Publication P5

J. Martikainen and S. J. Ovaska
"Optimizing dynamical fuzzy systems using aging evolution strategies"
in
Proc. of the 9th IASTED International Conference on Artificial Intelligence and Soft Computing
Benidorm, Spain, 2005, pp. 5-10.

OPTIMIZING DYNAMICAL FUZZY SYSTEMS USING AGING EVOLUTION STRATEGIES

Jarno Martikainen
Helsinki University of Technology
Institute of Intelligent Power Electronics
P. O. Box 3000 FIN-02015 HUT
FINLAND
E-mail: jkmartik@cc.hut.fi
URL: http://powerelectronics.hut.fi

Seppo J. Ovaska
Helsinki University of Technology
Institute of Intelligent Power Electronics
P. O. Box 3000 FIN-02015 HUT
FINLAND
E-mail: ovaska@ieee.org
URL: http://powerelectronics.hut.fi

ABSTRACT
This paper introduces an evolutionary optimization algorithm taking advantage of multiple populations and an adaptive aging parameter to achieve faster and more robust convergence. As challenging test cases, the evolutionary algorithm is used to optimize parameters for dynamical fuzzy systems. Our results show that the proposed algorithm is capable of outperforming the traditional reference algorithm. The effect of sampling the membership functions of the dynamical fuzzy system in feed-forward and feedback configurations is also studied.

Keywords: Evolution strategy, multipopulation, adaptive evolutionary algorithm, dynamic fuzzy system

1. INTRODUCTION

In [7] the authors presented an evolutionary computation scheme [6] in which instead of concentrating to enhance a single solution, the characteristics of the whole solution population were adjusted in order to produce competitive results. The presented multi-population genetic algorithm (MPGA), later renamed as *n*-population genetic algorithm (*n*PGA), divided the solution population into elite and plain populations with different characteristics allowing also interpopulation migration of solutions. The proposed scheme proved to produce better results more reliably compared to a standard genetic algorithm.

In this paper we extend our *n*PGA scheme by introducing an adaptive age parameter to the system. Besides, we use an evolution strategy (ES) instead of a genetic algorithm (GA). Age parameters in evolutionary algorithms have been used before followed by improvement in performance [4,5]. Instead of simply trying to determine the remaining age of an individual solution, we take into account characteristics concerning the whole solution population instead of that solution alone. This fits well in the earlier *n*PGA scheme, in which we aim to produce favorable environment for producing quality results reliably, instead of trying to fine tune single solutions. Especially in time-consuming optimization tasks the reliability of an algorithm is crucial, and our results show that using the *n*PGA scheme ($\rightarrow$ *n*PES) accompanied with adaptive age parameters helps to produce an evolution strategy with competitive performance.

The performance of the proposed algorithm is evaluated by optimizing the parameters of dynamical fuzzy systems containing linguistic information feed-forwards or feedbacks [8]. The parameters of such dynamic fuzzy systems have not been optimized before using evolutionary computation.

This paper is structured as follows. Section 2 introduces the proposed evolution strategy. Section 3 discusses the fuzzy systems used as test cases. Section 4 summarizes the results and Section 5 finally concludes the paper.

2. EVOLUTIONARY COMPUTATION

Evolutionary computation is divided into three main categories, namely genetic algorithms, evolutionary programming (EP), and evolution strategies [6]. Each of these nature-inspired approaches to optimization is based on the principles of recombination, mutation, and selection; the form of which varies from implementation to another.

2.1. EVOLUTION STRATEGIES

In this paper, evolution strategies were used for the sake of simplicity of implementation. The methods proposed here can be directly implemented also in other evolutionary computation methods with only minor modifications. The very basic evolution strategy is encapsulated by Fogel in [6]:

1. Generate initial population of parent vectors, $\mathbf{x}_i, i = 1, 2, \ldots, P$.
2. Create offspring from each parent vector by adding a Gaussian random variable with zero mean to each component of $\mathbf{x}_i$.

3. Select P best vectors to act as the parents for the next generation.
4. These steps are repeated until satisfactory results have been achieved, or run-time requirements have been met.

