{n+1-i}{e \cdot n}$, since inputs from A_i have $n + 1 - i$ neighbors with a larger function value. □

In Section 5, an upper bound of $O(n \log n)$ on the expected runtime of the (1+1)EA for an arbitrary linear function is established. The simple method used in this section seems to be too weak for this purpose. Unimodality is a natural property of many functions. Hence, people have looked for unimodal functions where the (1+1)EA needs exponential time. Such a function has to take exponentially many different function values and it has to be unlikely that one jumps over many possible function values. Long path functions (Horn, Goldberg, and Deb (1994)) have the property that there is a path (Hamming distance between a point and its successor equals one) of exponential length where the value of the function increases along the path. The values of points not on the path lead the (1+1)EA to the source of the path. Rudolph (1997a) has described explicitly paths which additionally have the property that for each $i \le n^{1/2}$ and each point x on the path there is at most one successor on the path whose Hamming distance to x equals i. Rudolph (1997a) has applied Theorem 1 to this function. Droste, Jansen, and Wegener (2000) have obtained an asymptotically matching lower bound. They have proved that the (1+1)EA meets the path with probability at least 1/2 in the first half of the path and that with high probability it needs $\Omega(d)$ improvement steps if its distance to the terminal of the path equals d.

Theorem 4. *There are explicitly defined unimodal functions where the expected runtime of the (1+1)EA grows exponentially. The success probability for some exponentially increasing time bound is exponentially small.*

The method also works for the dynamic (1+1)EA. If the mutation probability $1/n$ is good, we expect not to loose more than a factor of $O(\log n)$.

Theorem 5. *The expected runtime of the dynamic (1+1)EA is bounded by*

- $O(n(\log n)I(f))$ *for unimodal functions,*
- $\Theta(n^2 \log n)$ *for* LO_n*,*
- $O(n^2 \log n)$ *for all linear functions,*
- $O(n \log^2 n)$ *for* $ONEMAX_n$*.*

We omit the proofs due to Jansen and Wegener (2000). They also have shown that the dynamic (1+1)EA can be very useful. We explain their results not in detail. They have defined a function PJ_n (path and jump) with the following features. There is an island I of optimal points (all vectors with $\log n$ ones and only zeros at the first $2\log n$ positions), a path P of length $n/4$ from $1^{n/4}0^{3n/4}$ to 0^n consisting of all $1^i 0^{n-i}$, and a bridge B of all x with $n/4$ ones. Outside $I \cup P \cup B$ the function PJ_n is defined such that one efficiently finds the bridge but not the island or the path. The PJ-value on the bridge increases with the number of ones among the first $n/4$ positions. Hence, we look for $1^{n/4}0^{3n/4}$, the source of P. For this purpose, mutation probabilities of size $1/n$ are good. The same holds on the path from $1^{n/4}0^{3n/4}$ to 0^n. It is very unlikely to find the island before having reached 0^n. Then a jump to a point in Hamming distance $\log n$ is necessary and mutation probabilities of size $(\log n)/n$ are good.

Theorem 6. *Let* $p_m(n) = \frac{\alpha(n)\ln n}{n}$*. If* $\alpha(n) \to \infty$ *or* $\alpha(n) \to 0$ *for* $n \to \infty$*, the probability that the* $(1+1)$*EA on* PJ_n *needs a superpolynomial number of steps converges to 1. If* $\alpha(n) = c$*, the expected runtime is* $O(n^{2+c}\ln^{-1} n + n^{c-\log c-\log\ln 2})$ *which is* $O(n^{2.361})$ *for* $c = 1/(4\ln 2)$*. The expected runtime of the dynamic* $(1+1)$*EA on* PJ_n *is* $O(n^2 \log n)$ *and the success probability is* $1 - e^{-\Omega(n)}$ *if* $t \ge dn^3 \log n$ *for some constant* d*.*

Until now we have seen examples where steps with bad values of $p_m(n)$ have neither positive nor negative effects. However, there are examples where bad values of $p_m(n)$ lead to a disaster. There is a function PWT_n (path with a trap) with a path P of polynomial length and a trap T. PWT_n takes its optimal value at the end of P, the value is increasing along the path but the values in T are second best. All values outside $P \cup T$ are smaller and are defined in such a way that it is very likely that $P \cup T$ is reached at the beginning of P. For $p_m(n) = 1/n$ it is very likely that we follow the path to the optimum without reaching the trap which is too far away (distance approximately $\log n$). Using the dynamic $(1+1)$ EA there are steps with mutation probabilities close to $(\log n)/n$. Since the trap is large enough it is likely to reach the trap. The probability to return to the path is reduced to a jump to the unique optimum which is quite unlikely.

