# EVOLUTIONARY ALGORITHMS FOR FUZZY LOGIC: A BRIEF OVERVIEW

Thomas BÄCK, Frank KURSAWE

University of Dortmund, Department of Computer Science, LS XI
D–44221 Dortmund, Germany
{baeck,kursawe}@LS11.informatik.uni-dortmund.de

**ABSTRACT**

*Evolutionary algorithms* are direct, global optimization algorithms gleaned from the model of organic evolution. The most important representatives, *genetic algorithms* and *evolution strategies*, are briefly introduced and compared in this paper, and their major differences are clarified. Furthermore, the paper summarizes the application possibilities of evolutionary algorithms in the design of *fuzzy logic controllers*. The optimization of fuzzy membership functions turns out to be a promising and successful application domain for evolutionary algorithms, while the automatic learning of fuzzy control rules by means of *fuzzy classifier systems* is still in an early stage of research.

## 1 Evolutionary Algorithms

Evolutionary algorithms are a class of direct, probabilistic search and optimization methods based on the model of organic evolution (where they also borrow most of the terminology from). The algorithms exploit the collective learning process within a *population* of *individuals*, and each of the individuals represents a search point in the space of potential solutions to a given problem. The start population (which is often randomly initialized) evolves towards increasingly better regions of the search space by means of randomized processes of *selection*, *mutation*, and *recombination*. The selection operator favours individuals of higher fitness (quality in terms of the objective function $f : M \rightarrow I\!R$ which characterizes the optimization problem) to reproduce more often than individuals of lower fitness. Recombination allows for the exchange of information (partial solutions) between individuals, and mutation introduces innovation into the population.

Using this high level of abstraction, we can formulate a general basic algorithm which subsumes the existing evolutionary algorithms. In the following, $t$ denotes a *generation* (iteration) counter and $P(t) \in I^{\mu}$ is a population of $\mu$ individuals at generation $t$. $I$ denotes the space of individuals and is not necessarily identical to the optimization problem's search space $M$ because individuals may carry additional information. $P'(t) \in I^{\lambda}$ and $P''(t) \in I^{\lambda}$ are used to indicate intermediate populations of size $\lambda$ ($\mu = \lambda$ is possible). Furthermore, we use $Q \in \{P(t), \emptyset\}$ to denote a set of individuals which might be taken into account by selection in addition to the intermediate population $P''(t)$. The resulting evolutionary algorithm consists of a simple loop of recombination, mutation, fitness evaluation, and selection which is iterated until a specific termination criterion is fulfilled:

ALGORITHM 1 (Basic evolutionary algorithm)

$t := 0$;
*initialize* $P(t) \in I^{\mu}$;
*evaluate* $P(t)$;
**while not** *terminate*$(P(t))$ **do**
  *recombine*: $P'(t) := r(P(t))$;
  *mutate*: $P''(t) := m(P'(t))$;
  *evaluate* $P''(t) \in I^{\lambda}$;
  *select*: $P(t+1) := s(P''(t) \cup Q)$;
  $t := t + 1$;
**od**
**return**(*best individual in* $P(t)$);

The currently most important and widely known concrete representatives of this general outline are *genetic algorithms* (GAs) [10, 12, 8], *evolution strategies* (ESs) [21, 22, 23], and *evolutionary programming* (EP) [7, 5, 6]. In the following, we restrict the presentation to a short overview of GAs and ESs (modern variants of EP are quite similar to ESs) to clarify the most important characteristics and differences of both approaches. The interested reader is referred to the original literature or the more detailed overview given in [2].

## 1.1 Genetic Algorithms

In traditional genetic algorithms, individuals are always represented by binary vectors $\vec{a} \in I\!B^l$ ($I\!B = \{0,1\}$) of fixed length $l$. The reasons for this choice are both of historical and theoretical nature (the theory of genetic algorithms is based on the analysis of so-called *schemata* — similarity templates representing hyperspaces of $I\!B^l$ — and schema processing properties of the algorithm; see [8] for more details). In case of optimization problems which are not formalizable as functions $f : I\!B^l \to I\!R$, a coding mechanism is utilized to represent the search space of the optimization problem by binary vectors (finding such a code is often a difficult task, such that some recent applications of "genetic algorithms" are based on a combination of direct representations of candidate solutions and problem-specific genetic operators; see e.g. [18]).

