Theory of Genetic Algorithms
—extended abstract—

Thomas Bäck[*], Jeannette M. de Graaf,
Joost N. Kok and Walter A. Kosters
Leiden University, Department of Computer Science
P.O. Box 9512, 2300 RA Leiden, The Netherlands
Email: joost@cs.leidenuniv.nl

1 Introduction

In this paper we introduce some methods for examining the fundamental properties of genetic algorithms ([Hol75, Jon75, Gol89, Mit96]), probabilistic search and optimization algorithms that work on populations of bit strings, i.e., strings of a fixed length consisting of zeroes and ones.
Genetic algorithms are a subfield of evolutionary computation. For a good overview of evolutionary computation one may consult the Handbook of Evolutionary Computation [BFM97]. In the Handbook also many applications of genetic algorithms can be found.
We do not give a complete overview of the theory of genetic algorithms; in future papers we intend to cover more related topics. We will present results and methods that give an impression of the kind of theory that is present in evolutionary computation today.
We hope that this paper provides a starting point for researchers in theoretical computer science who are interested in the theory of evolutionary computation. This paper is an extended abstract of a paper that appeared in the Bulletin of the European Association for Theoretical Computer Science, Number 63, October 1997, pp. 161–192.

2 Evolutionary Computation

Genetic algorithms form a subclass of evolutionary computation, as is the case for evolution strategies (e.g., [Rec73, Sch77, Rec94, Sch95, Bäc96]), evolutionary programming (e.g., [FOW66, Fog95]), classifier systems (e.g., [HHNT86]) and genetic programming (e.g., [Koz92a, Kin94]). Evolutionary computation is a problem solving method, inspired by the biological principle of evolution. The algorithms based on this evolution principle are generally called evolutionary algorithms. They are mostly used to solve search and optimization problems. For more general overviews of the field the reader is referred to [Fog95, Bäc96, Mic96, BFM97]; see also Section 7.

2.1 Evolutionary Algorithms

Evolutionary algorithms work on populations of individuals, each individual representing a point in search space. An individual may occur more than once in a population, i.e., populations are multisets of individuals. There is a fitness function f (also referred to as objective function) which measures the quality of the individuals in the population. An evolutionary algorithm tries to improve (to a better fitness) a population of individuals by creating new search points, the so-called offspring population. Selection of individuals plays an important role in this process: offspring is created by applying genetic operators—mostly recombination and mutation—to selected multisets of individuals and often only selected offspring is used for the new population. Selection is a two-stage process: first it has to be decided who can compete (in analogy with the natural process of mating selection) and after applying the genetic operators a

[*] Also affiliated with the Center for Applied Systems Analysis, Informatik-Centrum Dortmund, Joseph-von-Fraunhofer-Str. 20, 44227 Dortmund, Germany

selection mechanism (analogous with the natural process of environmental selection) decides who will survive. The general form of an evolutionary algorithm is:

$t := 0$;
initialize(P(0));
evaluate(P(0));
while not done **do**
  $\mathrm{P}'(t)$:= mating_selection($\mathrm{P}(t)$);
  $\mathrm{P}'(t)$:= recombination($\mathrm{P}'(t)$);
  $\mathrm{P}'(t)$:= mutation($\mathrm{P}'(t)$);
  evaluate($\mathrm{P}'(t)$);
  $\mathrm{P}(t+1)$:= environmental_selection($\mathrm{P}(t), \mathrm{P}'(t)$);
  $t := t+1$;
od

In the algorithm, t denotes a generation counter, and evaluate(P) implies an evaluation of the objective function values for all members of population P. The algorithm terminates when for instance the fitness of the current population $\mathrm{P}(t)$ at time t does not improve anymore, or after a fixed number of iterations.

2.2 Selection

The selection process is usually based on the fitness of the individuals. There are three parameters that play an important role: μ is the population size, λ is the number of offspring and κ is the maximal lifetime (or age) of individuals in the population. They are usually fixed during the evolution process.
Next we describe four popular selection mechanisms. First we have examples of mating selection, whereas the last one is the most commonly used environmental selection. An overview of selection methods commonly used in evolutionary algorithms may be found in [DG91, Bäc94b, BT96].

