A Multivalued Evolutionary Algorithm

Hans–Michael Voigt *,
Joachim Born † & Ivan Santibanez–Koref †

TR-93-022

April 1993

Abstract

With this paper we present a Multivalued Evolutionary Algorithm (MEA) which is inspired by fuzzy set theory. The genetic representation and encoding is done in such a way that no inferences can be drawn from phenotype to genotype. This representation influences the used genetic operators. The basic operators of the algorithm will be explained and comparisons for global optimization problems with recently published results will be presented.

[1]International Computer Science Institute (ICSI),1947 Center Street, Suite 600, Berkeley, CA 94704, e–mail: voigt@icsi.berkeley.edu. On leave from the Technical University Berlin, Bionics and Evolution Techniques Laboratory, Bio – and Neuroinformatics Research Group, Ackerstrasse 71 – 76 (ACK1), D – 1000 Berlin 65, e-mail: voigt@fb10.tu–berlin.de and the Center for Applied Computer Science (GFaI), Rudower Chaussee 5, D – 1199 Berlin

[2]Technical University Berlin, Bionics and Evolution Techniques Laboratory, Bio – and Neuroinformatics Research Group

1 Introduction

The evolution of a population including genetic laws can be perhaps sketched by Figure 1. This figure reflects the generation of new individuals as replicating units of a population by genetic operators like recombination, crossing–over, inversion, deletion including erroneous replication due to mutations. By these operators one gets the genotype of an individual. Even these operators are subject to influences from the environment, especially with respect to mutations by e.g. radiation. The genotype will then be expressed via several stages also influenced by the environment, e.g. by metabolic processes, to a structural phenotype. This intermediate stage goes through an individual adaptation process to be a mature phenotype. All these different levels of development from a genotype to a mature phenotype may be determined as a developmental process corresponding to Figure 2. Once more it should be emphasized that the necessary transformations are all subject to environmental influences. This phenotype represents a number of attributes which are aggregated to a fitness value. Here fitness is also determined by the environment which on the other hand generates selection processes over the fitness values. These selection schemes include e.g. intra– and inter–specific cooperation and competition, mating success, figure of merit for offspring production etc.

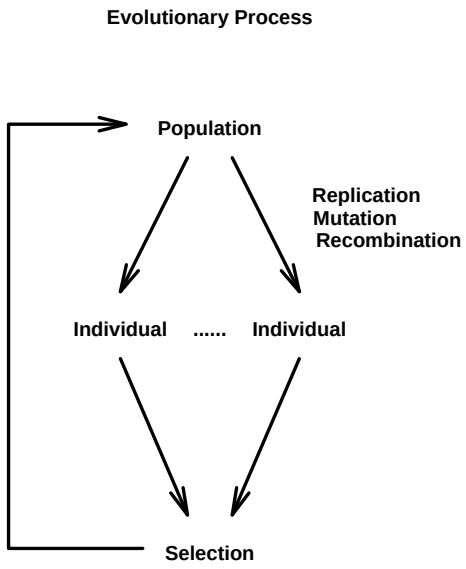

Figure 1: General Evolution Scheme

The modeling of the developmental process can be done on different levels and with different goals. For the structuring of artificial neural networks we used e.g. generative grammars [18], especially Lindenmayer–Systems [6]. Here we want to model such a

developmental process as a multivalued decision process, closely related to fuzzy set theory [19], [5], to emphasize the complexity of development from genotype to a mature phenotype. Generic to the incorporation of individual development into an overall evolution process is the impossibility of drawing inferences from phenotype to genotype, i.e. the mapping from genotype to phenotype is not isomorphic. The MEA follows this line of consideration.