Genetic algorithms put a strong emphasis on the recombination (*crossover*) operator as the main search operator. In its simplest form, crossover exchanges all bits to the right of a randomly chosen position between two individuals [10]. This one-point crossover can naturally be extended by sampling more than one breakpoint and alternately exchanging each second of the resulting segments [12]. In the extreme case of *uniform crossover*, a random decision whether to exchange it or not is made for each bit position of the individuals [26]. Besides the number of crossover points, the operator is characterized by the *crossover probability* $p_c$ which denotes the probability per individual to undergo recombination (often, $p_c \approx 0.6$ is chosen)

The role of mutation is normally interpreted to be only of marginal importance in GAs (a "background" operator [10]). It works by occasionally inverting single bits of individuals with an extremely small probability $p_m$ (e.g., $p_m \approx 0.001$ [12]). Recent investigations, however, clarify that the importance of mutation was so far underestimated and a more recommendable setting is given by $p_m = 1/l$ [19, 1].

Selection in genetic algorithms is a probabilistic operator which works by copying individuals from $P''(t)$ ($Q = \emptyset$) into the new parent population $P(t+1)$. Each individuals' selection

probability (the probability to be copied) is given by the proportion of its fitness from the total population fitness (*proportional selection*):

$$p(\vec{a}_i) \quad = \quad \frac{f(\vec{a}_i)}{\sum_{j=1}^{\mu} f(\vec{a}_j)} \; .$$

Notice that this definition assumes positive fitness values and a maximization task; otherwise, so-called *scaling mechanisms* have to be used in combination with proportional selection [8].

Genetic algorithms always maintain a constant population size (i.e., $\mu = \lambda$) which is of the order of 50–100 individuals. Normally, the start population is randomly initialized (with probability 0.5 for a one respectively a zero bit) and the algorithm is terminated after a predefined number of generations has passed.

### 1.2 Evolution Strategies

Initially developed for experimental optimization purposes [21], evolution strategies are nowadays important computer algorithms for continuous parameter optimization problems $f : I\!R^n \rightarrow I\!R$ [22, 23]. In contrast to genetic algorithms, candidate solutions are directly represented by real-valued vectors $\vec{x} \in I\!R^n$, and individuals $\vec{a} = (\vec{x}, \vec{\sigma})$ consist not only of the vector $\vec{x}$, but also incorporate an additional, $n$-dimensional vector $\vec{\sigma} \in I\!R^n_+$ of positive standard deviations $\sigma_i$. These *strategy parameters* $\sigma_i$ are utilized by the mutation operator to modify the corresponding object variables $x_i$ ($i \in \{1, \ldots, n\}$).

Mutation works for each of the object variables $x_i$ by adding normally distributed random numbers with expectation zero and variance $\sigma_i^2$ (indicated by the notation $N(0, \sigma_i^2)$). The standard deviations $\sigma_i$ are neither constant nor explicitly controlled, but they also undergo a logarithmic-normally distributed variation mechanism:

$$\begin{aligned} \sigma_i' &= \sigma_i \cdot \exp\left(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)\right) \; , \\ x_i' &= x_i + \sigma_i' \cdot N_i(0,1) \; . \end{aligned}$$

The mutation of $\sigma_i$ is based on a *global* factor $\tau' \cdot N(0,1)$ (the random number is sampled only once for the complete individual) and a *local* factor $\tau \cdot N_i(0,1)$ (the random number is sampled anew for each component). Schwefel recommends the settings $\tau' \sim 1/\sqrt{2n}$ and $\tau \sim 1/\sqrt{2\sqrt{n}}$ for the "learning rates" $\tau'$ and $\tau$ [22].

It is important to see that selection works implicitly on the strategy parameters $\vec{\sigma}$ by exploiting the link between advantageous changes of object variables (i.e., a large improvement of fitness) and useful standard deviations (i.e., appropriate *internal models* of the objective function topology). This mechanism of *self-adaptation* of strategy parameters allows for an adaptation of these parameters without the need for finding an appropriate exogenous control mechanism [24, 9]. Besides the standard deviations up to $n \cdot (n-1)/2$ covariances of the generalized $n$-dimensional normal distribution may also be taken into account for self-adaptation, which introduces linear *correlated mutations* to the algorithm. This mechanism is able to accelerate the search in case of a complicated local topology [23].