2.2 *n*PES EVOLUTION STRATEGY

Our *n*PES scheme aims to emulate animal and human populations by dividing the whole solution population into *elite* and *plain* populations. In the elite population, animals and humans are better considering a certain skill or characteristic. In evolution strategy this means that the solutions in the elite population are on the average better than the ones in the plain population. Solution vectors in the elite population also undergo smaller mutations than the ones in the plain population. So, in principle, the elite population conducts local search among the already fit-proven individuals, whereas the plain population conducts global search. Once assigned to elite population, the individual may fall back to the plain population if there are good candidate solutions in the plain population. The *n*PES algorithm can be summarized as follows.

1. Generate an initial population of *N* (here 20) random vectors
2. Run the ES algorithm for *G* generations.
3. At the end of the *G*th generation, arrange the solutions into ascending order based on their cost. Assign the *K* (here 5) best solutions to the elite population and the $N-K$ other solutions to the plain population.
4. Operate the two populations separately using different variances for the Gaussian random variables in the different populations.
5. If the cost of the best solution in the plain population is lower than that of the best individual in the elite population, exchange the worst elite solution and the best plain solution.
6. Continue until satisfactory convergence is reached, or runtime requirements are met.

The two-population scheme causes only a slight increase in the in the administration costs, but the number of solution evaluations remains the same.

2.3 AGING PARAMETER

Aging of individuals affects all biological systems, the clearest impact of which is the fact that the individual perishes after it has reached its maximum age. Various aging schemes have been implemented in evolutionary algorithms to control the performance of the algorithm. Most papers support the idea of preventing stagnation by prohibiting the domination of a single solution for a very long time [1,4]. So, aging parameter increases the diversity in evolutionary computation.

According to the aging scheme presented here, each candidate solution is given a life time when it is created. Each solution is assigned the same life time, which is the number of generation the solution is going to survive. Elitism in our aging scheme is preserved by adding one to the life time of the best solution of each generation. When a solution dies away, its replacement is created from the best candidate solution in the same way as an offspring would be created.

2.4 ADAPTIVE AGE PARAMETER

Apart from a constantly decreasing age parameter, we designed a system for modifying the age of a candidate solution adaptively. An adaptive parameter was implemented in order to improve the performance of the algorithm, since a preset parameter values rarely are optimal for the entire runtime of the algorithm. Fuzzy logic systems have been successfully controlling evolutionary algorithms previously for example in [9,10]. In other words, the age of a solution was modified using fuzzy logic system based on the fitness of the offspring created from a parent. If the parent was able to produce fit offspring, the age of the parent was increased and vice versa.

The adaptive life time fuzzy system includes two inputs and one output. The inputs describe the value of an individual offspring relative to the best candidate solution, and also the dynamic cost range of the whole population. The input concerning the improvement, Δc , over the best cost by a candidate solution is calculated as

$$\Delta c = 1 - c_o / c_b \tag{1}$$

where c_b denotes the cost of the best individual, and c_o denotes the cost of the offspring. The dynamic cost range, Δd , is calculated as

$$\Delta d = 1 - c_b / c_w \tag{2}$$

where c_w denotes the cost of the worst individual.

Fig. 1 shows the improvement, dynamic range, and life time membership functions used in the systems.

In this study, we optimized two kinds of dynamic fuzzy systems: *linguistic information feed-forward-based dynamical fuzzy system* (LIFFDFS) [2] and *linguistic information feedback-based dynamical fuzzy system* (LIFDFS) [3]. Sample optimization runs demonstrated different dynamic in the solution pool, so individual fuzzy membership functions were used for these two schemes. The membership function describing the improvement was the same for both of the systems.

