Parallel Evolutionary Algorithms for Optimizing Data–based Generated Fuzzy Systems

P. Krause[1], D. Wiesmann[2], T. Slawinski[1]

[1] Chair of Electrical Control Engineering
University of Dortmund
Germany
{krause,slawinski}@esr.e-technik.uni-dortmund.de
[2] Chair of Systems Analysis
University of Dortmund
Germany
wiesmann@ls11.cs.uni-dortmund.de

Abstract. In the field of data–based fuzzy modeling, the complexity of applications and the amount of data to be processed have grown continuously. Thus, the computational effort for solving these applications has also increased drastically. In order to meet this challenge, parallel computing approaches are applied. The task here is the optimization of data–based generated fuzzy rule bases. For this kind of application the fitness evaluation of an individual is very time consuming. Here, a parallel genetic algorithm is applied to solve the optimization problem in an acceptable amount of time. Furthermore, it will be analyzed how the quality of the results changes with the use of multi–population models or neighborhood models. This will be illustrated by two example applications.

1 Introduction

In recent years, many methods for the different data–mining techniques have been developed. One field of data mining is fuzzy modeling. Fuzzy modeling is quiet popular, because it is possible to generate comprehensible models of high quality. Here, we use the Fuzzy–ROSA[1] method [7] for fuzzy modeling. It is based on testing single fuzzy rules for their capability of describing a relevant aspect of the system to model. The idea of testing only single fuzzy rules, in contrast to testing complete fuzzy rule bases, makes it possible to handle highly complex systems. On the other hand, because the final fuzzy rule base of the model is built of rules, which have been tested independently, there is still space for improvements. The optimization problem here, consists of finding the optimal selection of rules, in order to obtain a small rule base and a high quality of the model.

Here, for the rule base optimization a genetic algorithm is used. Even though, for more complex applications it can take an unacceptable amount of time for the

[1] **R**ule**O**riented **S**tatistic **A**nalysis

rule optimization, because of the time consuming fitness calculations. Sometimes the result of the optimization is not satisfying also. Therefore, we apply and adapt a parallel genetic algorithms with the aim to gain not only a speedup but also better results. Therefore different population models, like multi–population models or neighborhood models, are analyzed with respect to the speedup and the quality of the results.

First, we give a short overview over the Fuzzy–ROSA method. In the next section different models of parallel evolutionary algorithms are discussed. Finally, three different models of parallel evolutionary algorithms are applied to two fuzzy modeling problems. By an empirical comparison we try to find out the best parallel model for a rule base optimization within the Fuzzy–ROSA method.

2 The Fuzzy–ROSA Method

In this section the basic ideas of the Fuzzy–ROSA method are briefly presented and the starting point for the parallelization is described.

2.1 Concept of rule generation

The starting point of fuzzy modeling is the existence of data describing the input/output behavior of the system under consideration. The basic idea of the rule generation process is to apply a relevance test to single IF/THEN statements to assess their ability to describe a relevant aspect of the system under consideration [8, 2, 4]. This allows not only to get transparent and comprehensible rule bases, but reduces the immense problem of finding a good rule base to the much smaller problem of finding single relevant rules.

Instead of complete rules that consider every input variable in each premise, generalizing rules are used that consider only a part of the input variables in the premise. The advantage of generalizing rules is that they cover not only one but several input situations and usually, less rules are necessary. For a high number of variables, an adaptive evolutionary search concept [5, 12] is used to find the relevant fuzzy rules in the search space of IF/THEN statements. An online rule reduction removes redundant rules during the rule generation process.

In comparison to methods that directly search for an optimal rule base [1, 10, 11, 3], the computing time of this approach is also practicable for applications with more than a handful variables. Even an industrial problem with 149 input variables has been successfully solved [13]. A potential drawback is, that the search for single relevant fuzzy rules does not consider the quality of the final fuzzy rule base. Therefore, the final rule base can be optimized with respect to the quality of the fuzzy model (see next section).