1. Proportional selection ([Hol75]): The probability for an individual to be selected is its fitness divided by the sum of the fitness values of the individuals occurring in the population. This is also called roulette-wheel selection ([Gol89]). For every individual one allocates some "space" on the roulette wheel proportional to the fitness. To select an individual the roulette-wheel is turned and an individual is selected. To select m individuals this process is repeated m times.

2. Tournament selection: Selection of an individual is done by means of a tournament. Uniformly random a number q of individuals is chosen from the population and the one with the highest fitness among those is selected. To select m individuals this process is repeated m times. The parameter of this selection mechanism is the size q of the tournament.

3. Linear ranking ([Bak85]): Individuals in the population are sorted according to their fitness values. Each individual is assigned a rank in the range from one to the number of elements in the population μ. Let $\mathcal{P}(i)$ denote the probability that individual i (in rank) is chosen. The lowest rank corresponds to the lowest fitness. Linear ranking has a parameter α for the probability that the individual with rank one is chosen:
$$\mathcal{P}(1) = \alpha \ .$$
Now each next element in rank gets an offset of β added to the probability of the individual one lower in rank:
$$\mathcal{P}(i+1) = \mathcal{P}(i) + \beta \ .$$
Individuals are chosen relative to their rank instead of their fitness value. The offset β should be chosen such that
$$\sum_i P(i) = 1 \ .$$

4. Deterministic selection ([Sch77, Sch81, BS93, BHS97]): The best μ individuals are selected.

A more detailed analysis of different selection schemes can be found in the work of Bäck [Bäc94a], Deb and Goldberg [DG91] and Blickle and Thiele [BT96].

2.3 Recombination and Mutation

Generally a recombination operator repeatedly takes two individuals (the parents) and produces two or more new individuals (the offspring), which are a combination of their parents. The most popular example is crossover, an operator that exchanges a (usually small) number of random substrings between the parents. For a detailed overview of recombination operators in genetic algorithms, the reader is referred to the Handbook of Evolutionary Computation [BFM97].
The mutation operator toggles bits with a certain small probability. Mutation guarantees that the evolutionary algorithm can reach every point in the search space, and even ensures convergence under suitable conditions.

3 Genetic Algorithms

In genetic algorithms individuals are encoded as bit strings, i.e., strings of a fixed length n consisting of zeroes and ones. A population is a multiset of bit strings of fixed length. A genetic algorithm applies genetic operators to selected individuals in order to improve the multiset, or rather the fitness of its individuals. A standard mating selection mechanism is proportional selection, and standard genetic operators are uniform crossover [Sys89], one-point crossover [Hol75] and mutation [Hol75]. After applying the genetic operators, giving new offspring, the offspring replaces the parent population.
The genetic algorithm has the following form:

$t := 0$;
initialize(P(0));
evaluate(P(0));
while not done **do**
    $\mathrm{P}'(t)$:= proportional_selection($\mathrm{P}(t)$);
    $\mathrm{P}'(t)$:= crossover($\mathrm{P}'(t)$);
    $\mathrm{P}'(t)$:= mutation($\mathrm{P}'(t)$);
    evaluate($\mathrm{P}'(t)$);
    $\mathrm{P}(t+1) := \mathrm{P}'(t)$;   (*)
    $t := t+1$;
od

Many generalizations of the classical genetic algorithm as defined above are used nowadays, and there is not always a clear distinction between these generalized genetic algorithms and other evolutionary algorithms. It is for instance quite natural to keep the best individual in the population in order to ensure an improving course of evolution. This method is often called elitism. When assumptions like elitism must hold one should consider generalized genetic algorithms with a slight adaptation of (*), for example by always keeping the best individual from $\mathrm{P}(t)$ in the new population $\mathrm{P}'(t)$.
In Section 2.2 we described the method of proportional selection. Next we explain how a population changes under the genetic operators uniform crossover, one-point crossover and mutation. Uniform crossover ([Sys89]) takes two bit strings (parents) and with probability p_c (the crossover rate) produces an offspring by taking randomly, with equal probability, as the i-th element one of the two i-th elements from the parents. One-point crossover ([Hol75]) takes two parents and with crossover rate p_c selects a random crossover point $\ell \in \{1, \ldots, n-1\}$ and gives as result a bit string consisting of the ℓ first bits from one parent and the $n-\ell$ last bits from the other parent. In both crossovers, if the operation is not chosen (i.e., with probability $1-p_c$), then the result is (randomly, with equal probability) one of the parents. A genetic algorithm using one-point crossover is called a simple genetic algorithm. Mutation changes each element of a bit string to the opposite value with probability p_m (the mutation rate). This mutation rate is usually a very small value assuring mutation to happen rarely [Hol75, Gol89].

