Optimization by Means of Genetic Algorithms

Thomas Bäck
baeck@lumpi.informatik.uni-dortmund.de

University of Dortmund · Department of Computer Science · Chair XI
P.O. Box 50 05 00 · D–4600 Dortmund 50 · Germany

Abstract

Genetic Algorithms (GAs) are powerful heuristic search strategies based upon a simple model of organic evolution. The basic working scheme of GAs as developed by Holland [Hol75] is described within this paper in a formal way, and extensions based upon the *second-level learning* principle for strategy parameters as introduced in *Evolution Strategies* (ESs) are proposed. First experimental results concerning this extension of GAs are also reported.

1 Introduction

Natural evolution has proven to be a powerful mechanism for emergence and improvement of the living beings on our planet by performing a randomized search in the space of possible DNA-sequences. According to our current knowledge about the features of evolution, some researchers tried to use the mechanisms of evolution as a basis of optimum-seeking techniques in case of vast search spaces.

In general, the algorithmic models of Darwinian evolution maintain a *population* of *individuals* (biological terminology has been adopted in this field). The population is able to adapt to a given (static or dynamically changing) environment by randomized processes of *selection*, *reproduction*, sexual *recombination*, and *mutation*. The environment provides a quality information (*fitness*) for the individuals, and the selection process favours the individuals of higher quality to survive ("survival of the fittest"). Even the reproduction process often favours structures of higher quality. Thus, during the evolution process the average quality of the population increases, hopefully leading to an optimum solution.

Algorithms following this general approach have been summarized under the term *Evolutionary Algorithms* elsewhere [BH91b]. Some important representatives are the *Genetic Algorithm* (GA) [Hol75] and *Classifier System* (CS) [Hol86] by Holland, the *Evolution Strategy* (ES) [Rec73, Sch81] by Rechenberg and Schwefel, and the *Genetic Programming Paradigm* [Koz89]. The approaches are mainly differing with respect to the structure of the individuals, which directly influences the recombination and mutation operators. ESs benefit from the additional capability of learning on the level of strategy parameters by *self-adapting* them during the search (*second-level learning*) [Sch81] — a feature that has not yet been introduced into the general concept of other Evolutionary Algorithms.

2 The traditional Genetic Algorithm

A general definition of the GA does not exist. As an abstraction from the work of Holland [Hol75] and the nowadays often used quasi-standard given by a computer-implementation of John Grefenstette [Gre87], the following 8-tuple formalization is used here:

$$\mathrm{GA} = (P^0, \lambda, l, s, \rho, \Omega, f, t) \tag{1}$$

where

P^0	$=$	$(a_1^0, \ldots, a_\lambda^0) \in I^\lambda$	$I = \{0,1\}^l$	initial population
λ	$\in$	$\mathbf{N}$		population size
l	$\in$	$\mathbf{N}$		length of individuals' representation
s	$:$	$I^\lambda \to I^\lambda$		selection operator
ρ	$:$	$I \to \Omega$		operator determination function
Ω	$\subseteq$	$\{\omega : I \times I^\lambda \to \mathcal{P} \to I\}$[1]		genetic operator set
f	$:$	$I \to I\!R$		fitness function
t	$:$	$I^\lambda \to \{0,1\}$		termination criterion

GAs are working on binary encoded individuals, and the bits are often referred to as *alleles*. In this raw form the algorithm requires an additional mapping from the *genotype-level* of bitstrings to a more complicated problem representation (e.g. permutations or real-valued object variables). The choice of using binary alleles is theoretically founded by the argument of maximizing the number of *schemata* (hyperplanes of varying dimension in the l-dimensional binary space) available with respect to a given code [Hol75, Gol89].

P^0 is the randomly generated initial population, and the parameters λ and l describe the number of individuals representing one generation and the length of the 'genetic' representation of each individual, respectively.