2.2 Concepts for optimization

After a fuzzy rule base has been obtained there are remaining free parameters, that can be used for optimization. These include the parameters of the fuzzy system, like logical operators or defuzzification and the shape and the position of the membershipfunctions. For our application the combination of the rules seems to be most promising for a optimization. The aim is to find a rule base, which is as small as possible and models the input/output behavior of the system as good as possible. For this combinatorial problem we use a genetic algorithm approach for optimization [6].

Given a rule base of r rules, the individuals of the genetic algorithm are a binary string with r bits. Each bit represents the status of a single rule. Each rule can be included in the rule base or excluded from the rule base. As the optimization has two objectives (small and good rule base), a weighting factor w can be chosen to define the priority of the two criterions [6]. In order to calculate the fitness of one individual the number of active rules and the quality of the model are needed. Evaluating the quality of the model means to simulate the fuzzy model with all learning data points and to calculate the difference between the simulated output and the original value at the corresponding point. This is usually, as mentioned before, very time consuming. The recombination used is 4–point crossover with a recombination rate of 0.6. The mutation rate was $1/r$.

3 Parallel Evolutionary Algorithms

There are different approaches for parallel evolutionary algorithms proposed in literature [14]. These range from just using sequential algorithms in parallel to developing parallel variants based on biological paradigms. By using structured populations a parallelization can change the quality of the results. Besides thoughts about the structural approach it has to be decided whether to implement the algorithm synchronous or asynchronous. Synchronous means, that after each generation the parallel processes are synchronized. In contrast, if an asynchronous algorithm is used, a modification of the standard selection mechanism is needed, because there are no fixed generations anymore. In the following sections, we present the methods used in our work. A formal description of population structures can be found in [14].

3.1 Master–Slave Model

The easiest way to parallelize an evolutionary algorithm is to calculate the fitness of the individuals in parallel. The fitness calculation of the individuals are independent of each other and the evaluation often needs the main part of processing time in one generation. This is especially true for our application as described in 2.2. Thus, a synchronous implementation of this approach differs not from a sequential approach regarding functionality. A nearly linear speedup can be expected, if the time needed for one fitness calculation is large compared to the time needed for communication.

3.2 Pollination Model

The pollination model is a multi–population model. A population of λ individuals is divided into p subpopulations Q_i. Each subpopulation exchanges certain individuals with their neighbor subpopulations (Fig. 1). Thus, the parallelization is exploited on the level of populations. This concept introduces additional parameters for the evolutionary algorithm:

– **Isolation Time f:** The frequency of the exchanges between the subpopulations
– **Exchange Volume:** The number of individuals exchanged
– **Type of Individuals:** Which individuals are exchanged (e.g. best individual, random individual)
– **Integration of Individuals:** Which individuals are replaced (e.g. replacement of the worst)

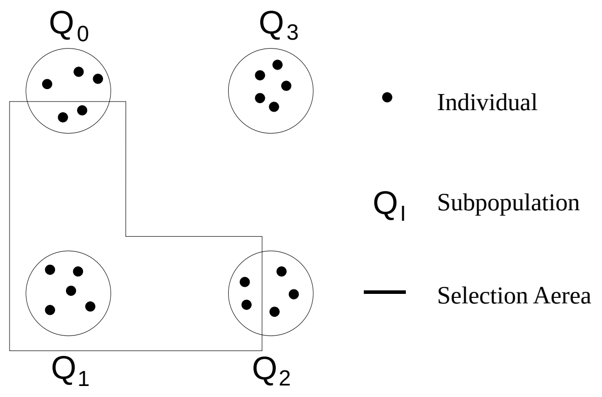


Fig. 1. Example of a pollination model.

Because of the low communication needed here, a nearly linear speedup can be expected, too. Considering the quality of the result, we expect an increase in the quality, because of the greater diversity maintained by the subpopulations.