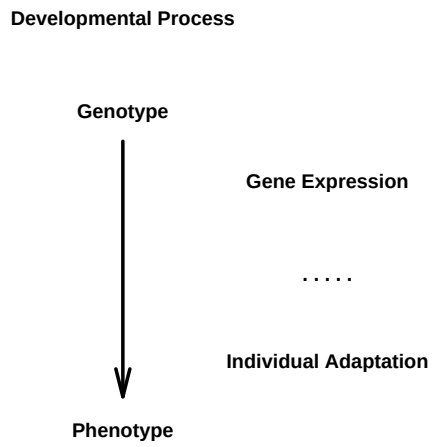

Figure 2: General Developmental Scheme

The proposed MEA is by no means an absolutely new algorithm. As it will be clear by the subsequent sections most of the genetic operators are inspired by Evolution Strategies [9], [10], [11], [12] and Genetic Algorithms [4], [2] and [3]. Especially the mutation operator was adapted from the Breeder Genetic Algorithm (BGA) [8] and extended to a more global orientation. Parts of the algorithm were first proposed in [16].

2 Representation

Let us assume that there is a fitness function $f(x)$, dependent on the phenotypic features $x_i \in G_i \subset R, i = 1, ..., n$ in the feasible region G_i. The phenotypic features can be scaled to x_{si} on the unit interval

$$s_i : G_i \rightarrow (0, 1). \tag{1}$$

Every scaled phenotypic feature should be influenced by a number of genes $a_{ij}, i = 1, ...n, j = 1, ..., m$. The a_{ij} are from the unit interval $a_{ij} \in (0, 1)$. Furthermore we have a mapping function g such that

$$g : (0, 1)^m \rightarrow (0, 1) \tag{2}$$

which maps the gene values to the scaled phenotypic features. The mapping g may be any meaningful coding function as e.g.

$$x_{si} = \frac{1}{2^{m-1}-1} \sum_{j=1}^{m} a_{ij} 2^{j-1} \tag{3}$$

well known from Genetic Algorithms. This coding function will be used in the subsequent sections.

This representation implies a number of consequences. For $m = 1$ this representation corresponds to the representation used in Evolution Strategies, it is a one–to–one mapping. For $m > 1$ the mapping is no longer isomorphic. More than one gene configuration may lead to the same phenotypic feature. As we will see it influences the functioning of the used genetic operators.

3 Selection

The used truncation selection corresponds to the $(\mu+\lambda)$-selection in Evolution Strategies where μ is the number of individuals in the parent population and λ the number of offspring produced by these parents. The selection intensity is then given by the equation

$$I = \frac{\phi(z)}{1 - \Phi(z)} \tag{4}$$

where $\phi(z)$ is the probability density function and $\Phi(z)$ the probability distribution function of the normalized and standardized normal distribution. Typical values are given in Table 1. where $T = (1 - \Phi(z))$ corresponds to the percentage rate of the selected best individuals to form the new parent population.

T	80 %	50 %	40 %	28 %	20 %	10 %	1 %
I	0.34	0.8	0.97	1.2	1.4	1.76	2.66

Tab. 1. Selection intensity for $N \to \infty, N$ population size

Because this is a very powerful measure for the selection pressure as it was shown in [8] we will use it in the following.

4 Mutation

For mutations we adopted and extended the BGA mutation scheme. With the probability p_m a phenotypic feature x_i will be selected and mutated corresponding to

$$x_i := x_i \pm \Delta_i \tag{5}$$

where Δ_i will be distributed at random over the gene values such that

$$a_{ij} := a_{ij} + \Delta_{ij} \tag{6}$$

and

$$\Delta_i = g(\Delta_{i1}, ..., \Delta_{im}) \tag{7}$$

in accordance with the coding hold. The mutation range A_i will be set usually to the value $0.1 \cdot G_i$. Thus, mutations act in the same way as for the BGA.

- **BGA mutation scheme**[8]

  The BGA mutation operator generates at random one of the points as Δ

  $$\pm \{2^{-15}A, 2^{-14}A, ...2^0 A\}. \tag{8}$$

  It is a robust mutation operators which approximates the optimum with a limited precision. The normalized expected progress is given by

  $$\frac{1}{32} \leq \frac{E(R)}{R} \leq \frac{1}{16}. \tag{9}$$

- **Extended BGA mutation scheme**

  To have more global oriented mutations combined with the robust features of the simple BGA mutation scheme we use also as Δ randomly chosen points from

  $$\pm \{2^{-15}A, 2^{-14}A, ...2^0 A, \frac{1}{16}(G-A)+A, \frac{2}{16}(G-A)+A, ..., \frac{16}{16}(G-A)+A\}. \tag{10}$$

  In this case the normalized expected progress is given by

  $$\frac{1}{64} \leq \frac{E(R)}{R} \leq \frac{1}{32}. \tag{11}$$

If $p_m = 1/n$ and $f(x)$ is a rotation invariant unimodal function the normalized expected progress slows down by $1/n$ [8] for both mutation schemes.