5 THE (1+1)EA ON LINEAR FUNCTIONS

Theorem 7. *The expected runtime of the* $(1+1)EA$ *on a linear function is* $O(n \log n)$.

Sketch of Proof. (Droste, Jansen, and Wegener (1998, 2000)).
We have to improve the simple $O(n^2)$ bound of Theorem 3. W.l.o.g. $w_1 \geq w_2 \geq \ldots \geq w_n > 0$. It is possible that the Hamming distance from the unique optimum 1^n increases during the application of the $(1+1)$EA. E.g., for BV_n 10^{n-1} is better than 01^{n-1}. The following technical trick is well-known from the analysis of efficient algorithms. The $(1+1)$EA works on some linear function f but we investigate its behavior w.r.t. to some other function, a so-called potential function.

It has turned out that one linear function can serve as a potential function for all linear functions, namely the linear function val_n with the weights $w_1 = \cdots = w_{n/2} = 2$ and $w_{n/2+1} = \cdots = w_n = 1$. Obviously, $I(val_n) = (3/2)n + 1$. we are investigating the $(1+1)$EA working on f. A step is called successful if it changes the actual point. Let $t(x)$ be the random number of successful steps until the $(1+1)$EA on f starting at x samples for the first time a point x' where $\text{val}_n(x') > \text{val}_n(x)$. The main step is the proof that $E(t(x)) \leq c$ for a constant c which does not depend on x. Since val_n is close enough to ONEMAX_n, this result implies the upper bound of $O(n \log n)$ on the expected runtime by bounding the expected number of unsuccessful steps similarly as in the proof of Theorem 3 for ONEMAX_n.

The claim $E(t(x)) \leq c$ is hard to prove. Let $x \in \{0,1\}^n$ and let x' be the random point produced by the $(1+1)$EA on f by a successful step. Let S be the random set of all i where $x_i = 0$ and $x'_i = 1$. Then $S \neq \emptyset$ and $D_S(x) = \text{val}(x') - \text{val}(x)$ describes for some fixed set S the random gain w.r.t. val. The random variable $D_S(x)$ is difficult to analyze. Therefore, a random variable $D^*_S(x)$ taking integer values is defined such that $D^*_S(x) \leq 1$ and $\text{Prob}(D^*_S(x) \leq r) \geq \text{Prob}(D_S(x) \leq r)$ for all r. Then the existence of a constant $d > 0$ (which neither depends on x nor on S) such that $E(D^*_S(x)) > d$ is proved. These steps need an involved case inspection. Finally, a slight generalization of Wald's identity is proved to obtain the result. □

This result indicates the difficulty of obtaining tight results on the expected run-time of simple variants of evolutionary algorithms on simple functions. However, typical tools used for the analysis of randomized algorithms have to be applied also for the analysis of evolutionary algorithms.

6 THE HARD CASE – EVOLUTIONARY ALGORITHMS WITH CROSSOVER

Evolutionary algorithms without crossover are much easier to analyze than those with crossover. Crossover is defined for two points. The success of crossover depends in the case of uniform crossover on the number and the position of those bits where the parents differ. For many functions, among them many of those discussed in this paper, crossover seems to be unable to improve evolutionary algorithms. Jansen and Wegener (1999) were the first to prove for a specific function the usefulness of crossover. Let $\mathrm{JUMP}_{m,n}(x) = ||x||_1 = x_1 + \cdots + x_n$ if $0 \leq ||x||_1 \leq n - m$ or $||x||_1 = n$ and let $\mathrm{JUMP}_{m,n}(x) = -||x||_1$ otherwise. The set of x where $\mathrm{JUMP}_{m,n}(x) < 0$ is called the hole. We are only interested in the case $m = O(\log n)$. A random population of polynomial size contains with overwhelming probability no optimal point and no point in the hole . If the selection procedure rejects points in the hole and if we only use mutation, an evolutionary algorithm needs with overwhelming probability $\Omega(n^{m-\epsilon})$ steps for the optimization of $\mathrm{JUMP}_{m,n}$ ($\varepsilon > 0$ constant).

Theorem 8. *Let A be the evolutionary algorithm with crossover described in Section 3 and let $p_c(n) = 1/(n \log n)$.*

a) Let m be a constant and $f = \mathrm{JUMP}_{m,n}$. Then
- $S_{A,f,t} = 1 - \Omega(n^{-k})$ *if* $t \geq c_k n^2 \log n$ *for some constant* c_k,
- $S_{A,f,t} = 1 - e^{-\Omega(n)}$ *if* $t \geq cn^3$ *for some constant* c, *and*
- $R_{A,f} = O(n^2 \log n)$.

b) Let $m = O(\log n)$ and $f = \mathrm{JUMP}_{m,n}$. Then
- $S_{A,f,t} = 1 - e^{-\Omega(n^{\delta})}$ *if* $t \geq c(n^3 + n^2(\log^5 n + 2^{2m})) = \mathrm{poly}(n)$ *for some constants* $\delta > 0$ *and* c, *and*
- $R_{A,f} = O(n \log^3 n(n \log^2 n + 2^{2m})) = \mathrm{poly}(n)$.