4 The Schema Theorem

The first attempt to theoretically analyze the performance of genetic algorithms led to the well-known Schema Theorem by Holland ([Hol75], [Gol89]). It explains the nature of a genetic algorithm in terms of schemata, and the effect of crossover and mutation on them. We assume here that the genetic algorithm

uses one-point crossover, so we consider simple genetic algorithms. Furthermore the population size is finite (and fixed).
A schema describes a subset of strings (individuals) with similarities at certain positions. To be more precise: a schema is a string of length n containing characters from the alphabet $\{0,1,*\}$, and it represents all strings that match it on every position other than those with a $*$. For example, the schema $*101*0$ represents the four strings 010100, 010110, 110100 and 110110. The order of a schema H, denoted by $o(H)$, is the number of fixed positions, i.e., the number of 0's and 1's present in the schema. For example, $o(*10*1*) = 3$. The defining length of a schema H, denoted by $\delta(H)$, is the distance between the first and the last fixed position in H. For example, $\delta(*10**0) = 6-2 = 4$. The fitness of a schema in a population is defined as the average fitness of all strings in the population that match the schema.
In order to determine the combined effect of the genetic operators proportional selection, one-point crossover and mutation on schemata contained within a population of strings, let $m(H,t)$ be the expected number of strings in the population at time t matching the schema H. Then it is not difficult to show that ([Hol75, Gol89]):

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\overline{f}} \cdot (1 - p_c \cdot \frac{\delta(H)}{n-1}) \cdot (1-p_m)^{o(H)} \ .$$

Here f is the fitness function, $f(H)$ denotes the average fitness of the schema H and $\overline{f}$ is the average fitness of the population. From this equation Holland concluded ([Gol89, Mic96]):

Schema Theorem: Short, low-order, above-average schemata receive an exponentially increasing number of occurrences in subsequent populations.

Of course the exponential increase of schema-occurrences can only be true for a short period, since the population size is finite. Furthermore, from Equation (4) at a certain time t we can not conclude in general an increase of $m(H,t)$ in subsequent generations, since the average fitness $f(H)$ of the schema and $\overline{f}$ of the population change when the population changes.
The short, low-order, above-average schemata, which seem to play an important role in the performance of the genetic algorithm, were named building blocks. The genetic algorithm discovers these building blocks and then repeatedly recombines them in order to produce strings of higher fitness, or in the words of Goldberg [Gol89]:

Building Block Hypothesis: A genetic algorithm seeks near optimal performance through the juxtaposition of building blocks.

There is a lot of discussion on the value of this hypothesis and whether or not it really explains the true nature of a genetic algorithm. Although some research has been done in order to support or even prove this hypothesis, only empirical results are available (see for a recent empirical result e.g., [WLR97]). Several authors have offered their remarks on Holland's hypothesis. See for instance Grefenstette and Baker [GB89] and Bäck [Bäc96], where some remarks on the multi-armed bandit analogy used by Holland are offered. Note that the hypothesis implies that the coding of the problem at hand is very important and should stimulate the formation of building blocks. (See [Rad94, RS95] for a discussion of more general representations and the corresponding generalizations of schema theory.)
A further possible—and sometimes criticized—explanation of the reason why genetic algorithms perform well was given by Holland when he showed that in a population of μ individuals there are at least μ^3 schemata being processed usefully, that is with an exponentially increasing amount of occurrences. Holland called this property implicit parallelism. (See Holland [Hol75], and later Bertoni and Dorigo [BD93].)
Note that the results in this section are more or less independent of the fitness function f. More knowledge about f should give more information and better results. Furthermore one might question the practical usefulness of the results stated, especially because the Schema Theorem focusses on the destructive rather than on the constructive properties of genetic operators. In recent years more analytic results have been given to explain the performance of genetic algorithms and (more general) of evolutionary algorithms, without the use of schemata.

