Adaptive Scheduling of Optimization Algorithms in the Construction of Interpolative Fuzzy Systems

Krisztián Balázs
Department of Telecommunications
and Media Informatics
Budapest University of
Technology and Economics
Magyar tudósok körútja 2.
Budapest, H-1117, Hungary
Email: balazs@tmit.bme.hu

László T. Kóczy
Department of Automation
Széchenyi István University
Egyetem tér 1.
Győr, H-9026, Hungary
Email:koczy@sze.hu
Department of Telecommunications
and Media Informatics
Budapest University of
Technology and Economics
Magyar tudósok körútja 2.
Budapest, H-1117, Hungary
Email: koczy@tmit.bme.hu

***Abstract*—This paper presents an adaptive scheduling approach applied for constructing interpolative fuzzy rule based systems. This is continuation of our preceding work, where the same approach was used for dense fuzzy rule bases.**

During the optimization process different optimization algorithms are scheduled according to their respective local efficiency, i.e. according to their convergence speed in various phases of the machine learning process. The scheduled optimization techniques are evolutionary algorithms that have shown efficiency in the construction of interpolative fuzzy rule based systems.

Simulations are carried out on standard benchmark sets in order to evaluate the established system and to compare it to fuzzy systems built up by deploying the same optimization techniques without the scheduling approach.

***Index Terms*—Adaptive scheduling of optimization algorithms, Fuzzy rule based knowledge extraction, Interpolative fuzzy systems**

I. Introduction

Knowledge extraction, or machine learning [1], aims at gathering information from data sets by finding highly descriptive hidden structures in them in order to model the whole system or phenomena, where the particular data sets originate from. The resulting model can be then used for predictions, evaluations, diagnoses, decision making and in a number of other ways based on the nature of the subject of the model.

When performing knowledge extraction the resulting model is usually selected from a family of possible models by finding (at least in theory) the optimal one from a pool of candidates. Actually, this means that the parameters of a parameterizable modeling architecture are tried to be adjusted to the most suitable values, in which case the architecture can imitate the characteristics of the involved data set most properly. This way the learning problem can be formulated as an optimization task, which can be solved by optimization methods.

There are a plethora of parameterizable modeling architectures and it is a very difficult and ambiguous task to find the one possessing the best properties from the point of view of accuracy and complexity. In order to illustrate this, two reasons are mentioned. The first one is that practically applicable results do not exist dealing with the modeling power of the majority of modeling approaches. Although, a large part of modeling architecture types is proved to possess the universal approximator property, this is rather theoretical and not practically applicable [2], because real universal approximators have unbounded demands in both space and time complexity. The second reason is that most of the efficient modeling architectures are difficult enough to lead to an analytically unpredictable learning process, including the quality of the resulting model. Therefore, in a particular problem field the 'best' modeling architecture can only be selected by a lot of experience.

On the other hand, if the need arises for an architecture, using which the extracted knowledge is not only applicable in further operations, but is also conveyable to humans, i.e. it is understandable, then a modeling technique having outstanding inherent interpretability properties is searched for. In this case the most appropriate choice is definitely to apply fuzzy rule based architectures for modeling, because besides being universal approximators and being capable of solving different kinds of machine learning problems, they have rather unique inherent interpretability properties possessed by no other modeling architectures [3].

Among fuzzy systems there are classes applying different complexity reduction techniques. A way of complexity reduction is to decrease the number of rules and to use interpolation [4], [5] between them, whereas another way is to divide the input space into several parts and to construct different sub-rule bases for each part (moreover possibly do this recursively) resulting in a hierarchical rule base [6], [7]. Additionally, the two complexity reduction approaches may be combined, thus hierarchical-interpolative fuzzy systems [8], [9] may also be

used in order to achieve higher efficiency.

As it was mentioned above, a machine learning process is actually an optimization process, where the aim is to find the optimal parameters of the modeling architecture. Since this is a complex optimization task, only global optimization techniques are viable. Based on the literature evolutionary optimization methods, especially, Bacterial Evolutionary Algorithm and its memetic variants are such very efficient techniques (cf. [10], [11], [12] and [13]).