3.3 Neighborhood Model

In the neighborhood model the parallelization is performed on the level of individuals. Between the individuals a neighborhood relation is defined, which determines which individuals can recombine with each other. The selection is applied locally within these neighborhoods also. It is guaranteed that the information of each individual can be spread to every point in the population through overlapping neighborhoods (Fig. 2). Here, the following parameters have to be chosen:

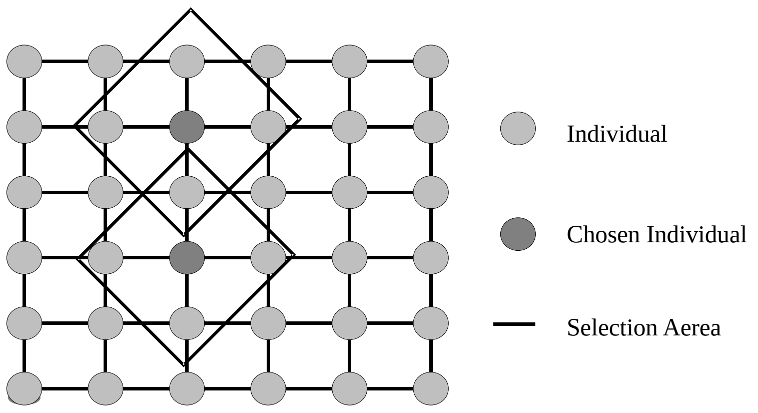


Fig. 2. Example of a neighborhood model.

- **Size of Grid:** The size of the grid strongly depends on the available parallel computer
- **Number of Individuals per Process:** Because the number of processors is strongly limited, several neighboring individuals can be mapped to a single processor. This increases the load of each processor. On the other hand it increases the possible number of individuals and reduces the overhead in communication.

This concept contains many aspects of massive parallelism. There is no global control instance, the processes are independent of each other and the communication is locally limited.

4 Application of Parallel Evolutionary Algorithms

We have chosen two examples, which differ greatly in size, to see how the parallel evolutionary concepts work.

- **HEAT:** The first example is the modeling of an heat exchanger in an industrial process. Therefore, the task is to model the output variable on the basis of three process variables. There are 1126 data points for training. Four, 6 and 9 fuzzy sets are defined for the input variables and 9 fuzzy sets are defined for the output variable. The rule generation process results in 55 rules. Thus, the search space for the optimization is $2^{55} = 3\ .6 \cdot 10^{16}$
- **SAT:** As the second example, the benchmark problem 'landsat satellite image'[2] [9] is chosen. The original landsat data is from the NASA and contains the intensities of four spectral bands for 82 x 100 pixels, each pixel covering an area on the ground of approximately 80*80 meters. Each pixel belongs

[2] ftp: ics.uci.edu in /pub/machine-learning-databases

to one of seven classes: red soil, cotton crop, gray soil, damp gray soil, soil with vegetation stubble, very damp gray soil, mixture. The task is to classify the soil of each pixel on basis of the intensities of the four spectral bands of the pixel and its 8 neighbors. Thus, it is a classification problem with 36 continuous input variables and six output classes (the class 'mixture' is not considered). There are 4435 data points in the training set and 2000 data points in the test set. Nine fuzzy sets are defined for each input variable. The maximum number of input variables considered in a premise is fixed at six. The rule generation process results in 27701 relevant generalizing fuzzy rules that must be reduced in the second step. First the data-based conflict reduction method is applied, which results in a reduced number of rules of 2403. Thus, the search space for the optimization is $2^{2403} = 2\,.96 \cdot 10^{722}$

In the following, the pollination model and the neighborhood model are compared to the master–slave model, which correspond to a serial genetic algorithm. Differing from standard genetic algorithms, we used a $(\mu + \lambda)$ evolutionary algorithm approach as the selection scheme. The sizes of the populations used are given in the graphics. The parallel computer used, is a SGI Origin 2000 with 16 processors. For better comparison the fitness is shown against the number of fitness calculations and not against the number of generations. The other important quantity, besides the quality of the results, is the efficiency as a measure for the speed gain. The efficiency is calculated by

$$\text{Efficiency} = \frac{\text{SpeedUp}}{\#\text{processors}} \tag{1}$$

with

$$\text{SpeedUp} = \frac{\#\text{fitness calculations}_{parallel} \cdot \text{time}_{serial}}{\text{time}_{parallel} \cdot \#\text{fitness calculations}_{serial}} \quad . \tag{2}$$