5 Probabilities

Following Liepins and Vose [LV91], we consider infinite populations, i.e., we view a population as a probability distribution and we study the way in which such a distribution changes under the genetic operators. By repeated application of the genetic operators to the distributions, it is possible to trace the distribution from generation to generation, thus simulating genetic algorithms. There is a relationship between the deterministic path of the distributions and models of genetic algorithms with finite population size motivating the tracing of distributions (Nix and Vose [NV92], Vose [Vos92], Whitley [Whi92]). Using special functions (e.g., Walsh products [FK95]) and Fourier analysis ([KKF97]) it is possible to give a detailed analysis of this evolving distribution. Instead of experimenting with just a few populations this theory enables us to follow the full evolution of the distribution.

We use distributions $P(x)$ to express the probability that a string x occurs at a certain time. A distribution associates to each bit string x a probability $P(x)$ such that $\sum_x P(x) = 1$. We are interested in the connection between the distribution just before ($P(x)$) and immediately after ($P'(x)$) a genetic operation. Of course, it is also possible to keep part of the original distribution, using $(1-q)P(x) + qP'(x)$, where q denotes the probability of the genetic operation involved.

We identify bit strings consisting of n bits with subsets of a universe U with n elements in the appropriate way: an element of U is in the "set" x if and only if the corresponding bit equals 1. Using the usual binary representation it is also possible to view bit strings as integers.

First we consider how the distribution changes under the application of the operators. From the definitions of the operators we obtain the following formulae corresponding to the Vose and Liepins model [LV91].

For proportional selection, the probability of a bit string is weighted proportional to its fitness. For example, if the fitness of a bit string is twice as good as the average, proportional selection results in doubling the probability. Selection uses a fitness function f, and satisfies

$$P'(x) = \frac{f(x)}{E[f]} P(x) \ ,$$

where $E[f]$ denotes the expected value of f with respect to the probability distribution P.

For mutation we have

$$P'(x) = \sum_y p^{||\mathrm{xor}(x,y)||}(1-p)^{n-||\mathrm{xor}(x,y)||} P(y) \ ,$$

where $p \in [0,1]$ is the mutation rate, $\mathrm{xor}(l,m) = (l \setminus m) \cup (m \setminus l)$ is the symmetric difference of the sets l and m; here $l \backslash m$ consists of those elements in l that are not in m and $||s||$ denotes the number of elements of the set s. The summation runs over all bit strings (sets) y; in order to generate x all bits in $\mathrm{xor}(x,y)$ should flip, the other bits remaining the same.

Uniform crossover can be expressed as follows. If both parents have a 1 in a certain bit position, the child receives a 1 too (then we are part of the intersection of the two sets); the same holds for a 0. If the two bits from the parents differ, 0 and 1 are equally likely for the child (now we are part of the symmetric difference). Hence we get

$$P'(x) = \sum_{\substack{y,z \\ y \cap z \subseteq x \subseteq y \cup z}} (1/2)^{||\mathrm{xor}(y,z)||} P(y)P(z) \ .$$

For Walsh products $R_i(x) = (-1)^{||i \backslash x||}$, one can show that in this case the expected values of these functions after and before the crossover are related by

$$E'[R_i] = (1/2)^{||i||} \sum_{j \subseteq i} E[R_j]E[R_{i \backslash j}] \ .$$

In [FK95] and [KKF97] this and similar results are basic for a deeper analysis of the genetic algorithm.