Our recent work [14] proposed an approach for combining algorithms by adaptively scheduling them during an optimization process in order to improve the convergence to the optimum. This approach can also be applied for the construction of fuzzy systems.

After the above discussions the aim of this paper may be formulated. Our goal is to establish a promising knowledge extracting system being able to build accurate knowledge bases as fast as possible with low complexity and outstanding inherent interpretability properties. For this purpose interpolative fuzzy systems are used together with the adaptive scheduling of the mentioned bacterial algorithms.

Despite the importance of the interpretability property, it will not be exploited in the present paper. Its investigation will be the aim of future works.

Although, in fuzzy systems the interpretability of the extracted information can be ensured by using appropriate methods (which are not considered in this paper, but can be found e.g. in [15] and [16]), like in case of the majority of modeling approaches, fuzzy architectures are also too difficult to determine analytically the speed of the learning process and the accuracy of the resulting knowledge base. Hence the established systems will be evaluated based on the results of simulation runs carried out on different machine learning problems.

The next section gives a theoretical overview of the above mentioned techniques and approaches. Section III describes the establishment of the investigated fuzzy rule based machine learning systems. In Section IV the constructed systems are evaluated and compared to those not applying the adaptive scheduling approach. Finally, Section V summarizes the paper, draws some conclusions and appoints aims for future works.

II. Theoretical background

In this section the theoretical background of the methods merged during our work in the construction of sparse rule bases will be described shortly.

First the schema of the established knowledge extraction systems will be explained followed by a brief description of its key components, namely, interpolative fuzzy systems as knowledge representing architectures, evolutionary optimization methods and the adaptive scheduling approach.

A. Supervised machine learning

Supervised machine learning is a branch of knowledge extraction approaches. Given a system (or phenomenon) characterized by a data set (samples) and a modeling architecture with adjustable parameters. The samples are formed by input-output $((x_i, d_i)_{i=1}^n)$ pairs corresponding to the system.

The task of supervised machine learning is to adjust the parameters of the modeling architecture to make it similar in 'behavior' to the system. The 'behavior' of the model is characterized by input-output pairs, where the inputs $(\{x_i\}_{i=1}^n)$ are the same inputs as the system has, respectively, and the outputs $(\{y_i\}_{i=1}^n)$ are the corresponding responses of the modeling architecture. The degree of similarity between the behaviors is measured by a predefined error function being a function of the parameters of the model. For example, *Mean Squared Error* is a commonly applied error definition:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - d_i)^2 \tag{1}$$

The higher the degree of similarity, the lower the value of the error function. Hence the task of supervised machine learning is actually to find values to the parameters, in which case the error is minimal. This way supervised machine learning problems can be reduced to optimization problems.

Most often, the samples are divided at least into two parts. If they are separated into two sets, then the first set (training samples) is used during the learning process and involved in the adjustment of the parameters of the model through the error function. The second set (test samples) is applied after the learning in the evaluation of the accuracy of the obtained model.

If the original data set is divided into more than two parts, then additional tests can be performed during the learning process e.g. in order to avoid overfitting [1].

B. Interpolative fuzzy systems

The concept of traditional fuzzy inference is well-known (see e.g. [17]). The input of the system (observation) is compared with the antecedent parts of the rules contained by the rule base, and the degree of matching between the observation and the antecedent part of each fuzzy rule is calculated. This value is positive for a rule if and only if the antecedent sets of the rule overlap with the observation in each input dimension. Then, based on these values the fuzzy inference engine computes a conclusion according to the respective consequent parts of the rules. Finally, after defuzzification a crisp conclusion is produced by the system.

Due to this mechanism, if an observation can be given to the system, for which in the rule base there are no rules matching with positive degree, then the inference engine is unable to compute any conclusions when that particular observation occurs. As a result, in a properly working fuzzy system the union of the antecedent parts of the rules must cover the whole input space. Such rule bases are called *dense*.

Moreover, it is a usual requirement that every element of the input space should be contained by at least one rule antecedent with at least a predefined $\alpha > 0$ membership value.