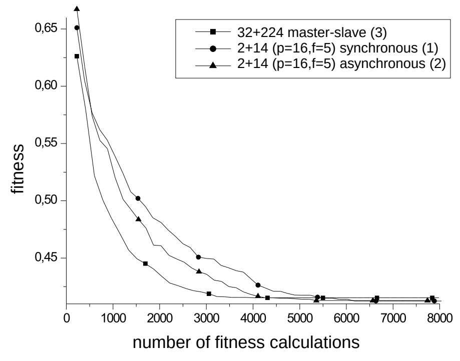
4.1 Pollination Model

First, we analyzed the behavior of the pollination model. Therefore, different numbers p of subpopulations and different isolation times f were chosen. The best individuals of the subpopulations are exchanged and the worst individuals are replaced in the corresponding subpopulation. Fig. 3(a)[3] shows the results for different numbers of subpopulations for HEAT. Here, five experiments have been carried out for each parameter set and the average of the results has been taken. We see, that the pollination model, synchronous or asynchronous, gives better results, but converges slower than the master–slave model. The synchronous algorithm has a slight advantage over the asynchronous algorithm referring to the final results. Fig. 3(b) shows, that the efficiency of the pollination model is higher than the efficiency of the master–slave model and thus is faster.

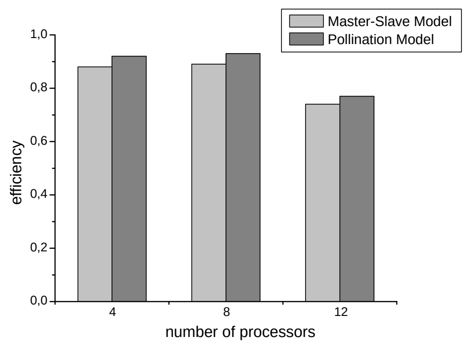
If the influence of the isolation time is analyzed, we can see, that higher isolation times lead to better results. On the other hand we can see, that with less

[3] In the following, the number behind the graph in the legend indicates the ranking of the result after the last fitness calculation.

subpopulations, the convergence behavior of the pollination model improves, but does not reach the convergence behavior of the master slave model. Overall, the



(a) Simualtion results



(b) Efficiency

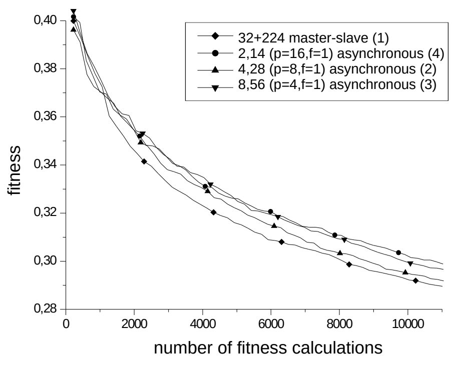
Fig. 3. Results for the pollination model for HEAT

usage of the pollination model supplies better results in the end, because of the higher diversity due to the subpopulations. On the other hand, the convergence is slower than the serial algorithm in terms of fitness calculations. On the other hand, through the enhanced efficiency, it needs less time to reach the final results.

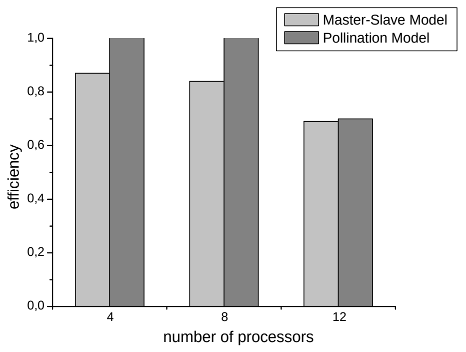
The results for SAT are shown in Fig. 4. Looking at Fig. 4(a), we find, that the pollination model has no advantage over the serial algorithm with respect to the quality. On the other hand, the efficiency of the pollination model is higher than the efficiency of the master slave model (Fig. 4(b)). Due to the complexity of SAT, the experiments are very time consuming. Thus, as we can see, the fitness has not settled after 10000 fitness calculations. It is questionable if the final results of the pollination models are better than the results of the master–slave model.