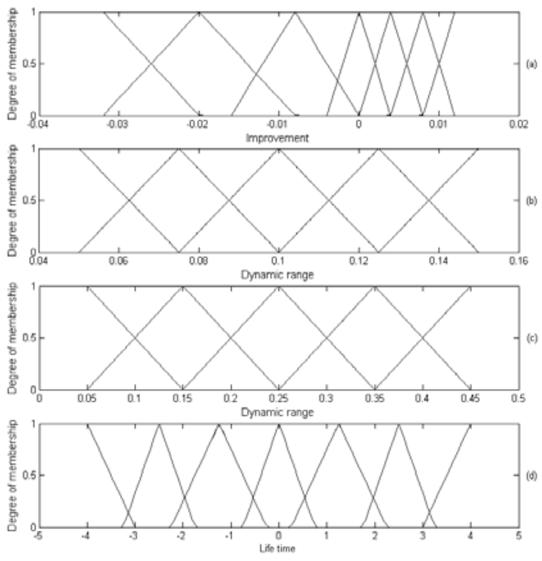

Fig. 1. Fuzzy membership functions related the adaptive life time parameter. ((a) Δc, improvement, (b) Δd in LIFFDFS scheme, (c) Δd in LIFDFS scheme, (d) life time parameter.

The fuzzy rule base was constructed intuitively so that the remaining life time of a solution was increased if it produced a better solution than the previous best solution. The remaining life time was decreased if the produced solution was worse than the previous best. The amount of modification of the solutions' life time depends also on the dynamics of the population. Large improvement in cost when the dynamics is large does not increase the life time of the parent solution so much as it would when the dynamics is small. Dynamics is often much larger when the evolution strategy is still in its early stages and has not yet converged considerably toward the final value.

The applied fuzzy rule base is shown in Table 1. Improvement (**I**) can be negative large (NL), negative medium (NM), negative small (NS), zero (ZE), positive small (PS), positive medium (PM), positive large (PL). Dynamics (**D**) of the population can be very small (VS), small (S), medium (M), large (L), or very large (VL). The constant to be added to the life time of the solution can be negative large (NL), negative medium (NM), negative small (NS), zero (ZE), positive small (PS), positive medium (PM), or positive large (PL).

Table. 1. Fuzzy rule base used in determining the life time.

I \ D	**VS**	**S**	**M**	**L**	**VL**
NL	NL	NL	NL	NL	NL
NM	NM	NL	NL	NL	NL
NS	NS	NM	NL	NL	NL
ZE	ZE	ZE	ZE	NS	NM
PS	PS	PS	PS	ZE	ZE
PM	PM	PS	PS	PS	ZE
PL	PM	PM	PM	PS	PS

Both LIFFDFS and LIFDFS predictors were designed to predict the well-known 296-sample Box-Jenkins time series [11]. The cost of a candidate solution is calculated in this time series prediction task as

$$cost = \sum_{k=1}^{296} \left[y(k) - \tilde{y}(k) \right]^2 \tag{3}$$

where $y(k)$ is the actual time-series value and $\tilde{y}(k)$ is the output of the fuzzy predictor.

The principle of the complete *n*PES algorithm using the adaptive life time parameter is shown in Fig. 2. The algorithm is first initialized, and it is run as a single population algorithm for *G* generations after which it is divided into two populations for the rest of the simulation.

3. DYNAMICAL FUZZY SYSTEMS WITH LINGUISTIC INFORMATION FEEDFORWARD OR FEEDBACK

Fuzzy systems (FS) [8] are frequently used in industrial applications. They are capable of transforming human knowledge into computer operable programs. FS, however, usually lack dynamical properties. On the other hand, this is not the case in physical systems. To overcome this problem, Gao and Ovaska introduced linguistic information feedforward (FF) and feedback (FB) schemes in [2] and [3], respectively.

In both of those schemes previous outputs of a single fuzzy rule are aggregated to the current output of a fuzzy rule. The previous outputs are not combined as such, but they can be manipulated using scaling and shifting parameters, marked as *a* and *b*, respectively. The shifting parameter is able to shift the centre of the fuzzy rule output. And the Scaling parameter can scale the magnitude of the fuzzy rule output. It is imperative to note that the scaling can in our implementation have also values outside the range [0,1].