Although, in the above inference schema the application of dense rule bases is necessary for a properly working fuzzy inference system, denseness leads to unfavorable complexity.

Namely, if the input space is k dimensional and there are maximum T antecedent sets along each dimension, then the rule base must contain $\mathcal{O}(T^k)$ rules. This results in the exponential growth of the rule base size in terms of the number of input dimensions, strongly affecting not only the space complexity, but (which may be more important) also the computational demand of the system.

This problem motivated the invention of complexity reduction approaches for fuzzy inference systems, such as interpolative [4], [5] and hierarchical techniques [6], [7] as well as combined hierarchical-interpolative methods [8], [9].

In case of interpolative systems the described inference schema is modified [4], [5]. During the establishment of the conclusion the consequent parts of the rules are combined based on the distance of the respective antecedent parts from the observation. The closer a rule is, the bigger influence it has in the shaping of the conclusion.

When interpolation methods are used, gaps are allowed between the antecedents of the rules within the rule base, because fuzzy inference can be performed even if there are no rule antecedents overlapping with the given observation. That is, the rule base can be *sparse*.

This way T can be decreased to $T' < T$ in the above complexity, i.e. the complexity can be reduced to $\mathcal{O}(T'^k) < \mathcal{O}(T^k)$.

C. Bacterial Evolutionary Algorithm and its memetic variant

Evolutionary algorithms (EAs) are iterative stochastic global optimization methods imitating the evolutionary processes observed in the nature. For the given optimization task they maintain a group of candidate solutions, called *individuals* forming the *population*, which are continuously modified iteration-by-iteration (or *generation-by-generation*) using the *evolutionary operators*. The candidate solutions are represented in the form of numerical vectors, whose values are changed by the evolutionary operators.

The individuals compete each other in a similar way as species and individuals do in the nature. Their success is determined by the *fitness function* being a monotonic transformation of the objective function of the optimization problem. Thus, the higher the fitness of the particular individual, the closer the represented candidate solution to the optimal one.

The Bacterial Evolutionary Algorithm (BEA) [18] is a very simple, but rather efficient evolutionary technique. Based on a plethora of experimental comparisons (cf. e.g. [10], [11], [12] and [13]) they clearly outperform the well-known and widely-applied famous Genetic Algorithm [19].

After initialization within the main iteration loop there are two evolutionary operators: bacterial mutation and gene transfer.

Although, evolutionary algorithms explore the whole search space defined by the given optimization task, due to the mechanism of their usual operators they do not explore it in details. In contrast, *local search methods* are optimization techniques exploring only a local neighborhood of the initial points within the search space, however they explore it rather comprehensively compared to EAs. In order to unite their advantages, they can be united, if e.g. local search steps are executed as evolutionary operators within the main iteration loop of an evolutionary technique. These combinations are called *memetic algorithms* [20].

For example, Bacterial Memetic Algorithm (BMA) [10] can be constructed if either Steepest Descent [21] or Levenberg-Marquardt [22], [23] iterations are executed among the steps of BEA. In this paper BMA refers to the technique involving Levenberg-Marquardt gradient steps.

D. Adaptive scheduling of optimization algorithms

The problem of adaptive scheduling of optimization algorithms was introduced recently in [14]. Its motivation and goal are the following.

As a rule of thumb it can be said that for simpler optimization tasks simpler algorithms, for more difficult ones more complex optimization techniques are suitable. An illustrative, exaggerated example can be the minimization of a simple one dimensional quadratic function and of a thousand dimensional stochastically strongly perturbed multimodal discontinuous surface that does not have a closed analytic form even without the stochastic perturbation. Whereas in the first case the exact minimum can be found analytically in a few steps, well-known from calculus, in the second case the exact minimum is certainly unreachable and efficient global optimization techniques like EAs are only viable together with long execution times.

However, a similar dichotomy can be observed when a sufficiently difficult optimization problem is solved. Namely, at the beginning of the optimization process simpler methods converge faster than more compound ones, because the former techniques have lower computational demands and since the initial points within the search space usually have low quality, it is easy to reach better and better candidate solutions iteration-by-iteration. However, in longer terms the higher computational demand of more compound algorithms becomes a small drawback compared to the advantage coming from the higher improvement power of more complex methods. Thus, it heuristically follows that for more difficult problems in longer terms the best more complex optimization approaches can outperform the best simpler ones. Indeed, there are many simulation results in the literature confirming this heuristic (e.g. [11], [12] and [13]).