While mutation is the most important search operator in evolution strategies, recombination on strategy parameters and object variables is necessary for the self-adaptation process

and often helpful for the progress of the search. Normally, an *intermediate recombination* operator is recommended for strategy parameters, i.e. a strategy parameter of an offspring individual results from taking the average of the corresponding strategy parameters of both parents. In case of the object variables, a *discrete recombination* operator where each $x_i$ is randomly copied from either the first or the second parent (in analogy with uniform crossover in genetic algorithms) is normally preferred. The most appropriate choice, however, clearly depends on the particular objective function topology.

A further, secondary function of recombination in evolution strategies consists in changing the population size from $\mu$ to $\lambda$ individuals (a setting of $\mu = 15$, $\lambda = 100$) is quite normal). This works by repeating the application of recombination on the level of individuals $\lambda$ times (in contrast to genetic algorithms, recombination in evolution strategies is always applied, i.e., no parameter comparable with the crossover rate exists in ESs).

Finally, the selection operator in evolution strategies is completely deterministic and works by choosing the $\mu$ best individuals out of $P''(t)$ ($Q = \emptyset$, ($\mu$,$\lambda$)-selection) respectively out of $P''(t) \cup P(t)$ ($Q = P(t)$, ($\mu$+$\lambda$)-selection) to become parents of the next generation. The ($\mu$,$\lambda$)-selection is preferred in modern implementations of the ES, because it supports the self-adaptation mechanism (by providing the possibility to become extinct for inappropriate strategy parameter settings) and allows for an application of evolution strategies in case of time-varying or perturbed objective functions.

Since self-adaptation of strategy parameters is certainly the most distinguishing (and complicated) feature of ESs, we summarize the criteria which were identified by Schwefel to be critical for a successful self-adaptation mechanism [25]: A ($\mu$,$\lambda$)-strategy should be used, with a not too small value of $\mu$ (i.e., the selective pressure should not be too strong), e.g., a (15,100)-strategy, and the recombination operator should be applied also on strategy parameters (especially intermediate recombination).

### 1.3 A Summary of Differences

Although, at first glance, the representation of individuals seems to be the distinguishing property of genetic algorithms and evolution strategies, the self-adaptation concept of strategy parameters — which is completely missing in genetic algorithms — is of much more importance. The process of tuning strategy parameters "by hand" for a particular application problem, which is often a time-consuming problem in applying a genetic algorithm, is not required for evolution strategies. Concerning the genetic operators, the roles of mutation in evolution strategies respectively recombination in genetic algorithms reflect an emphasis on different operators which might be explained by a phenotypical level of modeling in case of ESs and a genotypical level in case of GAs. Finally, the selection operators are characterized by deterministic, rank-based survival of the top group in evolution strategies versus a probabilistic mechanism with nonzero chances of survival even for the worst individuals in case of genetic algorithms. Currently, much of the theoretical basis to understand the pros and cons of these different principles of representations and operators is still missing and subject to active research.

## 2 Evolutionary Algorithms for Fuzzy Logic

The application possibilities for evolutionary algorithms in the field of fuzzy logic are documented by a number of recent research publications, which can roughly be divide into two groups:

- Optimization of the membership functions of fuzzy sets.
- Automatic learning of fuzzy rules.

In the following, we will briefly discuss the general idea for both topics and refer the interested reader to the literature cited for more details on the applications and fuzzy logic in general (see [15, 16]).

## 2.1 Optimization of Membership Functions

Fuzzy membership functions provide the characterization of fuzzy sets by establishing a connection between linguistic terms (such as "slow", "medium", "fast" for a speed variable) and precise numerical values of variables in a physical system. A fuzzy membership function approximates the confidence with which a numerical value is described by a linguistic term. A typical example of triangular fuzzy membership functions for a speed variable is given in figure 1.