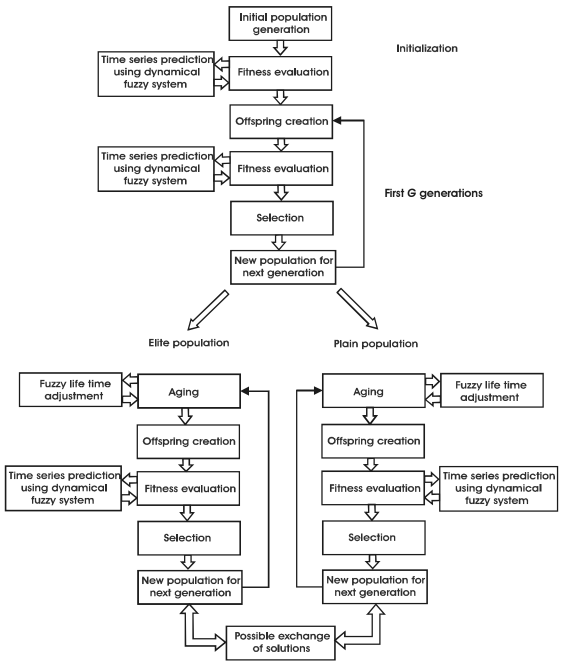
Initial population generation
Initialization
Time series prediction using dynamical fuzzy system
Fitness evaluation
Offspring creation
Time series prediction using dynamical fuzzy system
Fitness evaluation
First G generations
Selection
New population for next generation
Elite population
Plain population
Fuzzy life time adjustment
Aging
Aging
Fuzzy life time adjustment
Offspring creation
Offspring creation
Time series prediction using dynamical fuzzy system
Fitness evaluation
Fitness evaluation
Time series prediction using dynamical fuzzy system
Selection
Selection
New population for next generation
New population for next generation
Possible exchange of solutions

Fig. 2. The principle of the *n*PES algorithm with adaptive life time parameter.

In addition, it is possible to sample the feed-forward or feedback membership functions using different amount of samples. This is an issue related to the computational complexity of these systems, as well as their mapping accuracy.

3.1 LIFFDFS AND LIFDFS STRUCTURES

Linguistic Information Feed-Forward-based Dynamical Fuzzy System adds dynamics to a fuzzy system by aggregating the current output of a fuzzy rule and the one-step-delayed output of the same fuzzy rule. The structure of the LIFFDFS is shown in Fig. 3. Each rule has its own feed-forward loop.

There is a delayed membership function for each rule; and in our case there are 37 rules [2]. Each of these feed-forwards or feedbacks can be shifted (left/right) and/or scaled. This means that we have 74 different parameters to be optimized in our system.

Whereas LIFFDFS introduced limited dynamics taking into account the previous rule output, takes LIFDFS advantage indirectly of the whole output history of a single fuzzy rule. Fig. 4 shows the principle of the LIFDFS. The offspring were created similarly to that of the LIFFDFS scheme.

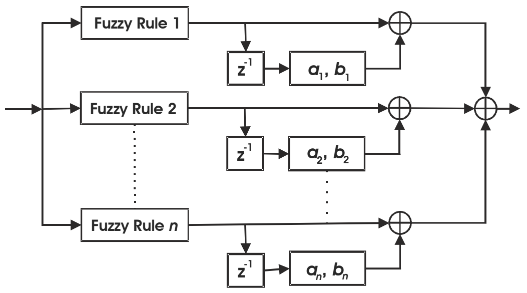

Fig. 3. Linguistic information feed-forward-based dynamical fuzzy system.

In the adaptive age *n*PES scheme the elite offspring was created as follows

$$a_{o,n} = a_{p,n} + \mathrm{round}(-1 + r \cdot 2) \tag{4}$$

$$b_{o,n} = b_{p,n} + 0.05(-1 + r \cdot 2) \tag{5}$$

The plain offspring was created as

$$a_{o,n} = a_{p,n} + \mathrm{round}(-5 + r \cdot 10) \tag{6}$$

$$b_{o,n} = b_{p,n} + 0.2(-1 + r \cdot 2) \tag{7}$$

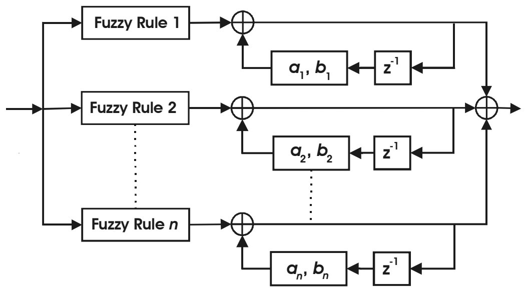

Fig. 4. Linguistic information feedback-based dynamical fuzzy system.