For example, Fig. 1 shows the *characteristic*s, i.e. the convergence speeds in terms of fitness level, of BEA and BMA during a fuzzy rule based learning process [14].

After this observation, the idea to use a simpler method arises in the early parts of an optimization process and when a more complex technique becomes more efficient, then switch between the algorithms and continue with the execution of the more compound one. Moreover, this idea can be generalized to the approach to always (i.e. in each iteration during the optimization) apply the currently most efficient technique, that is, to adaptively schedule the optimization algorithms during the whole optimization process.

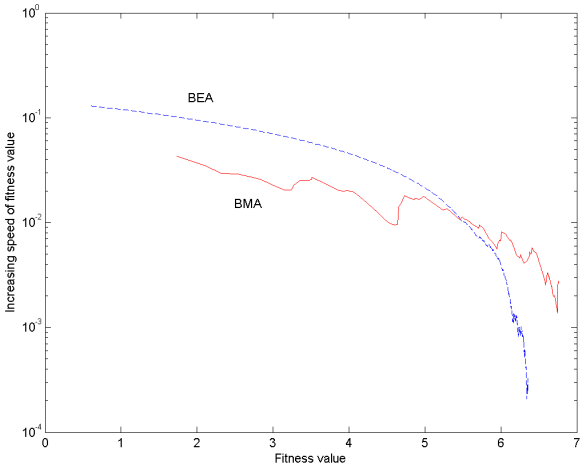

Fig. 1. Characteristics of BEA and BMA during an example optimization process.

For this scheduling problem two scheduling strategies (*schedulers*) were defined [14], and very recently an improvement was proposed to the second one [24].

1) Greedy Scheduler: Greedy scheduler executes all the algorithms simultaneously, and after each iteration (or time slot) the currently best candidate solution is selected according to its objective function value. In case of population based techniques (like BEA or BMA) the best population is selected based on some quality measure, e.g. the fitness value of the best individual. In the subsequent iteration (or time slot) every algorithm is initialized to the selected candidate solution or population and executed simultaneously. Then, another comparison is performed, and so forth.

The seemingly hopeless drawback of this simple scheduler is its huge overhead originating form the parallel execution of all algorithms during the whole optimization process.

2) Fast Greedy Scheduler: Sometimes the optimization problem has favorable properties and particular optimization algorithms can perform best during long periods. In this case the number of switches between the algorithms in an optimal schedule may be low.

In order to exploit this advantage of such problems, another version of the above discussed Greedy scheduler was proposed. If the scheduling problem has the mentioned favorable property, this scheduling method is faster, than the previous one.

The Fast Greedy Scheduler does not compare the optimization algorithms in each step, i.e. it does not execute them simultaneously all the time, but only after a predefined '*blind running*' time, while it applies only the last locally best algorithm.

This way the mentioned drawback of the previous scheduler can be eased.

3) A recently proposed improvement for Fast Greedy Scheduler: If it can be assumed that the characteristics, i.e. the convergence speeds in terms of fitness level, are monotonic decreasing functions, then the following straightforward improvement can be applied to Fast Greedy Scheduler, resulting in the approach Monotonic Fast Greedy Scheduler (MFGS).

After each blind running phase the convergence speed of the active algorithm is compared to the last calculated convergence speeds of the inactive ones, which are the values calculated at the end of the last performed comparing phase. If the convergence speed of the active method is still greater than or equal to all the other values, then at the current fitness level the active algorithm must still be the most efficient one due to the monotonicity of the characteristics assumed. In this case the subsequent comparing phase is postponed and the blind running continues. During the remaining part of this lengthened phase the convergence speed is continuously compared to the last calculated convergence speeds of the inactive methods. When the efficiency of the active algorithm decreases below the highest last measured efficiency level of the inactive methods the postponed comparing phase takes place, and then the algorithms are executed simultaneously for a short period.