5 Recombination

For the mixing of the genetic material we have implemented two different recombination operators.

- **Multivalued Discrete Recombination**

  Let $x = (a_{11}, a_{12}, ..., a_{n(m-1)}, a_{nm})$ and $y = (b_{11}, b_{12}, ..., b_{n(m-1)}, b_{nm})$ be the parent strings. Then a child $z = (c_{11}, c_{12}, ..., c_{n(m-1)}, c_{nm})$ is given by

  $$c_{ij} = a_{ij} \quad or \quad c_{ij} = b_{ij} \tag{12}$$

  where either equation is used with probability 0.5. For $m = 1$ this corresponds to the BGA discrete recombination, it generates corners of the hypercube spanned by the components of x and y. For $m > 1$ and $a_{ij} \leq b_{ij}, \forall i, \forall j$ it generates corners or discrete interior points of the hypercube. For $m > 1$ and $a_{ij} \leq b_{ij}, \forall i, \neg\forall j$ it generates corners and discrete interior and exterior points of the hypercube.

- **Multivalued intermediate recombination**

  $$c_{ij} = a_{ij} + \alpha_{ij}(b_{ij} - a_{ij}) \tag{13}$$

  where α_{ij} is selected with uniform probability from the unit interval (0,1). For $m = 1$ or for $m > 1$ and $a_{ij} \leq b_{ij}, \forall i, \forall j$ a point will be chosen from the interior and the boundary of the hypercube spanned by the parents components. For $m > 1$ and $a_{ij} \leq b_{ij}, \forall i, \neg\forall j$ the generated points may be also outside the hypercube.

Each recombination operator has its own problem solving properties. Multivalued intermediate recombination with $m > 1$ is e.g. very useful for nonseparable optimization problems. As proposed in [8] higher order multivalued recombination operators will be introduced in the future.

To confirm the empirical laws concerning selection and discrete recombination formulated in [8] for different values of m we made the same experiments with $m = 1$ and $m = 2$ for the the functions $F_0 = \sum_{i=1}^{n} |x_i|$ and $F_6(x) = n \cdot 10 + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$ with $-5.12 \leq x_i \leq 5.12$. Though F_6 is a highly multimodal function it shows a unimodal metastructure [15]. In the following gen denotes the number of generations until convergence, σ_{gen} the standard deviation of gen, f_{eval} the number of function evaluations, and Δf the distance of the best found value to the global optimum. Table 2. shows that the observations in [8] hold equally for $m = 1$ and $m = 2$. Higher values of m lead to an increase in gen and therefore in f_{eval} but to a better quality approximation.

Table 3. shows the quality of solution for different population sizes N and fixed selection intensity $I = 1.2$. It is remarkable that for a large population sizes N the quality of approximation converges for different m.

N	F	I	gen		σ_{gen}		f_{eval}		Δf	
			1	2	1	2	1	2	1	2
256	F_0	1.6	14.3	20.7	0.9	2.3	3904	5555	3.8	2.5
256	F_6	1.6	14.1	20.6	1.1	3.0	3853	5530	29.7	32.2
192	F_0	1.2	19.2	31.0	1.7	3.0	3878	6144	3.3	2.2
192	F_6	1.2	19.5	29.8	1.3	2.6	3936	5904	28.4	25.1
128	F_0	0.8	25.0	42.6	1.9	4.3	3322	5581	3.6	2.2
128	F_6	0.8	25.5	43.5	1.7	4.7	3392	5690	31.6	25.9

Tab. 2. Discrete recombination, $n = 20$, 20 runs

N	gen		σ_{gen}		Δf		f_{eval}	
	1	2	1	2	1	2	1	2
32	9.7	12.4	1.4	2.1	109.0	90.4	341	427
128	17.1	25.0	1.5	2.6	40.6	33.9	2317	3322
256	20.6	34.3	1.3	3.5	24.1	19.5	5517	9037
512	24.1	47.2	1.5	3.6	10.9	10.9	12826	24653
1024	26.8	63.2	1.3	6.4	5.7	5.2	28467	65741

Tab. 3. Quality of solution for $F_6, I = 1.2, n = 20$

Table 4. shows the simulation results for constant N/I and different numbers of variables n. We have chosen smaller population sizes then [8] because of the observations in Table 3. The observation made in [8] concerning the $quotient = gen(2n)/gen(n)$ hold for different m but the quality of approximation is in any case better for $m = 2$.