The selection operator s produces an intermediate population P'^t from the population P^t by the generation of copies of elements from P^t: $P'^t = s(P^t)$. This is done by taking λ subsequent samples from $P^t = (a_1^t, \ldots, a_\lambda^t)$ according to the probability distribution $\mathcal{P} \ni p_s : I \to [0,1]$, where $p_s(a_i^t) = f(a_i^t) \;/\; \sum_{j=1}^{\lambda} f(a_j^t)$ denotes the probability of individual a_i^t to be sampled. This selection scheme is called *proportional selection* [Hol75]. It leads to the expectation of individual a_i^t to occur $\eta_i^t = \lambda p_s(a_i^t)$ times in generation $t+1$ (generally η_i^t is called the *expected value* of a_i^t).

After the selection phase has taken place, genetic operators are to be applied. One may think of a mapping $\rho : I \to \Omega$ which determines an operator $\omega_i^t \in \Omega$ for each individual $a_i'^t \in P'^t$ which will be applied to this individual: $\rho(a_i'^t) = \omega_i^t \;\forall i \in \{1, \ldots, \lambda\}$.

The genetic operator set $\Omega \subseteq \{\omega : I \times I^\lambda \to \mathcal{P} \to I\}$ includes genetic operators like *crossover* and *mutation* [Hol75]. The stochastic elements of these operators (application probabilities, e.g. $p_m \approx 0.001$ for mutation and $p_c \approx 0.6$ for crossover, choice of loci) are included in a somehow abstract manner in the probability distribution $p \in \mathcal{P}$. According to an individual $a_i'^t \in P'^t$ the operator selects mating partners out of P^t if needed (e.g. for

[1] $\mathcal{P} = \{p : I \to [0,1]\}$ denotes the set of probability distributions on I.

crossover) and determines the new individual. Operators leading to more than one offspring (e.g. crossover) at first select one of the offspring at random to be the result, while the other offspring is discarded.

Implementations often allow more than one operator to be applied sequentially to one individual, e.g. mutation is often applied afterwards to the offspring created by crossover. This might be included in the scheme described above by allowing the function obtained from ρ to be composites of operators, i.e. $\omega_i^t = \omega_{i_1}^t \circ \omega_{i_2}^t \circ \ldots \circ \omega_{i_m}^t$, e.g.: $\Omega = \{\omega_m \circ \omega_{c_1}, \omega_m\}$. Now the transition from generation t to generation $t+1$ may be described as follows:

$$\begin{array}{rcll} s(P^t) & = & P'^t = (a_1'^t, \ldots, a_\lambda'^t) & \\ \rho(a_i'^t) & = & \omega_i^t & \forall i \in \{1, \ldots, \lambda\} \\ a_i^{t+1} & = & \omega_i^t(a_i'^t, P^t) & \forall i \in \{1, \ldots, \lambda\} \\ P^{t+1} & = & (a_1^{t+1}, \ldots, a_\lambda^{t+1}) & \end{array} \tag{2}$$

Although the λ-tuple notation used here to describe a population seems to imply a deterministic order of individuals, the stochastic effect caused by the selection mechanism leads to a new shuffle of the population during each selection step.

The fitness values are obtained by the *fitness function* $f : I \to I\!R$, which is usually seen as a black box containing components like decoding of individuals, constraint handling, *scaling*, and the objective function itself. Note, that with respect to proportional selection fitness values must always be positive, which led to the development of scaling techniques. $t : I^\lambda \to \{0, 1\}$ denotes a termination criterion.

Recently it has been proven that under certain assumptions the algorithm described so far finds a global optimum point $a^* \in I$ ($f^* := f(a^*) = \max\{f(a) \mid a \in I\}$) with probability 1, i.e. [Har90]:

$$\lim_{t \to \infty} \mathrm{P}[a^* \in P^t] = 1$$

where $\mathrm{P}[\cdot]$ denotes a probability. Necessary requirements are:

(1) The sequence of populations $P^0, P^1, \ldots$ is monotone, i.e.

$$\forall t \in I\!N : \quad \max\{f(a) \mid a \in P^{t+1}\} \geq \max\{f(a) \mid a \in P^t\}$$

(in case of a maximization problem). To guarantee this, an explicit *elitist* selection variant must be used, which always maintains the currently best individual to be a member of the next generation. Then, a population containing a^* is an absorbing state of an appropriately defined markov chain.