With this improvement MFGS clearly reduces the comparing overhead of the schedulers further.

III. Establishing interpolative fuzzy systems involving the scheduling approach

This section first explains the application of the described bacterial optimization techniques during the construction of fuzzy knowledge bases within a supervised machine learning process. Then, the recently proposed scheduler is introduced into the process.

A. Applying the bacterial techniques in interpolative systems

As it was explained in the previous section BEA represents the candidate solutions of an optimization problem in the form of numerical vectors. In case of machine learning tasks the components of these vectors correspond to the parameters of the modeling architecture. In fuzzy systems these parameters characterize the rule bases. The rule bases always contain trapezoidal membership functions in every rule in the present work. The number of fuzzy rules in the knowledge bases is predefined. Therefore, it is obvious to characterize the rule bases by the breakpoints (vertices) of the trapezoids, i.e. to assign these points to the components of the vectors represented by the individuals of BEA [10]. Thus, in the numerical vectors all the breakpoints of every trapezoid from each fuzzy rule is simply collected.

B. Adaptively scheduling the bacterial methods

After the two bacterial techniques are adapted, it is straightforward to introduce MFGS into the optimization process. Since the scheduler is a meta-optimizer, it has no direct connection to the underlying fuzzy system. MFGS controls from the top; it only monitors the bacterial methods, compares them and switches between them.

IV. Experimental analysis

Since the established systems are too difficult to determine analytically their main capabilities, namely the speed of the learning process they perform and the accuracy of the resulting knowledge base, they were evaluated based on the results of simulation runs discussed in this section.

After defining the involved machine learning problems and enumerating the parameters of the optimization processes, the results of the experiments and their explanation will be presented.

A. Involved data sets

The following two machine learning data sets were involved in the experiments from the well-know KEEL machine learning data set repository [25].

1) Friedman benchmark function: The five (input) dimensional benchmark dataset proposed by Friedman in 1991 forms a synthetic learning problem. Each sample of the set was generated using the following method.

Five values $(x_1, x_2, \dots x_5)$ were generated independently according to the uniform distribution over $[0, 1]$. Then, the output (target value) was defined applying the equation:

$$y = 10 \sin(\pi) x_1 x_2 + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e, \quad (2)$$

where $e \sim N(0, 1)$ is a standard Gaussian random noise.

The data set contains 1200 instances, from which 200 are used as training samples and 1000 as test samples in the simulations.

2) Treasury data set: The fifteen dimensional Treasury data set (Treasury) aims at predicting the so-called one month certificate of deposit rates based on fifteen economic features.

There are 1049 instances in the data set (corresponding to the weeks from April 1, 1980 through April 2, 2000), from which 349 are used as training samples and 700 as test samples in the simulations.

B. Technical details of the optimization processes

During the experimental analysis the Stabilized KH interpolation technique was used [26].

In the simulations the parameters had the same values as in our previous works (see [12] and [13]), because before carrying out simulations in those investigations after a number of test runs these values seemed to be the most suitable.

The number of individuals in a generation was 8 in both BEA and BMA algorithms. The maximum number of rules in the rule base was 4. The number of clones was 5 and 4 gene transfers were carried out in each generation. In the memetic technique the local search techniques executed 4 iterations in each generation.

MFGS compared the efficiency of the different algorithms during 125 seconds long comparing phases in order to have a meaningful comparison, since the fitness values may not increase during a long period. The length of the blind running phases were set to 500 seconds.

The optimization processes stopped after reaching the time limit, which was 2000 seconds uniformly.

At the end of each optimization processes the accuracy of the resulting rule bases was measured using the Mean Squared Error (MSE).

The number of executed generations during the learning process was also observed.

In case of all algorithms for each parameterization 8 runs were carried out for every learning problem, then the mean of the obtained values were taken.

During the runs the fitness values of the best individuals were monitored in terms of time. These fitness values were calculated based on the MSE values measured on the training samples (in contrast with the final errors, which were measured on the test samples) as follows:

$$F = \frac{10}{\mathrm{MSE} + 1} = \frac{10m}{\sum_{i=1}^{m} (d_i - y_i)^2 + m}. \quad (3)$$

The purpose of this definition is to obtain a strictly monotonic decreasing fitness function taking its values from the interval $[0, 10]$, which differentiate candidate solutions giving lower error values stronger.