For one-point crossover we take the sum over the crossover points of the probabilities for the corresponding prefix and the corresponding postfix:

$$P'(x) = \frac{1}{n-1} \sum_{\ell=1}^{n-1} \left(\sum_{\substack{y \ : \ \mathit{prefix}(y,\ell)= \\ \mathit{prefix}(x,\ell)}} P(y) \ \cdot \sum_{\substack{z \ : \ \mathit{postfix}(z,n-\ell)= \\ \mathit{postfix}(x,n-\ell)}} P(z) \right) \ ,$$

where $\mathit{prefix}(x,\ell)$ means the prefix of length ℓ of bit string x and $\mathit{postfix}(x,\ell)$ denotes the postfix of length ℓ of bit string x.

If we consider the simple genetic algorithm (proportional selection followed by one-point crossover and mutation), then we can combine the definitions into a transition matrix. This matrix has a number of interesting symmetries and it is possible to numerically trace the probabilities (see Liepins and Vose [LV91] and Whitley [Whi92]).

6 Convergence

Suppose we have a genetic algorithm, operating on a finite set of populations. For the sake of convenience, the fitness of a population is defined as the fitness of its best individual. Now we have (cf. [EAH91]):

Convergence theorem: Suppose that the genetic algorithm is elitist, meaning that in every step at least the current best individual survives. Suppose furthermore that for every population P there is a nonzero probability $\mathcal{P}(P)$ that in the next generation the fitness of the population is really better. Then the fitness of the population at time t converges to the optimal value, for $t \to \infty$.

For most genetic algorithms the sequence of populations is a Markov chain. The relation between the theory of Markov chains and evolutionary algorithms is extensively studied in [Rud96], giving deeper insight in the convergence rate.

7 Further Reading

We do not claim that the overview given in this paper is close to complete. Only some of the methods used in evolutionary computation are treated. If one wants to know more about the theory and applications of genetic algorithms or other types of evolutionary algorithms one is referred to the references already mentioned in the text and the suggestions for further reading below.
As good starting points for further studies on the different types of evolutionary algorithms we recommend [Bäc96, BFM97, Dav91, Fog95, Gol89, Kin94, Koz92b, Mic96, Mit96, Sch95]. Several moderated mailing lists (e.g., `GA-List-Request@aic.nrl.navy.mil` and `EP-List-Request@magenta.me.fau.edu`) and newsgroups (such as `comp.ai.genetic`) allow for keeping track of current events and discussions in the field. Useful WWW-sites are:

`http://alife.santafe.edu/~joke/encore/www/`,
`http://www.aic.nrl.navy.mil/galist`,
`http://www.dcs.napier.ac.uk/evonet/evonet.htm` and
`http://www.scs.carleton.ca/csgs/resources/gaal.html`.

There are several relevant conferences in the field of evolutionary computation:

- Parallel Problem Solving from Nature, PPSN [VERS96],
- International Conference on Genetic Algorithms, ICGA [Bäc97],
- IEEE International Conference on Evolutionary Computation, ICEC [IEE97],
- Annual Conference on Evolutionary Programming, EP [ARME97],
- Annual Conference on Genetic Programming, GP [KDD+97].

References

[ARME97] P. J. Angeline, R. G. Reynolds, J. R. McDonnell, and R. Eberhart, editors. *Proceedings of the Sixth International Conference on Evolutionary Programming, EP97*, volume 1213 of *Lecture Notes in Computer Science*. Springer, Berlin, 1997.

[Bäc94a] Th. Bäck. Evolutionary algorithms: Comparison of approaches. In R. C. Paton, editor, *Computing with Biological Metaphors*, chapter 14, pages 227–243. Chapman & Hall, London, 1994.

[Bäc94b] Th. Bäck. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 57–62. IEEE Press, Piscataway, NJ, 1994.

[Bäc96] Th. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[Bäc97] Th. Bäck, editor. *Genetic Algorithms: Proceedings of the 7th International Conference*. Morgan Kaufmann Publishers, San Francisco, CA, 1997.

[Bak85] J. E. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 101–111. Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[BD93] A. Bertoni and M. Dorigo. Implicit parallellism in genetic algorithms. *Artificial Intelligence*, 61(2):307–314, 1993.

[BFM97] Th. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[BHS97] Th. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: History and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[BS93] Th. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[BT96] T. Blickle and L. Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.

[Dav91] L. Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[DG91] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. IlliGAL Report 91009, University of Illinois at Urbana-Champaign, IL, December 1991.

