Genetic Algorithms and the $O(n \ln n)$ Complexity on Selected Test Functions

Ralf Salomon
AI Lab, Computer Science Department, University of Zürich
Winterthurerstrasse 190, 8057 Zurich, Switzerland
FAX: +41-1-363 00 35; Email: salomon@ifi.unizh.ch

Abstract

Genetic algorithms (GAs) are useful in the field of continuous parameter optimization. Many results indicate that, due to their robustness and their ability of escaping from local optima, GAs are especially suitable for optimizing multimodal functions that contain millions of misleading local optima. The existing theory is based on spherical symmetric functions and suggest an $O(n \ln n)$ complexity for particular GAs. However, the widely-used test functions are not spherical symmetric but decomposable and recent results indicate that GAs fail under a a simple rotation of the coordinate system. This paper shows that, when applied to a decomposable function with n independent parameters, any GA that features an elitist selection scheme, a broad mutation operator, and uses a mutation probability $m_p = 1/n$, converges with an $O(n \ln n)$ complexity; it is not necessary to use a highly optimized genetic algorithms.

1 Introduction

Genetic algorithms (GAs) are a class of randomized optimization procedures that are based on natural selection and genetics. The basic scheme of a simple genetic algorithm (SGA) works on a population of fixed bit strings. Each iteration, also called generation, is performed in four steps. First, it selects a specified number of individuals as parents for the next generation.

Second, it generates two new offspring by selecting two different parents and applying mutation (e.g., bit inversion or adding small random numbers) to each of them. Third, it recombines both offspring by exchanging a subset of corresponding parameters. Fourth and finally, it evaluates and stores the fitness of each offspring. A thorough discussion of GAs and its applications can be found in Goldberg (1989) and Srinivas (1994).

Continuous parameter optimization is an interesting application domain for GAs. The ability of escaping from misleading local optima is a distinctive property of genetic algorithms. Among other things it has been claimed in Goldberg (1989) that "Because genetic algorithms operate at the coding level, they are difficult to fool even when the function may be difficult for traditional schemes." and in Mühlenbein et al. (1994) "the theory predicts that the multimodality of *[Schwefel's]* function is more or less noise for the algorithm.". Therefore, GAs are especially suited for optimizing multimodal functions (see also DeJong 1975, Goldberg 1989, Mühlenbein et al. 1993 and 1994, and Srinivas 1994). A multimodal function contains one global optimum and additional misleading local optima. Mühlenbein et al. (1993,1994) present a theory which suggests a $O(n \ln n)$ complexity for the breeder genetic algorithm (BGA) when applied to a variety of multimodal functions. This complexity is significantly lower than the $O(\exp(n))$ complexity of the pure gradient descent method. The BGA's theory is based on the assumption "...that the fitness function is spherical symmetric." Mühlenbein et al. (1993). Thus, in a strict sense the theory given so far does hold only for spherical symmetric test functions. But most of the widely-used test functions are *not spherical symmetric*. Recent results presented in Salomon (1994) indicate that a GA fails under a simple rotation of the coordinate system; i.e., the algorithm prematurely stagnates when optimizing $f(\mathbf{M}\,\vec{x})$, with $\mathbf{M}$ as an orthogonal n-dimensional rotation matrix.

The goal of this paper is to demonstrate why *any* GA that features a broad mutation operator, an elitist selection scheme, and a mutation probability $m_p = 1/n$ necessarily converges at most with an $O(n \ln n)$ complexity when applied to decomposable functions; the complexity is independent of the specific selection scheme (e.g., truncation selection, roulette wheel selection), the actual mutation operator, the population size (e.g., even a population with only one individual would be sufficient), and no matter if recombination is used or not. The derivation of this convergence complexity is based on the decomposability property of the widely-used test functions, since efficient GAs implicitly exploit this assumption as well. In addition, the derivation presented in this paper is short and comprehensive.

The remainder of this paper is organized as follows. Section 2 presents an excerpt of well-known test functions, which can be considered as a more or less standardized benchmark test. Section 3 gives a brief discussion of the BGA, which works as a very efficient function optimizer. Section 4 introduces the concept of the broad mutation operator. As announced, the goal of Section 5 is to present the derivation of the convergence speed.

2 Continuous Parameter Optimization

In the field of continuous parameter optimization, an algorithm has to find the optimum of a given function with respect to a given set of continuous

Table 1: Standardized test functions with some domain limits defined by other authors.