The means of the fitness values of the best individuals during the runs were presented in figures (Fig. 2 and Fig. 3) to get a better overview. The horizontal axes show the elapsed computation time in seconds, whereas the vertical axes show the fitness values of the best individuals at the current time.

In the figures the dashed lines show the results of BEA, the dotted lines present the performance of BMA, whereas the solid lines show the fitness values given by the scheduling approach.

The final error values of the simulation runs are presented in tables (Table I and Table II). In the tables "MSE mean" and "MSE std." denote the mean and standard deviation of MSE values, respectively, additionally, "No. of gen. mean" and "No. of gen. std." denote the mean and the standard deviation of the number of executed generations. The best error values are highlighted.

C. Simulation results

The results for the learning problems are shown in Fig. 2 and Fig. 3 as well as in Table I and Table II.

Observing the results given by the different interpolative systems, perhaps the most obvious fact is that MFGS was outperformed in the five dimensional case, however, for the fifteen dimensional problem MFGS was the most efficient one. The explanation of these results is the following.

In case of the simpler problem it is not worth to use the scheduling technique, because its overhead deteriorates the advantage gained by the possibility of adaptively switching to the currently best performing optimization algorithm.

However, in case of the more difficult problem the advantage of adaptively switching is higher than the disadvantage of the overhead originating from the comparing requirements, and thus the interpolative system involving the scheduling approach outperforms the other ones.

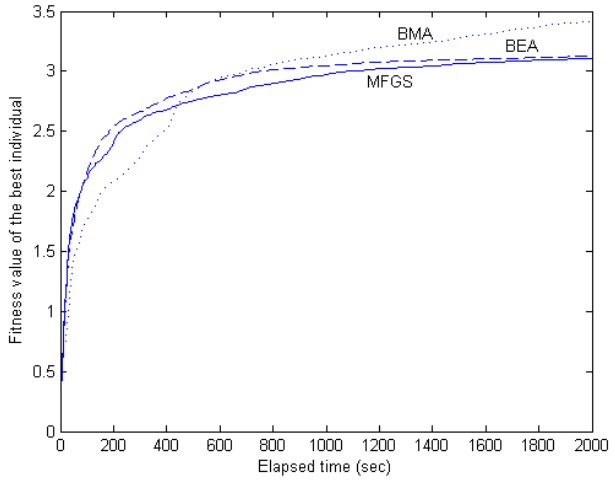

Fig. 2. Fitness values for the Friedman benchmark function (5 dimensions).

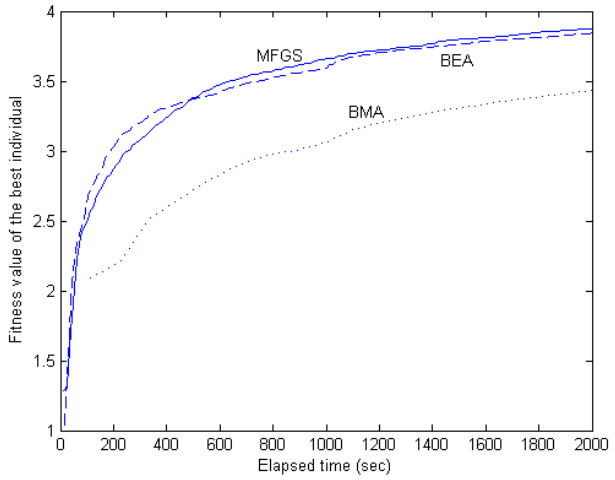

Fig. 3. Fitness values for the Treasury data set (15 dimensions).

Looking at the results given to the Treasury data set, one might have a presumption that BMA was overwhelmed in contrast with the conclusion of previous comparative works (cf. [12] and [13]). However, after a careful study it can be noticed that at the end of the optimization process the fitness curve of BMA is the steepest foreshadowing an intersection with the curve of BEA. Indeed, in previous works BMA showed a "slowly but surely" behavior, and if BMA had enough time for running, it outperformed any other techniques.