(2) For all $a, a' \in I$ individual a' must be *reachable* from a by means of mutation and recombination, where reachability is defined by a sequence of transitions in successive neighbourhood structures (this implies a mutation probability $p_m > 0$)[2].

However, these results are — like similar ones for Simulated Annealing [AK89] and ESs [Bor78] — of no practical worth. Due to this fact, a lot of additional heuristics have been introduced and tested for GA-applications. Some of the main extensions are presented in the next section.

[2]The detailed proof as well as the exact definitions are omitted here, but can be found in [Har90].

3 Extensions of the traditional Genetic Algorithm

Critical components of the GA are the selection operator, the recombination operator, and the mutation and recombination probabilities. Especially selection controls the character of the search and has to balance between the contradicting aims of finding the global optimum (searching volume-oriented, exploration) and converging quickly (searching path-oriented, exploitation).

Especially proportional selection can lead to a situation characterized by one individual with a very high fitness compared to the rest of the population. In the following generations this *superindividual* may completely dominate the population, thus leading to *premature convergence* towards a suboptimal solution. Two attempts to solve these problems are described here in short:

(1) Use *static* distributions for selection probabilities instead of dynamically changing, fitness-dependent ones. The first approach by Baker [Bak85] is called *linear ranking* and assumes the population to be sorted by fitness values, the most fit individual being a_1. Then, selection probabilities are constant values, given by the expression

$$p_s(a_i) = \frac{1}{\lambda}\left(\eta_{max} - (\eta_{max} - \eta_{min})\frac{i-1}{\lambda-1}\right)$$

where $\eta_{min} = 2 - \eta_{max}$ and $1 \leq \eta_{max} \leq 2$. Baker suggests a *maximum expected value* of $\eta_{max} = 1.1$.

(2) *Scaling* of fitness values transforms the fitness values $f(a_i)$ by an additional mapping $f' : \mathbb{R} \times I^{\lambda} \rightarrow \mathbb{R}$ to a scaled fitness $f'(f(a_i), P^t)$ before applying proportional selection. The aim of these techniques is to stretch the range of fitness values if the population has almost converged, and to penalize superindividuals. A lot of scaling techniques has been suggested [For85, GB89, Gol89], but only very few of them have been used practically.

Experimental results give strong indications for the advantage of ranking in case of multimodal problems [Bak85, Whi89], although these techniques have been neglected by the GA-community for a long time.

Furthermore, the mutation probability is a critical parameter of the algorithm, and any efforts to find an optimal probability by theoretical or experimental investigations did not lead to universally valid results [SCED89, HM91]. The problem is to balance between destroying good information and effectively using mutation as an additional search operator. Concerning mutation rates a promising approach is that of self-learning of strategy parameters as introduced in ESs. A first transfer of this technique to GAs works as follows:

(1) The genotype of the individuals is extended to include the encoding of 1 or n (n being the number of object variables the individual encodes) mutation rates.

(2) Mutation of an individual proceeds by the following steps:

- The mutation rates of the individual are decoded (i.e. the bitstrings are mapped to probabilities $p_{m_i} \in [0,1]$).
- For each mutation rate p_{m_i} the mutation rate itself is mutated with a bit-mutation probability p_{m_i}, yielding mutation probability p'_{m_i}. The associated object variable part is mutated with bit mutation probability p'_{m_i}.

Preliminary research results indicate that selection should be static and, in addition, be *extinctive* in order to enable higher mutation rates and their self-adaptation. Here extinctive means, that only $\mu < \lambda$ of the λ individuals are allowed to be selected — the rest of the population is discarded. Borrowed from ESs, the notion (μ,λ)-linear ranking or (μ,λ)-proportional selection is used to describe such a selection scheme [BH91a].

4 Experimental Results

To illustrate the self-learning capability of GAs concerning mutation rates a simple unimodal and a multimodal test function as described in table 1 are used.