	Function	Limits
1	$f_1(\vec{x}) = \sum_{i=1}^{3} x_i^2$	$-5.12 \leq x_i \leq 5.12$
2	$f_2(\vec{x}) = 100\,(x_1^2 - x_2)^2 + (1 - x_i)^2$	$-2.048 \leq x_i \leq 2.048$
3	$f_3(\vec{x}) = \sum_{i=1}^{5} \mathrm{integer}(x_i)$	$-5.12 \leq x_i \leq 5.12$
4	$f_4(\vec{x}) = \sum_{i=1}^{30} i\, x_i^4 + \mathrm{Gauss}(0,1)$	$-1.28 \leq x_i \leq 1.28$
5	$f_5(\vec{x}) = 0.002 + \sum_{j=1}^{25} \frac{1}{c_j + \sum_{i=1}^{2}(x_i - a_{ij})^6}$	$-65.536 \leq x_i \leq 65.536$
6	$f_6(\vec{x}) = 20 + \sum_{i=1}^{20} x_i^2 - \cos(2\,\pi\, x_i)$	$-5.12 \leq x_i \leq 5.12$
7	$f_7(\vec{x}) = \sum_{i=1}^{10} -x_i\, \sin(\sqrt{\lvert x_i \rvert})$	$-500 \leq x_i \leq 500$
8	$f_8(\vec{x}) = \sum_{i=1}^{5} 10^{i-1} x_i^2$	$-10.0 \leq x_i \leq 10.0$

parameters. The parameters are denoted by $x_i, 1 \leq i \leq n$, with n as the problem's dimensionality. Due to physical constraints and technical reasons, the search interval of each parameter is limited to a prespecified range $x_i \in [-\text{range}_i, \text{range}_i]$. It is sufficient to consider only minimization tasks, since the multiplication factor -1 converts any maximization problem into a minimization problem. Furthermore, it is sufficient to consider only symmetric search intervals; for asymmetrical intervals a simple transformation has to be used.

For evaluation and comparison purposes, the research community has developed a variety of test functions. Selected functions, which can be seen as a standardized benchmark test, are summarized in Table 1.

The first five functions of Table 1 are known as DeJong's test functions. The first four of them are unimodal, whereas the fifth function is multimodal. In addition, these test functions comprise other features such as a sharp ridge (f_2), piecewise continuity (f_3), and noise (f_4).

More recently, research interest has shifted to multimodal functions such as Rastrigin's function (f_6) and Schwefel's function (f_7). Rastrigin's function is highly multimodal and has its minimum at $\vec{x}_m = \vec{0}$ with $f_6(\vec{x}_m) = 0$. Most optimization procedures have problems finding this minimum, because the probability of making progress decreases rapidly as the minimum is approached; normally, they prematurely stagnate in its vicinity. Optimizing Schwefel's function, one encounters the problem that the global minimum is given by $\forall\, i\;\; x_i = 420.9687$ and that the second best minimum $x_j = -302.5253, \forall\, i\; i \neq j, x_i = 420.9687$ is far away from the global optimum (a detailed discussion can be found in Mühlenbein et al. (1991)).

Function f_8 is similar to function f_1; but f_8 features different eigenvalues along each axis and the graphs of constant function values appear as n-dimensional hyperellipses. Function f_8 is a classical test case for many optimization procedures (see, for example, Gill et al. 1981). For f_8-like functions it is well-known that the convergence speed crucially depends on the ratio of the smallest and largest eigenvalue.

All test functions of Table 1 have the following *decomposability* property:

$$f(\vec{x}) = \sum_{i=1}^{n} f^i(x_i) \qquad (1)$$

Thus, a function $f(\vec{x})$ is decomposable into a sum of n not necessaryly identical functions f^i. If a function f is decomposable, we say that its variables x_i are *independent*. It seems that GAs do not use the decomposability property at all. However, results presented in Salomon (1994) indicate that GAs implicitly exploit this property. Furthermore, the results indicate that GAs fail under a simple rotation of the coordinate system.

Function $f_2(x_1, x_2) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$ was first introduced by Rosenbrock (1960). Its minimum $\vec{x}_m = (1,1)$, $f(\vec{x}_m) = 0$ is located in a narrow curved valley. The orientation of this valley is not aligned with the coordinate axes. Even though Rosenbrock's function is unimodal and defined by only two parameters, it drastically slows down the highly optimized breeder genetic algorithm[1]. See Gill et al. (1981) for a thorough discussion of optimizing Rosenbrock's function.