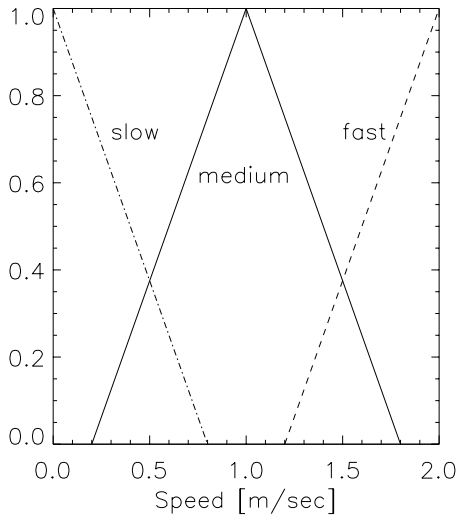


Figure 1: Membership functions for a physical variable "speed".

Notice that membership functions not necessarily have to be (isosceles) triangles as in this example. For instance, Gaussian membership functions

$$u(x) \quad = \quad \exp\left(\frac{-(x-c)^2}{2\sigma^2}\right)$$

provide an important alternative (e.g., see [13, 15, 29]). A further possibility for the shape of membership functions is to choose trapezoidals [14].

The correct choice of the membership functions, however, is by no means trivial but plays a crucial role in the success of an application. Several example applications demonstrate that evolutionary algorithms are capable of optimizing the membership functions of fuzzy logic controllers. The basic idea is to represent the complete set of membership functions by an

individual and to evolve shape and location of the triangles (respectively the Gaussian curves). Each triangle may be described by its anchor points on the abscissa axis, and the Gaussian membership functions are characterized by $c$ and $\sigma$.

Karr describes an application to the cart-pole balancing system and uses a genetic algorithm to evolve the membership functions of a fuzzy controller [13]. In order to evaluate the fitness of a controller, the system is simulated for a fixed simulation time, repeating the simulation four times for different initial conditions. The resulting, optimized fuzzy logic controller turns out to perform by far better than the controller based on membership functions designed by a human expert. Moreover, Karr also demonstrates that the genetic algorithm approach can be used successfully when the membership functions have to be altered in real time due to a variation of the cart mass (the expert-designed controller failed totally in this case) [13]. These promising results were recently confirmed by an application of the method for the online control of a laboratory pH system with drastically changing system characteristics [14].

Of course, rather than using a binary encoding of continuous parameters which characterize membership functions and a genetic algorithm, one might prefer an evolution strategy to optimize the membership functions. Wienholdt reports good results from an application of ESs to optimize *radial basis function* (RBFs — Gaussian membership functions) networks for time series prediction [29].

We conclude this section by referring to a problem which may arise from unconstrained variations of the membership function shape by the optimization algorithm: The *completeness* property, which requires that a fuzzy logic controller always be able to infer a control action for every state of the process (see [15]) might be violated if the anchor points of membership functions are shifted such that the possible range of values is no longer completely covered. In order to solve this problem, one might consider to introduce special constraints to the evolutionary algorithms' objective function.

### 2.2 Fuzzy Classifier Systems

Besides learning the membership functions, an even more challenging problem consists in the automatic learning of fuzzy *control rules*, i.e., the linguistic statements which are normally derived from expert knowledge [15]. This idea comes close to so-called *classifier systems* [4], rule-based systems which use a genetic algorithm as a rule-generation mechanism, such that the classifier system is capable of inductive learning [11]. Valenzuela-Rendón presented an extension of classifier systems which allows for the learning of fuzzy rules by incorporating fuzzyfication and defuzzyfication components as well as a fuzzy rule matching mechanism [27, 28]. Although the first application examples (the imitation of static linear respectively quadratic one-input one-output systems) are very simple, the approach may indicate a path towards automatic learning of fuzzy control rules. A promising further step into this direction was recently presented by Parodi and Bonelli, who extended Valenzuela-Rendón's approach by a mechanism which allows learning of fuzzy rules, membership functions, and output weights representing the relative importance of the fuzzy rules at the same time [20].

## 3 Conclusions

Evolutionary algorithms clearly represent a successful approach towards the optimization of fuzzy membership functions, and we expect this field of applications to grow remarkably in the near future. Of course, the method is only applicable if the quality function of the resulting fuzzy controller can be evaluated numerically without taking too much time, because

evolutionary algorithms typically require the evaluation of a large number of individuals. On the other hand, the global search characteristics of these algorithms yield membership functions of surprisingly high quality in comparison with those defined by human experts.