3.2 PLAIN EVOLUTION STRATEGY

The plain ES used for reference, is as presented in 2.1. The population size was 20 solutions and the algorithm was allowed to converge for 150 generations. 10 separate runs were conducted for statistical reliability. These values were selected based on the long running times of the algorithm. The offspring in the plain algorithm was created as follows

$$a_{o,n} = a_{p,n} + \text{round}\left(-4 + r \cdot 8\right) \quad (8)$$

$$b_{o,n} = b_{p,n} + 0.15\left(-1 + r \cdot 2\right) \quad (9)$$

where $a_{o,n}$ describes the offsprings shifting parameter for the *n*th rule, $a_{o,p}$ the parents shifting parameter for the *n*:th rule, “round” is a rounding operator, and *r* is a random variable with application specific variance.

4. OPTIMIZATION RESULTS

The optimization of LIFFDFS/LIFDFS parameters is computationally very challenging, since, among others, the fitness calculation of a single candidate solution requires evaluation of several membership functions for each sample value of the time series. Therefore, the number of membership functions' sampling points was optimized separately, and this result was then used when optimizing the shifting and scaling parameters of the system.

4.1 NUMBER OF SAMPLING POINTS

The results were surprising. Intuitively, the more points are used in the sampling the more accurate the feed-forward or feedback membership function is, and thus, the better the mapping result is. This was, however, not the conclusion from the results. As the results in Table 2 show, the system using five samples produced best results in terms of the median value. Ten separate tests were run for each number of samples.

The results can be interpreted that the sampling process may also aim to overcome the deficiencies of the original fuzzy system. If the selection and alignment of the original membership functions is not optimal, the situation could be improved by sampling the feedback using an application specific amount of samples.

Table 2. Performance of the plain ES using different number of samples.

# of samples	Average Cost	Std
1	470.74	10.94
5	416.64	8.63
10	438.50	9.36
25	443.49	4.50
150	436.13	6.60

4.2 ADAPTIVE LIFE TIME

Before implementing the fuzzy adaptive life-time scheme, different constant life-time parameters were tested, and the results are shown in Table 3. These test were carried out for the LIFFDFS.

Table 3. Performance of the *n*PES algorithm using constant life-time parameter.

Life Time	Average Cost	Std
2	415.96	5.00
3	424.56	7.86
5	419.19	4.11
10	422.11	7.36
25	417.18	7.55

As can be seen from Table 3, the performance of the algorithm varies depending on the life-time parameter. The life-time parameter being 2, it was clear that no single solution survived for long and replacing solutions were created from the best candidate solution at a rapid pace. Then again, using a large life-time parameter, the selection took care of discarding some solutions, and only a few new solutions were created because of the aging scheme. Creating a new solution to replace a perished solution because of aging adds to the computational costs of the algorithm.

Using adaptive life time parameter as described in 2.4. yielded better results. The improvement was not without extra cost. In the LIFFDFS scheme on the average 2.9 new solutions per generation were generated and evaluated. Then again, on the LIFDFS scheme the corresponding figure was 4.8.

Table 4. presents the results achieved optimizing the LIFFDFS and LIFDFS systems and using both the plain and adaptive methods. The results are averages of 10 runs.

The adaptive algorithm produces better results both in terms of average cost and standard deviation.

Figs. 5 and 6 illustrate the average convergence behavior of the same methods. Dashed line describing the adaptive method shows slightly faster convergence characteristics than the plain method.

Table 4. Performance of the *n*PES algorithms.

Algorithm	Average Cost	Std
Plain LIFFDFS	419.36	8.17
Adaptive LIFFDFS	413.12	5.42
Plain LIFDFS	197.66	10.81
Adaptive LIFDFS	193.50	8.64

5. DISCUSSION

In this paper, we introduced an evolution strategy using the *n*PES scheme that implemented an adaptive life-time extension. The results show improvement over the performance of a reference algorithm in a demanding parameter optimization application. Also, our research showed that the sampling fuzzy feed-forward of feedback membership functions using different number of points can enhance the performance of a dynamical fuzzy system.