6 Mutation and Recombination

Within this section we compare the influence of mutation and discrete recombination for different m on the performance of the algorithm.

Table 5. shows the results for F_6 and Table 6. for F_0. The observation made by [8] for $m = 1$ has to be emphasized once more also in the case $m > 1$: **Recombination and mutation (r&m) does in any case better then a single genetic operator (recombination r or mutation m)**.

n	N	I	gen		σ_{gen}		Δf		$quotient$	
			1	2	1	2	1	2	1	2
20	128	1.6	11.5	15.2	0.9	2.3	58.0	44.6		
40	128	1.6	15.2	19.3	0.9	2.2	154.8	140.0	1.32	1.27
80	128	1.6	21.0	24.2	1.2	2.1	402.3	370.5	1.38	1.25
160	128	1.6	27.8	29.1	1.6	3.7	1047.0	996.5	1.32	1.20
20	64	1.2	13.9	17.4	1.4	2.3	68.4	57.9		
40	64	1.2	18.7	22.9	1.4	2.4	189.2	152.7	1.34	1.32
80	64	1.2	24.7	29.4	1.6	2.2	505.7	410.2	1.32	1.28
160	64	1.2	32.2	37.9	1.7	3.5	1267.8	995.9	1.30	1.29
20	32	0.8	16.3	22.6	1.6	3.4	95.9	66.7		
40	32	0.8	20.5	28.6	1.7	2.9	253.6	182.9	1.26	1.27
80	32	0.8	29.1	33.3	2.4	4.6	639.4	485.1	1.42	1.16
160	32	0.8	35.3	40.0	2.6	4.8	1566.1	1204.6	1.21	1.20

Tab. 4. Quality of solution for F_6 for constant N/I

N	op	gen		σ_{gen}		Δf		f_{eval}	
		1	2	1	2	1	2	1	2
20	r	6.1	6.6	1.4	1.6	172.02400	130.13200	141	152
20	m	539.3	505.2	34.9	39.8	0.00009	0.00009	10806	10124
20	r&m	410.5	369.6	42.6	49.3	0.00009	0.00009	8229	7412
256	r	17.0	25.8	1.2	3.9	25.38160	6.59073	4595	6848
256	m	289.8	271.5	16.9	11.9	0.00009	0.00009	74445	69760
256	r&m	101.4	118.0	5.9	8.0	0.00009	0.00009	26214	30451

Tab. 5. Recombination, mutation, and both applied to $F_6, I = 1.4$

But it interesting to notice that for both functions for a small population size the performance for $m = 2$ is slightly better than for $m = 1$. The opposite is true for large population sizes. This conforms with the observations from Table 3.

N	op	gen		σ_{gen}		Δf		f_{eval}	
		1	2	1	2	1	2	1	2
20	r	5.7	6.6	1.1	1.4	20.7992	14.6232	133	152
20	m	539.4	502.8	22.7	28.1	0.0005	0.0005	10808	10075
20	r&m	381.6	356.2	33.0	40.4	0.0005	0.0005	7651	7143
256	r	17.2	26.5	1.3	2.7	2.8789	2.0811	4659	7027
256	m	292.3	278.3	8.7	8.7	0.0005	0.0005	75072	71488
256	r&m	104.5	108.0	3.2	4.0	0.0005	0.0005	27008	27891

Tab. 6. Recombination, mutation, and both applied to $F_0, I = 1.4$

7 Performance

For the performance evaluation of the MEA we use the functions $F_6 - F_8$ already used in [7], [15], and [17]. They are usual test functions in global optimization [14]. Function F_9 was proposed in [1]. So we are using the same test bed as in [8] for comparison. The test functions are listed in Table 7. For F_7 we used the extended BGA mutation scheme and for F_8 multivalued intermediate recombination. The termination criterion was used as in [15] and [8]. We stopped the computation if one of the expressions $|f^* - f^{best}| \leq \epsilon \cdot |f^{best}|$ or $f^* - f^{best}| \leq \epsilon$, with f^* the global optimum, was true.