3 A Typical GA: The Breeder Genetic Algorithm

There is not *one* genetic algorithm, but rather a variety of different variants. In the field of continuous parameter optimization, the breeder genetic algorithm (BGA) (Mühlenbein et al. 1993 and 1994) is one of the most efficient procedures. Its design is strongly influenced by the science of natural breeding. Due to its performance, the BGA is especially suited for optimizing multimodal functions such as f_6 and f_f. In addition, the BGA's behavior is supported by an extensive theory.

The breeder genetic algorithm works as follows. It maintains a population of size N. In each generation, it selects an arbitrary $T\%$ best individuals as parents for the next generation. This selection guarantees that the best

[1]Personal communication with Mühlenbein, September 30, 1994.

individual (parent) remains in the population (elitist selection). New offspring are produced by selecting two different parents and applying uniform recombination as well as mutation. At any locus i, uniform recombination exchanges the corresponding variable x_i of the two participating objects with a given probability $r_p = 0.5$. Mutation is established by the following random number generator[2]:

$$\sigma = s \sum_{i=0}^{15} \alpha_i \, 2^{-i} \qquad , \tag{2}$$

where each coefficient α_i is set to 1 with a probability of 1/16 and set to 0 with a probability of 15/16 respectively. In addition, it is guaranteed that at least one α_i is set to 1, and the sign s is randomly chosen from $\{-1, 1\}$ with equal probability. According to the following update rule, the BGA mutates each parameter x_i with a probability $m_p = 1/n$:

$$x_i \leftarrow x_i + 0.1 \, \text{range}_i \, \sigma \tag{3}$$

The BGA does not use any further crossover operator or any local hill climbing method. The mutation and recombination probabilities $m_p = 1/n$ and $r_p = 0.5$ respectively are currently standard in GAs.

4 Broad Mutation Operator

A mutation operator modifies one or more parameters x_i of a given object. The modified object (i.e., the offspring) appears in parameter space within a certain distance of the unmodified object (i.e., the parent). In one step, the mutation operator can reach points only within the distance given by the random number generator (2) and the update rule (3). The BGA's mutation operator is restricted to the interval $[-0.2 \, \text{range}_i, 0.2 \, \text{range}_i]$. This restriction is very efficient for functions such as f_1 to f_6. But it is not suitable when

[2]This approach is not very common, even though it is straightforward for this kind of application. In general, other procedures (DeJong 1975, Goldberg 1989, Mühlenbein et al. 1991, or Srinivas 1994) simply flip bits with a probability $m_p = 1/n$, where n denotes the total number of bits.

optimizing Schwefel's function. In Schwefel's function, the distance between the global optimum and the second best is approximately 723 whereas the mutation operator provides mutations only within a range of ± 200. In such a configuration, global convergence depends not only on the mutation operator, but also on recombination, the selection scheme, the size of the initial population, and the initialization process itself (see also Mühlenbein et al. 1993).

The limited mutation operator requires the BGA to maintain a minimal population size, which depends on the objective function as well as on the problem size. Furthermore, the BGA does not reliably converge to the global optimum, if the initial population is too small.

The problem explained above can be solved by a *broad* mutation operator that spawns numbers in the whole search interval of each parameter x_i. In other words, a broad mutation operator provides numbers at least in the interval $[-2\,\text{range}_i, 2\,\text{range}_i]$. The BGA could implement this by the following modified update rule

$$x_i \leftarrow x_i + \text{range}_i\, \sigma$$

Even though this change is small, it has a major impact on the subsequent analysis. Such a broad mutation operator is able to reach any desired point x_i^d from any arbitrary point x_i^p in one step with a finite probability $p > 0$. In addition, this mutation operator provides an excellent starting point for the derivation of the convergence speed and its complexity.

5 Convergence Speed and its Complexity

The following derivation of the convergence speed and its complexity makes the following four assumptions: (1) the objective function is decomposable, i.e., $f(\vec{x}) = \sum_{i=1}^{n} f^i(x_i)$, (2) the GA uses an elitist selection scheme (preserving the so far best solution), (3) the GA features an arbitrary broad mutation operator with probability $m_p = 1/n$, and (4) the GA features uniform recombination with the probability $r_p = 0.5$. In this stochastic context, convergence means that, with a prespecified probability p, the algorithm found at least one solution $\vec{x}_s$, $\|\vec{x}_s - \vec{x}_o\| \leq \epsilon$ within an arbitrary ϵ neighborhood of the optimum $\vec{x}_o$.