[EAH91] A. E. Eiben, E. H. L. Aarts, and K. M. Van Hee. Global convergence of genetic algorithms: An infinite Markov chain analysis. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature — Proceedings 1st Workshop PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 4–12. Springer, Berlin, 1991.

[FK95] P. Floréen and J.N. Kok. Tracing the behavior of genetic algorithms using expected values of bit and Walsh products. In L.J. Eshelman, editor, *Proceedings of the sixth international conference on genetic algorithms*, pages 201–208. Morgan Kaufmann Publishers, Inc., 1995.

[Fog95] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 1995.

[FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, New York, 1966.

[GB89] J. J. Grefenstette and J. E. Baker. How genetic algorithms work: A critical look at implicit parallelism. In Schaffer [Sch89], pages 20–27.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[HHNT86] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction: Processes of Inference, Learning, and Discovery*. The MIT Press, Cambridge, MA, 1986.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

[IEE97] IEEE. *Proceedings of the Fourth IEEE Conference on Evolutionary Computation, Indianapolis, IN*. IEEE Press, Piscataway, NJ, 1997.

[Jon75] K. A. De Jong. *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975. Diss. Abstr. Int. 36(10), 5140B, University Microfilms No. 76–9381.

[KDD+97] J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors. *Genetic Programming 1997. Proceedings of the Second Annual Conference*. Morgan Kaufmann, San Francisco, CA, 1997.

[Kin94] K. E. Kinnear, editor. *Advances in Genetic Programming*. The MIT Press, Cambridge, MA, 1994.

[KKF97] W.A. Kosters, J.N. Kok, and P. Floréen. Fourier analysis of genetic algorithms. Preprint, Leiden University, 1997. To appear in Theoretical Computer Science.

[Koz92a] J. R. Koza. *Genetic Programming.* MIT Press, 1992.

[Koz92b] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Complex Adaptive Systems. The MIT Press, Cambridge, MA, 1992.

[LV91] G. E. Liepins and M. D. Vose. Polynomials, basis sets, and deceptiveness in genetic algorithms. *Complex Systems*, 5:45–61, 1991.

[Mic96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin, 1996.

[Mit96] M. Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, MA, 1996.

[NV92] A. Nix and M. D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.

[Rad94] N. J. Radcliffe. The algebra of genetic algorithms. *Annals of Maths and Artificial Intelligence*, 10:339–384, 1994.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann–Holzboog, Stuttgart, 1973.

[Rec94] I. Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik.* Frommann–Holzboog, Stuttgart, 1994.

[RS95] N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In M. D. Vose L. D. Whitley, editor, *Foundations of Genetic Algorithms 3*, pages 51–72. Morgan Kaufmann Publishers, San Francisco, CA, 1995.

[Rud96] G. Rudolph. *Convergence Properties of Evolutionary Algorithms.* PhD Thesis, University of Dortmund, 1996.

[Sch77] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research.* Birkhäuser, Basel, 1977.

[Sch81] H.-P. Schwefel. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[Sch89] J. D. Schaffer, editor. *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[Sys89] G. Syswerda. Uniform crossover in genetic algorithms. In Schaffer [Sch89], pages 2–9.

[VERS96] H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors. *Parallel Problem Solving from Nature IV. Proceedings of the International Conference on Evolutionary Computation*, volume 1141 of *Lecture Notes in Computer Science.* Springer, Berlin, 1996.

[Vos92] M. D. Vose. Modeling simple genetic algorithms. In Whitley [Whi93], pages 63–73.

[Whi92] L. D. Whitley. An executable model of a simple genetic algorithm. In *Foundations of Genetic Algorithms 2* [Whi93], pages 45–62.

[Whi93] L. D. Whitley, editor. *Foundations of Genetic Algorithms 2.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[WLR97] A.S. Wu, R.K. Lindsay, and R.L. Riolo. Empirical observations on the roles of crossover and mutation. In Bäck [Bäc97], pages 362–369.

This paper has appeared in the 1998 newspaper of the Dutch Organization for Theoretical Computer Science (NVTI), pp. 27–35.