4.2 Neighborhood Model

For the usage of the neighborhood model, we analyzed the effects of different numbers of individuals and processors. Here, a comma selection is used and the neighborhood consists of the direct neighbors only. Fig. 5(a) shows the results for HEAT. The best result is achieved for 36 individuals on 9 processors. The efficiency for this configuration is good in comparison to the other configurations here (Fig. 5(b)). But, comparing it to the efficiencies in 4.1, we can see, that this model is rather slow. The reason for the low efficiency is the high demand on communication. When using the neighborhood model for SAT the same problems occur as described in 4.1. Again the area of convergence is not

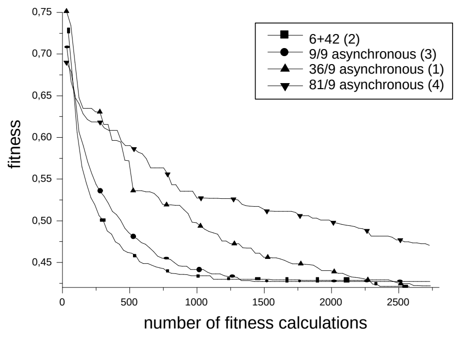


(a) Simualtion results

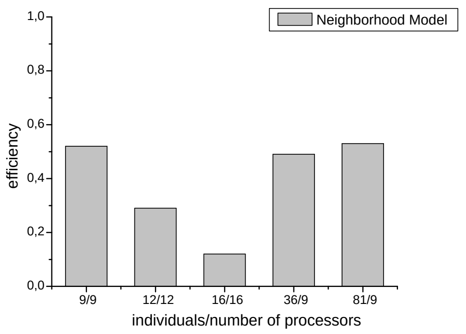


(b) Efficiency

Fig. 4. Results for the pollination model for SAT



(a) Simualtion results

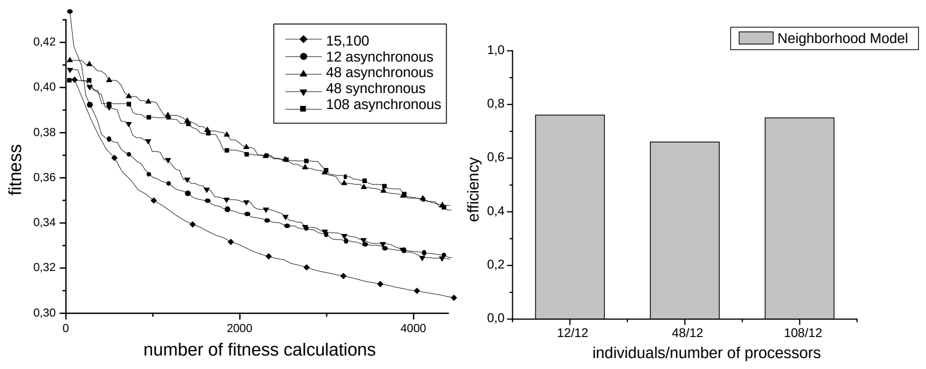


(b) Efficiency

Fig. 5. Results for the neighborhood model for HEAT

reached. However, we can see in Fig. 6(a) that the master–slave model converges faster than the neighborhood models. The efficiency is higher than the efficiency in HEAT, because the ratio between the time for one fitness calculation to the communication time is higher. But, the efficiency is still lower than the efficiency of the master–slave model or the pollination model.

Comparing the pollination model and the neighborhood model, we conclude, that the pollination model is more advantageous in our case, concerning the reached fitness and the efficiency.



(a) Simualtion results (b) Efficiency

Fig. 6. Results for the neighborhood model for SAT

5 Conclusions

The time needed for all fitness calculations necessary for the optimization of a fuzzy rule base with a genetic algorithm reaches easily dimension, where the use of a parallel computer can be justified. The use of a parallel computer does not only speedup the optimization process but gives also the opportunity to use structured populations. These can change the quality of the results. To test different approaches and find satisfying parameter settings of the parallel evolutionary algorithms, we have chosen two examples (HEAT and SAT) of different complexity. Besides the master–slave model, the pollination model and the neighborhood model have been applied and adapted to the examples. For the HEAT example, it has been shown, that the pollination model has advantages in the quality and the efficiency over the master–slave model. The neighborhood model shows some disadvantages in the results. The SAT examples shows similar results. In the future, further examination concerning the choice of the parameters of the parallel evolutionary algorithms have to be done. The SAT example has shown, that here more simulations are necessary to test the long term behavior of the parallel evolutionary algorithms for complex applications.