It is interesting that all test functions have been solved with a constant population size. The results of the MEA are within the performance range of the BGA. For F_7 we got better results with even a constant population size. The number of function evaluations scales almost exactly with $n \cdot \ln(n)$ even for F_7.

Function	Constraints
$F_6(x) = n \cdot 10 + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$	$-5.12 \leq x_i \leq 5.12$
$F_7(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$-500 \leq x_i \leq 500$
$F_8(x) = \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$	$-600 \leq x_i \leq 600$
$F_9(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	$-30 \leq x_i \leq 30$

Tab. 7. Test Examples

Rastrigin's Function F_6					Schwefel's Function F_7				
n	N	f_{eval}			n	N	f_{eval}		
		1	2	BGA			1	2	BGA
20	20	4234	3610	3608	20	20	3630	4599	16100
100	20	26949	27592	25040	100	20	25158	26840	92000
200	20	66954	65077	52948	200	20	69420	75771	248000
400	20	132744	136638	112634	400	20	150880	161236	699803
1000	20	380068	411208	337570	1000	20	319300	341232	

Tab. 7. $\epsilon = 9.0 \cdot 10^{-1}$ for F_6, $\epsilon = 5.0 \cdot 10^{-3}$ for F_7

Griewangk's Function F_8					Ackley's Function F_9				
n	N	f_{eval}			n	N	f_{eval}		
		1	2	BGA			1	2	BGA
20	500	——–	23625	66000	30	20	15110	14064	19420
100	500	——–	337925	361722	100	20	55582	51415	53860
200	500	——–	805950	748300	200	20	119542	106578	107800
400	500	——–	1676000	1630000	400	20	245924	228020	220820
					1000	20	672132	623190	548306

Tab. 8. $\epsilon = 10^{-3}$ for F_8, $\epsilon = 10^{-3}$ for F_9

In [13] a recent comparison of global optimization methods was done. They wrote *'From the test results of each method on a series of test functions, it appears that these methods out–perform other commonly used methods of global optimization'*. They did not consider Evolutionary Algorithms.

Function	Constraints
$f_{GP}(x) = \{1 + (x_1 + x_2 + 1)^2 \cdot(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\} \cdot\{30 + (2x_1 - 3x_2)^2 \cdot(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\}$	$-10.0 \le x_i \le 10.0$
$f_{BR}(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + e$ with $a = 1, b = 5.1/(4\pi^2), c = 5/\pi,$ $d = 6, e = 10, f = 1/(8\pi)$	$-5 \le x_1 \le 10$ $0 \le x_2 \le 15$
$f_{CH}(x) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$-3 \le x_1 \le 3$ $-2 \le x_2 \le 2$
$f_{SH}(x) = \{\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\}\{\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\}$	$-10 \le x_i \le 10$
$f_{EA} = -\cos(x_1)\cdot\cos(x_2)/\exp((x_1 - \pi)^2 + (x_2 - \pi)^2)$	$-100 \le x_i \le 100$

Tab. 9. Test Examples Bayesian/Sampling–methods

They used different methods and wrote *'Perttunen's method, Zilinskas' method and Shaltenis' method required an excessive amount of CPU time to run (in some cases exceeding 1 h on a VAX 8650)'*. The test functions had 2 and 4 dimensions. The 2–dimensional functions are listed in Table 9.

Table 10. shows the results for the different methods compared to the results of the MEA with multivalued intermediate recombination. For f_{EA} we used also the extended BGA mutation scheme.

No one of the global optimization methods converged uniformly in all cases. The methods of Zilinskas and Shaltenis were discarded because of the high CPU requirements. The MEA out-performed all methods almost uniformly.