In the one-dimensional case, recombination has no effect; it either does nothing or it merely swaps the parameters of each object. In both cases, it does not create any new combination, because each object has only one parameter. Featuring a broad mutation operator, the GA is able to place any new offspring at any place in parameter space with a probability $p > 0$. Thus, it eventually converges to the global optimum. The convergence speed merely depends on the probability distribution of the mutation operator as well as on the one-dimensional objective function. But for any given combination, the number of required generations is bounded and, thus, the complexity is $O(1)$.

Turning to the n-dimensional case, it is helpful to not consider any recombination in the first step. A mutation probability of $m_p = 1/n$ implies that, on average, the GA modifies one variable per offspring. By modifying only one parameter per offspring, the GA decomposes the optimization of a *decomposable* n-dimensional function into a sequence of n independent one-dimensional tasks. Since the complexity of the one-dimensional case is $O(1)$, convergence in the n-dimensional case is given by the probability p that all parameters $x_i, 1 \leq i \leq n$ have been chosen at least once. The probability q_i that the particular variable x_i has not been chosen within g generations is given by $q_i = [(n-1)/n]^g$. Consequently, the probability p_i that it has been chosen at least once is $p_i = 1 - q_i = 1 - [(n-1)/n]^g$. Since the choice of each parameter x_i is independent of all other parameters, p can be expressed as follows:

$$
\begin{aligned}
p &= \prod_{i=1}^{n} p_i \\
p &= \prod_{i=1}^{n} \left[1 - \left(\frac{n-1}{n}\right)^g\right] \\
p &= \left[1 - \left(\frac{n-1}{n}\right)^g\right]^n
\end{aligned}
$$

In order to guarantee that n parameters have been chosen within g generations, the number of generations $g > n$ has to be larger than n. Thus, the term $[1 - ([n-1]/n)^g]^n$ can be approximated by $1 - n\,[(n-1)/n]^g$:

$$p = 1 - n\left(\frac{n-1}{n}\right)^g \qquad . \tag{4}$$

Solving (4) for the number of generations g leads to

$$g = \frac{\ln(1-p) - \ln n}{\ln(\frac{n-1}{n})} \qquad . \tag{5}$$

For arbitrary n and a constant probability p the complexity is given by (all constant terms vanish):

$$g = O\left(\frac{-\ln n}{\ln(\frac{n-1}{n})}\right) \qquad . \tag{6}$$

For large n, the value of $\ln(\frac{n-1}{n})$ approaches 0. A Taylor sequence of $\ln x$ at $x = 1$ leads to $\ln x \approx x - 1$. Substituting this approximation into (6) results in

$$\begin{aligned} g &= O\left(\frac{-\ln n}{\frac{n-1}{n} - 1}\right) \\ g &= O\left(\frac{-\ln n}{\frac{-1}{n}}\right) \\ g &= O\left(n\,\ln n\right) \qquad . \end{aligned} \tag{7}$$

Now, it should be noted why (7) holds, no matter how many steps k are required in the one-dimensional case. The derivation is based on the probability that each x_i has been chosen at least once. If a GA needs k steps in one dimension, then convergence is determined by the probability that each variable has been chosen at least k times. It should be obvious that, in this case, a GA needs at least $g = n\,k$ generations. To this end, the term $[(n-1)/n]^g$ in (4) has to be substituted by the partial sum of the binomial distribution $b(i; g, 1/n)$.

$$p = 1 - n\sum_{i=0}^{k-1} b(i; g, 1/n) = 1 - n\sum_{i=0}^{k-1}\binom{g}{i}\left(\frac{1}{n}\right)^g\left(\frac{n-1}{n}\right)^{g-i} \qquad . \tag{8}$$

The poisson distribution $p(i;\lambda=g/n)=e^{-\lambda}\,\lambda^i/(i!)$ is a good approximation of the binomial distribution. Substituting the poisson distribution into (8) leads to

$$\begin{aligned} p &= 1-n\sum_{i=0}^{k-1}\frac{e^{-\lambda}\,\lambda^i}{i!} = 1-n\sum_{i=0}^{k-1}\frac{e^{-\frac{g}{n}}\,\left(\frac{g}{n}\right)^i}{i!} \\ \ln\left(\frac{1-p}{n}\right) &= \ln\left[\sum_{i=0}^{k-1}\frac{e^{-\frac{g}{n}}\,\left(\frac{g}{n}\right)^i}{i!}\right] \\ \ln(1-p)-\ln n+\frac{g}{n} &= \ln\left[\sum_{i=0}^{k-1}\frac{\left(\frac{g}{n}\right)^i}{i!}\right] \qquad . \end{aligned} \tag{9}$$