Acknowledgment

This research was sponsored by the Deutsche Forschungsgemeinschaft (DFG), as part of the Collaborative Research Center 'Computational Intelligence' (531) of the University of Dortmund[4].

[4] Part of the references can be downloaded from *http://esr.e-technik.uni-dortmund.de/winrosa/winrosa.htm.*

References

1. C.-C. Hsu, S.-I. Yamada, H. Fujikawa, and K. Shida. A fuzzy self–tuning parallel genetic algorithm for optimization. *Computers and Industrial Engineering*, 30:883–893, 1996.
2. H. Jessen and T. Slawinski. Test– and rating strategies for data–based rule generation. In *Reihe Computational Intelligence.* CI–39/98, Sonderforschungsbereich 531, Universität Dortmund, 1998.
3. Y. Jin, W. von Seelen, and B. Sendhoff. An approach to rule–based knowledge extraction. In *Proceedings of the Seventh IEEE International Conference on Fuzzy Systems, FUZZ–IEEE '98*, volume 2, pages 1188–1193, Anchorage, Alaska, USA, 1998.
4. H. Kiendl and M. Krabs. Ein Verfahren zur Generierung regelbasierter Modelle für dynamische Systeme. *at – Automatisierungstechnik*, 37(11):423–430, 1989.
5. A. Krone and H. Kiendl. Evolutionary concept for generating relevant fuzzy rules from data. *International Journal of Knowledge–based Intelligent Engineering Systems*, 1(4):207–213, 1997.
6. A. Krone, P. Krause, and T. Slawinski. A new rule reduction method for finding interpretable and small rule bases in high dimensional search spaces. In *Proceedings of IEEE International Conference on Fuzzy Systems, FUZZ–IEEE '00*, page in print, 2000.
7. A. Krone and U. Schwane. Generating fuzzy rules from contradictory data of different control strategies and control performances. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, FUZZ–IEEE '96*, volume 1, pages 492–497, New Orleans, USA, 1996.
8. A. Krone and H. Taeger. Relevance test for fuzzy rules. In *Reihe Computational Intelligence.* CI–40/98, Sonderforschungsbereich 531, Universität Dortmund, 1998.
9. D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine learning, neural and statistical classification.* Ellis Horwood, Hemel Hempstead, Großbitanien, 1994.
10. O. Nelles, M. Fischer, and B. Müller. Fuzzy rule extraction by a genetic algorithm and constrained nonlinear optimization of membership functions. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, FUZZ–IEEE '96*, volume 1, pages 213–219, New Orleans, USA, 1996.
11. D. Popovic and N. Xiong. Design of flexible structured fuzzy controllers using genetic algorithms. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, FUZZ–IEEE '96*, volume 3, pages 1682–1686, New Orleans, USA, 1996.
12. T. Slawinski, A. Krone, U. Hammel, D. Wiesmann, and P. Krause. A hybrid evolutionary search concept for data-based generation of relevant fuzzy rules in high dimensional spaces. In *Proceedings of IEEE International Conference on Fuzzy Systems, FUZZ–IEEE '99*, volume 3, pages 1432–1437, Seoul, Korea, 1999.
13. T. Slawinski, J. Praczyk, U. Schwane, A. Krone, and H. Kiendl. Data–based generation of fuzzy–rules for classification, prediction and control with the fuzzy–ROSA method. In *European Control Congress, ECC '99*, Karlsruhe, 1999.
14. Joachim Sprave. A unified model of non-panmictic population structures in evolutionary algorithms. In P. J. Angeline and V. W. Porto, editors, *Proc. 1999 Congress on Evolutionary Computation (CEC'99)*, volume 2, pages 1384–1391, Washington D.C., July 6–9, 1999. IEEE Press, Piscataway NJ.