Name	Description	Dim.	Characteristics	Ref.
f_1	sphere model $f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$ $-5.12 \le x_i \le 5.12$	$n = 30$	unimodal, high–dimensional	[HB91] [Jon75] [Sch81]
f_7	generalized Rastrigin's function $f_7(\vec{x}) = nA + \sum_{i=1}^{n} x_i^2 - A\cos(\omega x_i)$ $A = 10$; $\omega = 2\pi$; $-5.12 \le x_i \le 5.12$	$n = 20$	multimodal, high–dimensional, f_1 with sine wave superposition	[HB91] [TZ89]

Table 1: The set of test functions

To obtain the results, a modified version of Grefenstette's GENESIS–GA [Gre87] was used. The GA is defined by the following parameter and configuration settings: Crossover rate $p_c = 0.6$; population size $\lambda = 50$; length of an individual $l = 32n$, where n denotes the dimension of the objective function[3]; two–point crossover; Gray code. For ranking the usual setting of $\eta_{max} = 1.1$ was chosen according to [Bak85]. A GA with normal ranking was compared to (10,50)-linear ranking with 1 adaptive mutation rate for f_1 and 20 adaptive mutation rates for f_7 as well as the standard GA using proportional selection and the normal mutation rate $p_m = 0.001$. The best values in each generation, averaged over 10 runs, are presented in figure 1 (objective function values are minimized here).

In both cases, self-learning leads to much better solutions than the traditional GA and has the additional advantage of eliminating a part of the parameterization problem for GAs.

[3]A length of 32 bits per object variable is used for the representation of the real interval $[x_{\min}, x_{\max}]$ to which the bitstrings are mapped, in order to achieve a maximum resolution $\Delta x = (x_{\max} - x_{\min})/(2^{32} - 1)$.

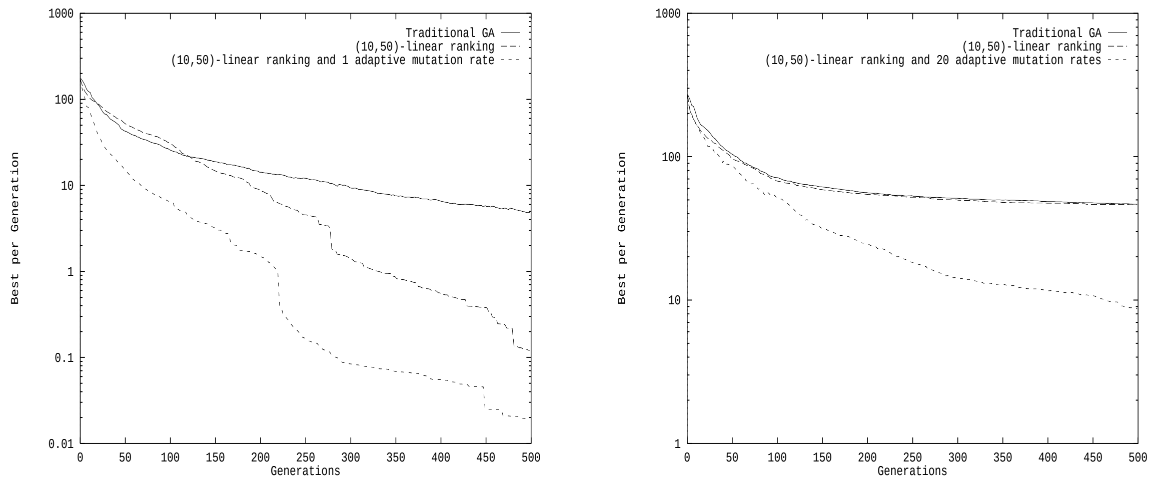

Figure 1: Different GA-runs on f_1 (left) and f_7 (right)

5 Conclusions

Although global convergence is not guaranteed within finite time, GAs have proven to be robust heuristic search techniques in many real-world applications [Gol89]. However, a lot of heuristic extensions were proposed and tested in the past, without leading to a clear statement how the GA in general should be parameterized and configurated.