This equation can be solved by the following two inequalities (a)

$$\begin{aligned} \ln(1-p)-\ln n+\frac{g}{n} &= \ln(1+\cdots)>0 \qquad \Rightarrow \\ \frac{g}{n} &> \ln n-\ln(1-p) \\ g &\geq O(n\,\ln n) \qquad , \end{aligned} \tag{10}$$

and (b), with $g\geq k\,n$

$$\begin{aligned} \ln(1-p)-\ln n+\frac{g}{n} &= \ln\left[\sum_{i=0}^{k-1}\left(\frac{g}{n}\right)^i\right] \\ \ln(1-p)-\ln n+\frac{g}{n} &= \ln\left[\frac{\left(\frac{g}{n}\right)^k-1}{\frac{g}{n}-1}\right]\leq C+(k-1)\ln\left(\frac{g}{n}\right) \qquad \Rightarrow \\ \frac{g}{n} &\leq \ln n-\ln(1-p)+C+(k-1)\ln\left(\frac{g}{n}\right) \qquad \Rightarrow \\ g &\leq O(n\,\ln n) \end{aligned} \tag{11}$$

and, thus, $g=O(n\,\ln n)$.

The final question is to what extent uniform recombination leads to an improved convergence speed. Uniform recombination generates new offspring by combining already optimized variables of different objects into new combinations. According to Goldberg (1989), schemata with a long defining length have a high probability of being destroyed by the recombination operator. In the context of uniform recombination, schemata can be considered as the number of already optimized variables. Thus, one can expect progress only if the schemata are short[3], or if both participating individuals have a high similarity with respect to their free variables. After the first phase, the population starts converging to a global equilibrium, where all individuals become similar to each other. Because BGA-like algorithms normally optimize only one parameter x_i per offspring, uniform recombination can, at best, combine two different parameters x_i and $x_j, i \neq j$ into a new offspring. Therefore, the number of parameters virtually reduced to $n/2$. But the resulting complexity

$$\begin{aligned} g &= O\left(\frac{n}{2}\ln\frac{n}{2}\right) \\ g &= O\left(n\,\ln n - n\,\ln 2\right) \\ g &= O\left(n\,\ln n\right) \end{aligned} \tag{12}$$

remains unchanged.

6 Discussion

The derivation of the complexity given in the previous section is mainly motivated by the fact that the widely-used test functions are decomposable. Furthermore, certain GA configurations (e.g., $m_p = 1/n$) implicitly use this property and, thus, fail under a simple rotation of the coordinate system. Furthermore, the result is independent of the particular broad mutation operator, the specific elitist selection scheme, or the chosen population size. Recombination merely speeds up convergence by a constant factor, which does not change the resulting complexity.

The presented derivation of the $O(n\,\ln n)$ complexity holds for all decomposable functions as long as the GAs features the described operators (i.e., broad

[3]Schemata are *not short* at the end of the optimization process, because many variables have been already optimized. Thus, schemata are short at the beginning of the optimization process and subsequently, recombination improves the fitness of the individuals in this phase.

mutations and elitist selection) and has the appropriate parameter settings (i.e., $m_p = 1/n$ and uniform recombination). In this sense, the derivation is highly general. On the other hand, the derived complexity does not hold, if the parameters can not be optimized independently. This coincides with the observation that GAs tend to fail if the parameters are not independent. In this sense, its applicability is restricted. Since the described class of GAs can be applied to some *non*-decomposable functions with the same efficiency, Appendix A relaxes the decomposability property a bit.

An interesting consequence of the $O(n \ln n)$ complexity is that it is *not optimal*. Any one-dimensional optimization procedure can optimize a set of independent parameters one after the other. The resulting complexity is only $O(n)$ as opposed to $O(n \ln n)$. This approach is less complex and, therefore, much easier to implement. There would be no need of maintaining a whole population, or using any selection scheme, or any sophisticated mating operator.

Acknowledgements This work was supported in part by a Human Capital and Mobility fellowship of the European Union, grant number ERBCHBICT941266. Thanks to Rolf Pfeifer, Reinhard Riedl, and Thomas Rothenfluh for helpful discussion and editorial assistance.