Research concerning the automatic learning of fuzzy control rules by means of evolutionary algorithms respectively classifier systems is surely in its initial stage and much work remains to be done. Nevertheless, we are sure that this line of research should also be followed, because an automatic tool for this time-consuming task is highly desirable.

The development and optimization of fuzzy controllers are important topics for further research where evolutionary algorithms will surely prove to be very helpful.

## References


[1] Th. Bäck (1992) The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm, In: Männer and Manderick [17], p. 85–94.

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

[3] R. K. Belew and L. B. Booker (Eds.) (1991) *Proceedings of the 4th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA.

[4] L. B. Booker, D. E. Goldberg, and J. H. Holland (1989) Classifier systems and genetic algorithms, In: J. G. Carbonell (Ed.), *Machine Learning: Paradigms and Methods*, The MIT Press / Elsevier, p. 235–282.

[5] D. B. Fogel (1991) *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*, Ginn Press, Needham Heights.

[6] D. B. Fogel (1992) *Evolving Artificial Intelligence*, PhD thesis, University of California, San Diego, CA.

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

[8] D. E. Goldberg (1989) *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, Reading, MA.

[9] F. Hoffmeister and Th. Bäck (1992) Genetic self–learning, In: F. J. Varela and P. Bourgine (Eds.), *Proceedings of the First European Conference on Artificial Life*, The MIT Press, Cambridge, MA, p. 227–235.

[10] J. H. Holland (1975) *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI.

[11] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard (1986) *Induction: Processes of Inference, Learning, and Discovery*, MIT Press.

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

[13] C. L. Karr (1991) Design of an adaptive fuzzy logic controller using a genetic algorithm, In: Belew and Booker [3], p. 450–457.

[14] C. L. Karr and E. J. Gentry (1993) Fuzzy control of pH using genetic algorithms, *IEEE Transactions on Fuzzy Systems*, 1(1):46–53.

[15] C. C. Lee (1990) Fuzzy logic in control systems: Fuzzy logic controller — Part I, *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):404–418.

[16] C. C. Lee (1990) Fuzzy logic in control systems: Fuzzy logic controller — Part II, *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):419–435.

[17] R. Männer and B. Manderick (Eds.) (1992) *Parallel Problem Solving from Nature 2.* Elsevier, Amsterdam.

[18] Z. Michalewicz (1992) *Genetic Algorithms + Data Structures = Evolution Programs*, Artificial Intelligence, Springer, Berlin.

[19] H. Mühlenbein (1992) How genetic algorithms really work: I. mutation and hillclimbing, In: Männer and Manderick [17], p. 15–25.

[20] A. Parodi and P. Bonelli (1993) A new approach to fuzzy classifier systems, In: S. Forrest (Ed.), *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, p. 223–230.

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

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

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

[24] H.-P. Schwefel (1987) Collective phenomena in evolutionary systems, In: *Preprints of the 31st Annual Meeting of the International Society for General System Research, Budapest*, volume 2, p. 1025–1033.

[25] H.-P. Schwefel (1990) Systems analysis, systems design, and evolutionary strategies, *Syst. Anal. Model. Simul.*, 7(11/12):853–864.

[26] G. Syswerda (1989) Uniform crossover in genetic algorithms, In: J. D. Schaffer (Ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, p. 2–9.

[27] M. Valenzuela-Rendón (1991) The fuzzy classifier system: A classifier system for continuously varying variables, In Belew and Booker [3], p. 346–353.

[28] M. Valenzuela-Rendón (1991) The fuzzy classifier system: Motivations and first results, In: H.-P. Schwefel and R. Männer (Eds.), *Parallel Problem Solving from Nature — Proceedings 1st Workshop PPSN I*, volume 496 of *Lecture Notes in Computer Science*, Springer, Berlin, p. 338–342.

[29] W. Wienholt (1993) Optimizing the structure of radial basis function networks by optimizing fuzzy inference systems with evolution strategy, Internal Report IR-INI 93-07, Institut für Neuroinformatik, Ruhr-Universität Bochum.