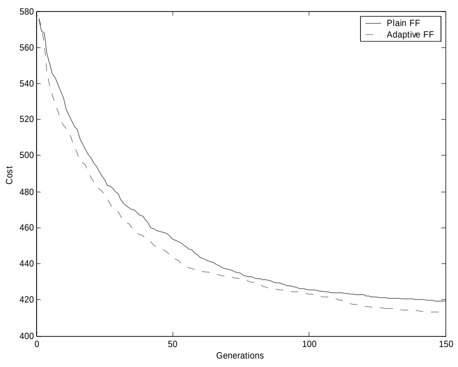

Fig. 5. Convergence behavior of the reference and the modified ES in the LIFFDFS scheme.

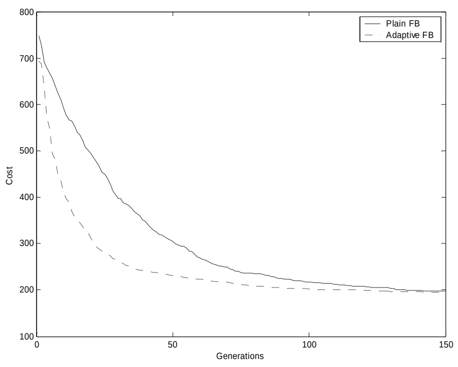

Fig. 6. Convergence behavior of the reference and the modified ES in the LIFDFS scheme.

ACKNOWLEDGMENT

The authors would like to thank Finnish IT Center for Science, CSC, for allowing us to use their computation services.

REFERENCES

[1] A. Ghosh, S. Tsutsui, & H. Tanaka, Function Optimization in Nonstationary Environment Using Steady State Genetic Algorithms with Aging of Individuals. *Proc. of the IEEE International Conference on Evolutionary Computation*, (Piscataway, NJ: IEEE Press, 1998), 666-671.

[2] X.Z. Gao, & S.J. Ovaska, Linguistic Information Feed-Forward-Based Dynamical Fuzzy Systems – Part II: Evaluation. *Proc. of the IEEE International Workshop on Soft Computing in Industrial Applications*, Binghamton, NY, 2003, 81-84.

[3] X.Z. Gao, & S.J. Ovaska, Dynamical Fuzzy Systems with Linguistic Information Feedback. *Systematic organisation of information in fuzzy systems*, P. Melo-Pinto et al., (Eds.), (Amsterdam, The Netherlands IOS Press, 2002), 179-195.

[4] A. Huber, & D.A. Mlynski, An Efficient Age-Controlled Evolution Approach Solving the Assignment problem on Analog Transistor Arrays. *Proc. of the 40th Midwest Symposium on Circuits and Systems*, Sacramento, CA, 1997, 1042-1045.

[5] M. Washita, & H. Iba, Island Model GP with Immigrants Aging and Depth-Dependent Crossover. *Proc. of the IEEE Congres on Evolutionary Computation*, Honolulu, HI, 2002, Vol. 1, 267-272.

[6] D.B. Fogel, *Evolutionary computation, toward a new philosophy of machine intelligence* (Piscataway, NJ: IEEE Press, 2000).

[7] J. Martikainen, & S.J. Ovaska, Designing Multiplicative General Parameter Filters Using Multipopulation Genetic Algorithm. *Proc. of the 6th Nordic Signal Processing Symposium*, Espoo, Finland, 2004, 25-28.

[8] L. Wang, *A course in fuzzy systems and control*. (Upper Saddle River, NJ: Prentice-Hall International, 1997).

[9] S. McClintock, T. Lunney, & A. Hashim, A Fuzzy Logic Controlled Genetic Algorithm Environment. *Proc. of the IEEE International Conference on Computational Cybernetics and Simulation*, Orlando, FL, 1997, Vol. 3, 2181-2186.

[10] Z. Bingul, A. Sekmen, & S. Zein-Sabatto, Evolutionary Approach to Multi-Objective Problems Using Adaptive Genetic Algorithms. *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, Nashville, TN, 2000, Vol. 3, 1923-1927.

[11] G.E.P Box, & G.M. Jenkins, *Time series analysis: forecasting and control*, 2nd ed, (San Francisco: Holden-Day, 1976).