F	N	f_{eval}	f^*_{eval}					
			S	M	P	T	C	A
GP	30	248	> 1000	> 1000	≤ 200	≤ 500	> 1000	> 1000
CH	30	158	≤ 500	≤ 500	≤ 200	≤ 500	> 1000	> 1000
BR	30	216	≤ 500	> 1000	≤ 200	≤ 500	> 1000	> 1000
SH	30	339	> 1000	> 1000	≤ 500	> 1000	> 1000	> 1000
EA	30	512	≤ 1000	> 1000	**	> 1000	> 1000	> 1000

Tab. 10. $\epsilon = 10^{-3}, I = 1.4$, 20 runs, f^*_{eval} solutions with B/S–methods S: Stuckman, M: Mockus, P: Perttunen, T: Torn, C: Monte Carlo, A: Simulated Annealing, ** no evaluation due to extreme CPU requirements

8 Conclusions

The MEA is a very robust optimization algorithm. In some cases it has a better performance than the BGA. In any case it out-performed methods from global optimization not based on evolution paradigms. It scales like $n \cdot \ln(n)$ for n up to 1000. Future research will be the inclusion of higher order recombination schemes and the implementation of distributed population models [8], [17].

References

[1] Ackley, D. (1987): A connectionist Machine for Genetic Hillclimbing. Boston: Kluver Academic Publishers

[2] DeJong, K.A. (1975): An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD Thesis, University of Michigan, Dissertation Abstracts International 36(10),5140B.(University Microfilms No. 76-9381)

[3] Goldberg, D.E. (1989): Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley

[4] Holland, J.H. (1975): Adaption in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press

[5] Kosko, B. (1992): Neural Networks and Fuzzy Systems. Englewood Cliffs: Prentice Hall 1992

[6] Lindenmayer, A. (1968): Mathematical Models for Cellular Interaction in Development, Parts I and II. J. Theor. Biology (18) 280–315.

[7] Mühlenbein, H.; Schomisch, M.; Born, J (1991): The Parallel Genetic Algorithm as Function Optimizer. Parallel Computing, 17:619–632, 1991

[8] Mühlenbein, H. & D. Schlierkamp–Voosen (1993): Predictive Models for the Breeder Genetic Algorithm. In: Evolutionary Computation No. 1. Cambridge: MIT Press

[9] Rechenberg, I. (1973): Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: F. Frommann

[10] Rechenberg, I. (1993): Evolutionsstrategie '93. Stuttgart: F. Frommann

[11] Schwefel, H.–P. (1977): Numerische Optimierung von Computer Modellen mittels der Evolutionsstrategie. Basel, Stuttgart: Birkhäuser Verlag.

[12] Schwefel, H.–P. (1981): Numerical Optimization of Computer Models. New York, London: John Wiley

[13] Stuckman, B. E. & E. E. Easom (1992): A Comparison of Bayesian/Sampling Global Optimization Techniques. IEEE Trans. Systems, Man and Cybernetics. Vol. 22, No. 5, pp. 1024–1032

[14] Toern, A. & A Zilinskas (1989): Global Optimization. Lecture Notes in Computer Science 350. Berlin: Springer-Verlag

[15] Voigt, H.-M., J. Born & J. Treptow (1991): The Evolution Machine. Manual, Version 2.1. Institute for Informatics and Computing Techniques. iir, inform., inf., rep. Berlin 7(1991)7, ISSN 0233–2582

[16] Voigt, H.–M. (1992): Fuzzy Evolutionary Algorithms. Technical Report TR–92–038, International Computer Science Institute, Berkeley, June 1992

[17] Voigt, H.–M., I. Santibanez–Koref & J. Born (1992): Hierarchically Structured Distributed Genetic Algorithms. In: R. Männer & B. Manderick (eds.): Parallel Problem Solving from Nature 2. Amsterdam, London, New York, Tokyo: North–Holland, pp. 155–164

[18] Voigt, H.–M., J. Born & I. Santibanez–Koref (1993): Structuring of Artificial Neural Networks by Generative Grammars. In: Proc. Seminar Neuroinformatik.Informatik Fachberichte. Berlin: Springer–Verlag

[19] Zadeh, L.A. (1965): Fuzzy Sets. Information and Control, vol. 8, 338–353, 1965