Self-adaptation with respect to the actual topology of the fitness function could be a promising way to solve these problems. Research from ESs confirms that under certain conditions (possibility to forget good information, recombination, extinctive selection) self-adaptation works well and enlarges effectivity *and* efficiency of an Evolutionary Algorithm [Sch88].

References

[AK89] Emile H. L. Aarts and Jan Korst. *Simulated Annealing and Boltzmann Machines.* Wiley, Chichester, 1989.

[Bak85] James Edward 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, Hillsdale, New Jersey, 1985. Lawrence Erlbaum Associates.

[BH91a] Thomas Bäck and Frank Hoffmeister. Extended selection mechanisms in genetic algorithms. In Richard K. Belew, editor, *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*, San Diego, California, USA, 1991. Morgan Kaufmann Publishers.

[BH91b] Thomas Bäck and Frank Hoffmeister. Global optimization by means of evolutionary algorithms. In Alexander N. Antamoshkin, editor, *Random Search as a Method for Adaptation and Optimization of Complex Systems*, pages 17–21, Divnogorsk, UdSSR, March 1991. Krasnojarsk Space Technology University.

[Bor78] Joachim Born. *Evolutionsstrategien zur numerischen Lösung von Adaptationsaufgaben.* Dissertation A, Humboldt-Universität, Berlin, GDR, 1978.

[For85] Stephanie Forrest. *Documentation for Prisoner's Dilemma and Norms Programs that use the Genetic Algorithm.* The University of New Mexico, Albuquerque, NM, 1985.

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

[Gol89] David E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.

[Gre87] John J. Grefenstette. *A User's Guide to GENESIS.* Navy Center for Applied Research in Artificial Intelligence, Washington, D. C., 1987.

[Har90] Richard F. Hartl. A global convergence proof for a class of genetic algorithms. University of Technology, Vienna, 1990.

[HB91] Frank Hoffmeister and Thomas Bäck. Genetic algorithms and evolution strategies: Similarities and differences. In Schwefel and Männer [SM91], pages 455–470.

[HM91] J. Hesser and R. Männer. Towards an optimal mutation probability in genetic algorithms. In Schwefel and Männer [SM91], pages 23–32.

[Hol75] John H. Holland. *Adaptation in natural and artificial systems.* The University of Michigan Press, Ann Arbor, 1975.

[Hol86] John H. Holland. Escaping brittleness: The possibilities of general–purpose learning algorithms applied to parallel rule-based systems. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning Vol.II*, chapter 20, pages 593–623. Morgan Kaufmann Publishers, Inc., 1986.

[Jon75] Kenneth 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.

[Koz89] John R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. In N. S. Sridharan, editor, *Eleventh international joint conference on artificial intelligence*, pages 768–774. Morgan Kaufmann Publishers, August 1989.

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

[SCED89] J. David Schaffer, Richard A. Caruna, Larry J. Eshelman, and Rajarshi Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer [Sch89], pages 51–60.

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

[Sch88] Hans-Paul Schwefel. Evolutionary learning optimum–seeking on parallel computer architectures. In A. Sydow, S.G. Tzafestas, and R. Vichnevetsky, editors, *Proceedings of the International Symposium on Systems Analysis and Simulation 1988, I: Theory and Foundations*, pages 217–225, Berlin, Sept. 1988. Akademie der Wissenschaften der DDR, Akademie–Verlag.

[Sch89] J. David Schaffer, editor. *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, San Mateo, California, June 1989. Morgan Kaufmann Publishers.

[SM91] Hans-Paul Schwefel and Reinhard Männer, editors. *Parallel Problem Solving from Nature.* Lecture Notes in Computer Science 496. Springer, Berlin, 1991.

[TZ89] A. Törn and A. Zilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science.* Springer, Berlin, FRG, 1989.

[Whi89] Darrell Whitley. The GENITOR algorithm and selection pressure: Why rank–based allocation of reproductive trials is best. In Schaffer [Sch89], pages 116–121.