Appendix A

Since GAs can be applied to non-decomposable functions with the $O(n \ln n)$ complexity, the decomposability assumption can be relaxed in the following way. The crucial requirement is that the parameters x_i can be optimized independently. Let $f^{'}_{x_i}(x_1, \ldots, x_n)$ the first derivative of $f(\vec{x})$ with respect to the parameter x_i and let '$\cdot$' denote an arbitrary value of $x_j, i \neq j$ within its range$_i$. All parameters x_i of a non-decomposable function $f(\vec{x}) \neq \sum_{i=1}^{n} f^i(x_i)$ can be optimized independently, if $\forall\, i, 1 \leq i \leq n$ there exist a solution x_i^o such that $f^{'}_{x_i}(\cdots x_i^o, \cdots) = 0$ and $f(\cdots x_i^o, \cdots)$ represents an optimum in x_i axis.

This might be clarified by the following example. The function $f(x_1, x_2) = x_1^2 + x_2^2$ is decomposable, whereas the function $g(x_1, x_2) = (x_1^2 + x_2^2)^2$ is not. But the parameters x_1 and x_2 of g can still be optimized independently. The first derivatives of g are $\frac{\partial g}{\partial x_1} = 4\,x_1\,(x_1^2 + x_2^2)$ and $\frac{\partial g}{\partial x_2} = 4\,x_2\,(x_1^2 + x_2^2)$ respectively. Clearly, the solutions are $x_1 = 0$ and $x_2 = 0$ and each result is independent of the other variable.

Griewangk's function $f(\vec{x}) = 1 + \sum_{i=1}^{10} x_1^2/4000 - \prod_{i=1}^{10} \cos(x_i\,/\,\sqrt{i})$ with $-600 \leq x_i \leq 600$ is another example of a hard to optimize multimodal function. It has its minimum at $\vec{x}_o = \vec{0}, f(\vec{x}_o) = 0$ and its local minima are approximately located at $x_i(k) \approx \pm k\,\pi\sqrt{i}$. The first derivative with respect to x_i is given by $\frac{\partial f}{\partial x_i} = x_i/2000 + sin(x_i)(1/\sqrt{i})\prod_{j\neq i} cos(x_j/\sqrt{j})$. All solutions $x_i = 0$ can be found independently. However, if a solution $x_i = 0$ is interpreted as a minimum or maximum depends on the initialization of the current value of all other variables $x_j, i \neq j$, i.e., the interpretation depends on whether $\prod_{j\neq i} cos(x_j/\sqrt{j})$ is less or greater than zero. After converver-gence, a simple method that performs a one-dimensional optimization along each axis x_i has to reinitialize all x_i's. But when applied to multimodal functions, a simple method has to do this reinitialization anyway; a GA does this by maintaining a large population. The probability p of finding the minimum $x_i^o = 0$ within r initializations is given by $p = 1 - 0.5^r$. This probability is independent of the number of parameters n and, thus, the complexity of such a simple method is $O(n)$; the resulting complexity of a GA is again $O(n\,\ln n)$.

References

DeJong, K.A. (1975) An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan.

Gill, P.E., Murray, W., Wright, M.H. (1981) *Practical optimization.* Academic Press.

Goldberg, D.E. (1989) *Genetic Algorithms.* Addison-Wesley Publishing Company, INC.

Mühlenbein, H., Schomisch, M., Born, J. (1991) The Parallel Genetic Algorithm as Function Optimizer. *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufman Publisher.

Mühlenbein, H. (1992) How genetic algorithms really work I: Mutation and hill climbing. *Proceedings of Parallel Problem Solving from Nature 2*, 15–26.

Mühlenbein, H., Schlierkamp-Voosen, D. (1993) Predictive Models for the Breeder Genetic Algorithm. *Evolutionary Computation*, MIT Press, 25–50.

Mühlenbein, H., Schlierkamp-Voosen, D. (1994) The Science of Breeding and its Application to the Breeder Genetic Algorithm. *Evolutionary Computation*, MIT Press.

Rechenberg, I. (1973) *Evolutionsstrategie*. Frommann-Holzboog.

Rosenbrock, H.H. (1960) An automatic method for finding the greatest or least value of a function. Computer J. **3**, 175-184.

Salomon, R. (1994) Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions. Submitted to *Evolutionary Computation*.

Schwefel, H.-P. (1977) *Numerische Optimierung von Computer-Modellen mittels Evolutionsstrategie*. Birkhäuser Verlag, Basel und Stuttgart.

Srinivas, M., Patnaik, L. (1994) Genetic Algorithms: A Survey. *Computer*, 17–26.