One more interesting property can be clearly seen from Table II. The number of generations executed by MFGS is almost the arithmetic mean of the generation numbers of the other two methods. Its reason is that after switching to BMA the number of generations executed in a time unit drops. (This phenomenon is not so obvious from Table I, it may be caused by late switching.)

Comparing the fitness values obtained in case of the fifteen dimensional problem to previous results published in our preceding paper [24], where Mamdani-inference was applied instead of rule interpolation, it can be observed that the present values lag behind the values given by the Mamdani-inference based system. The explanation of this is the fact that although, the interpolative system has better (lower) computational resource consumption, information is always lost in sparse models.

TABLE I
FINAL ERROR VALUES FOR THE FRIEDMAN BENCHMARK FUNCTION.

	BEA	BMA	MFGS
MSE mean	3.8313	**2.9879**	3.5871
MSE std.	0.6683	0.2706	0.5131
No. of gen. mean	1146.2	178.0	905.8
No. of gen. std.	34.1423	5.1478	25.4892

TABLE II
FINAL ERROR VALUES FOR THE TREASURY DATA SET.

	BEA	BMA	MFGS
MSE mean	1.6307	1.9160	**1.5813**
MSE std.	0.2154	0.2112	0.2234
No. of gen. mean	213.375	16.500	127.750
No. of gen. std.	27.2393	1.1952	38.3955

V. CONCLUSIONS

The goal of this paper was to apply the recently proposed Adaptive Scheduling of Optimization Algorithms approach in interpolative fuzzy rule based knowledge extraction processes in order to solve supervised machine learning problems. The scheduling was performed between Bacterial Evolutionary Algorithm and its memetic variant, Bacterial Memetic Algorihm, since according to previous studies (e.g. [12] and [13]), these are efficient techniques for constructing interpolative fuzzy systems.

Simulation runs were carried out on two data sets in order to compare the performance achieved by the application of the scheduling approach to the abilities of the bacterial methods.

The experiments showed that in case of the simpler problem it was not worth to use the scheduling technique, because its overhead deteriorated the advantage gained by the possibility of adaptively switching to the currently best performing optimization algorithm.

However, in case of the more difficult problem the advantage of adaptive switching was higher than the disadvantage of the overhead originating from the comparing requirements, and thus the interpolative system involving the scheduling approach outperformed the other ones.

Further research aims at improving the scheduling algorithm and applying the approach for other architectures, such as hierarchical and hierarchical-interpolative fuzzy systems.

ACKNOWLEDGMENT

This paper was supported by the National Scientific Research Fund Grants OTKA K75711 and OTKA K105529, a Szchenyi Istvn University Main Research Direction Grant and the Social Renewal Operation Programmes TMOP-4.2.2 08/1-2008-0021 and 421 B.

REFERENCES

[1] E. Alpaydin, "Introduction to Machine Learning" (The MIT Press, 2004) 445 p.

[2] E.P. Klement, L. T. Kóczy and B. Moser, "Are fuzzy systems universal approximators?", *International Journal of General Systems*, 28 i2–3 259–282.

[3] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning", parts I, II, and III, *Information Sciences*, 8, 8, 9 (1975) pp. 199–249, pp. 301–357, pp. 43–80.

[4] L. T. Kóczy and K. Hirota, "Approximate reasoning by linear rule interpolation and general approximation", *International Journal of Approximate Reasoning* 9 (1993) pp. 197–225.

[5] L. T. Kóczy and K. Hirota, "Interpolative reasoning with insufficient evidence in sparse fuzzy rule bases", *Information Sciences* 71 (1993) pp. 169–201.

[6] M. Sugeno, T. Murofushi, J. Nishio, and H. Miwa, "Helicopter control based on fuzzy logic", *Second Fuzzy Symposium on Fuzzy Systems and Their Applications to Human and Natural Systems* Tokyo – Ochanomizu, Japan, 1991.