Sketch of Proof. (Jansen and Wegener (1999)).
With overwhelming probability, the evolutionary algorithm produces within $O(n^2 \log n)$ steps a population of search points where each search point contains $n - m$ or n ones. This can be proved by a generalization of the solution to the coupons collector's problem. Crossover is not necessary for this but it also does not cause difficulties.

Either we have found the optimum or we have a population of individuals with $n - m$ ones. This property does not change in the future. Let us assume (incorrectly) that the individuals are random and independent strings with $n-m$ ones. If we choose two of them randomly, it is very likely that they have no common position with a zero. Then crossover creates with probability 2^{-2m} the optimal vector 1^n and mutation does not destroy this string with probability $(1-\frac{1}{n})^n \approx e^{-1}$.

The problem is that crossover and also mutation have the tendency to create positively correlated individuals. If all members of the population are equal (or each pair of individuals has many common zeros), crossover cannot destroy these zeros and this job again has to be done by mutation. Hence, we need crossover for a quick jump over the hole but crossover decreases the diversity in the population. This is the reason why we assume that $p_c(n)$ is very small. (It is conjectured that the theorem also holds for constant values of $p_c(n)$ but we cannot prove this.)

Now a tedious analysis is necessary to ensure that with overwhelming probability we have a population whose diversity is large enough. More precisely, it is proved that for a long period of time the following statement is true for each bit position i. The number of individuals with a zero at position i is bounded above by $\frac{1}{2m}n$. This property implies that, with probability at least 1/2, two individuals chosen randomly from the current population have no common zero. Since $p_c(n)$ is small enough, we may assume during these calculations that, if crossover does not create an optimal individual, it only has bad effects. $\square$

References

1. Bäck, T. (1993). Optimal mutation rates in genetic search. 5. Int. Conf. on Genetic Algorithms (ICGA), 2-8.
2. Bäck, T. (1998). An overview of parameter control methods by self-adaptation in evolutionary algorithms. Fundamenta Informaticae 34, 1-15.
3. Droste, S., Jansen, T., and Wegener, I. (1998). A rigorous complexity analysis of the $(1+1)$ evolutionary algorithm for linear functions with boolean inputs. ICEC '98, 499-504.
4. Droste, S., Jansen, T., and Wegener, I. (2000). On the analysis of the $(1+1)$ evolutionary algorithm. Submitted: Theoretical Computer Science.
5. Fogel, D. B. (1995). Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press.
6. Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley.
7. Hagerup, T. and Rüb, C. (1989). A guided tour of Chernoff bounds. Information Processing Letters 33, 305-308.
8. Holland, J. H. (1975). Adaption in Natural and Artificial Systems. Univ. of Michigan.
9. Horn, J., Goldberg, D. E., and Deb, K. (1994). Long path problems. 3. Int. Conf. on Parallel Problem Solving from Nature (PPSN), LNCS 866, 149-158.
10. Jansen, T. and Wegener, I. (1999). On the analysis of evolutionary algorithms – a proof that crossover really can help. ESA '99, LNCS 1643, 184-193.
11. Jansen, T. and Wegener, I. (2000). On the choice of the mutation probability for the $(1+1)$ EA. Submitted: PPSN 2000.

12. Jerrum, T. and Sorkin, G. B. (1998). The Metropolis algorithm for graph bisection. Discrete Applied Mathematics 82, 155-175.
13. Motwani, R. and Raghavan, P. (1995). Randomized Algorithms. Cambridge Univ. Press.
14. Mühlenbein, H. (1992). How genetic algorithms really work. I. Mutation and hill-climbing. 2. Int. Conf. on Parallel Problem Solving from Nature (PPSN), 15-25.
15. Rabani, Y., Rabinovich, Y., and Sinclair, A. (1998). A computational view of population genetics. Random Structures and Algorithms 12, 314-334.
16. Rabinovich, Y., Sinclair, A., and Widgerson, A. (1992). Quadratical dynamical systems. 33. FOCS, 304-313.
17. Rechenberg, I. (1994). Evolutionsstrategie '94. Frommann-Holzboog, Stuttgart.
18. Rudolph, G. (1997a). How mutations and selection solve long path problems in polynomial expected time. Evolutionary Computation 4, 195-205.
19. Rudolph, G. (1997b). Convergence Properties of Evolutionary Algorithms. Ph.D. Thesis. Dr. Kovač, Hamburg.
20. Schwefel, H.-P. (1995). Evolution and Optimum Seeking. Wiley.