[7] M. Sugeno, M. F. Griffin, and A. Bastian, "Fuzzy hierarchical control of an unmanned helicopter", *Proceedings of the 5th IFSA World Congress (IFSA93)*, Seoul, South Korea, 1993, pp. 1262-1265.

[8] L. T. Kóczy and K. Hirota, "Interpolation in hierarchical fuzzy rule bases with sparse meta-levels", Technical Report 97-3, Tokyo Institute of Technology, Yokohama, 1997.

[9] L. T. Kóczy, K. Hirota and L. Muresan, "Interpolation in hierarchical fuzzy rule bases", *International Journal of Fuzzy Systems* 1, no. 2 (1999) pp. 77–84.

[10] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano, "Fuzzy rule extraction by bacterial memetic algorithms", *Proceedings of the 11th World Congress of International Fuzzy Systems Association, IFSA 2005*, Beijing, China, 2005, pp. 1563–1568.

[11] K. Balázs, J. Botzheim and L. T. Kóczy, "Comparison of Various Evolutionary and Memetic Algorithms", *Proceedings of the International Symposium on Integrated Uncertainty Management and Applications, IUM 2010*, Ishikawa, Japan, 2010, pp. 431–442.

[12] K. Balázs, J. Botzheim and L. T. Kóczy, "Comparative Analysis of Interpolative and Non-interpolative Fuzzy Rule Based Machine Learning Systems Applying Various Numerical Optimization Methods", *World Congress on Computational Intelligence, WCCI 2010*, Barcelona, Spain, 2010, pp. 875–982.

[13] K. Balázs, L. T. Kóczy: Constructing Dense, Sparse and Hierarchical Fuzzy Systems by Applying Evolutionary Optimization Techniques, Applied and Computational Mathematics, Vol. 11, No. 1, 2012, pp. 81–101.

[14] K. Balázs, L. T. Kóczy: A Remark on Adaptive Scheduling of Optimization Algorithms, International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2010, Dortmund, Germany, 2010, pp. 719–728.

[15] J. Espinosa and J. Vandewalle, "Constructing Fuzzy Models with Linguistic Integrity from Numerical Data — AFRELI Algorithm", *IEEE Transactions on Fuzzy Systems*, *8*, no. 5 (2000) pp.591–600.

[16] J. M. Alonso, O. Cordón, S. Guillaume and L. Magdalena, "Highly Interpretable Linguistic Knowledge Bases Optimization: Genetic Tuning versus Solis-Wetts. Looking for a good interpretability-accuracy trade-off", *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE*, pp. 901–906.

[17] D. Driankov, H. Hellendoorn, M. Reinfrank, "An Introduction to Fuzzy Control", (Springer, New York, 1996) 316 p.

[18] N. E. Nawa and T. Furuhashi, "Fuzzy system parameters discovery by bacterial evolutionary algorithm", *IEEE Transactions on Fuzzy Systems*, 7, no. 5, 1999, pp. 608–616.

[19] J. H. Holland, "Adaption in Natural and Artificial Systems", The MIT Press, Cambridge, Massachusetts, 1992.

[20] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms", *Technical Report Caltech Concurrent Computation Program*, Report. 826, California Institute of Technology, Pasadena, USA, 1989.

[21] J. A. Snyman, "Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms" (Springer, New York, 2005).

[22] K. Levenberg, "A method for the solution of certain non-linear problems in least squares", *Quart. Appl. Math.*, 2, no. 2 (1944) pp. 164–168.

[23] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", *J. Soc. Indust. Appl. Math.*, 11, no. 2 (1963), pp. 431–441.

[24] K. Balázs, L. T. Kóczy: Constructing Dense Fuzzy Systems by Adaptive Scheduling of Optimization Algorithms, International Fuzzy Systems Association World Congress (IFSA 2013), 2013, (submitted paper).

[25] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework, Journal of Multiple-Valued Logic and Soft Computing 17:2–3 (2011) pp. 255–287.

[26] D. Tikk, I. Joó, L. T. Kóczy, P. Várlaki, B. Moser and T. D. Gedeon, "Stability of interpolative fuzzy KH-controllers", *Fuzzy Sets and Systems*, 125 (2002) pp. 105–119.