Guest editors' foreword

In the past few years computational intelligence has become one of the main research topics at the Széchenyi István University, supported by a continuously renewed annual grant provided by the Research Council.

The multidisciplinary character of the Faculty allows several interesting engineering application oriented research directions, such as applications in civil and transportation engineering. There is also basic research going on in models and algorithms, further in control and technological aspects of computational intelligence.

The multidisciplinary Ph.D. school has several students working in the field and by now the name of the University never misses from the list of authors of all major CI conferences in the world, such as WCCI, IFSA WC, IPMU, etc.

We successfully organized the First Győr Symposium on Computational Intelligence (GYSCI) on 23 September 2008. It was a great pleasure that some of the leading experts in the field in neighbouring countries and within Hungary have accepted our invitation to the First Győr Symposium, such as the keynote speaker Professor Erich Peter Klement from Johannes Kepler University Linz, or our invited speakers Professor Peter Sinčák and Dr. Ján Vaščák from Košice, Professors József Dombi from Szeged and Szilveszter Kovács from Miskolc, etc. A large number of our graduate students and colleagues have also presented their most recent results.

This special issue of the Acta Technica Jaurinensis is mainly based on lectures presented at the conference. It is our intention that we will periodically publish such thematic issues in the future, this is why the subtitle 'Series Intelligentia Computatorica', i.e. Series of Computational Intelligence was added, along with a change of design of the cover page. This series will be open to submissions from all areas of CI, especially Fuzzy Systems, Neural Networks, Evolutionary Computation and Expert Systems, both to speakers of the coming GYSCI conferences and prospective authors not connected with these symposia, or the University.

This volume contains 13 papers.

The first paper, written by Zlatko Fedor and Peter Sinčak propose and implement an incremental system for linguistic command recognition in multi-agent system MASS with Adaptive Resonance Theory.

Our paper with Péter Földesi proposes a way of defining fuzzy exponents that can be useful in the fuzzy extension of customer satisfaction models. In case of power functions and fuzzy exponents the asymmetric features of function values must be handled. The paper presents a simple solution for readjusting the asymmetry that can fulfil the requirements.

József Sziray deals with the calculations performed in the reasoning process of rule-based expert systems, where inference chains are applied. He presents a logic model for representing the rules and the rule base of a given system.

Our paper with Rita Lovassy and László Gál presents the concepts of fuzzy J-K and D flip-flops based on various t-norms. We propose a fuzzy neural network (FNN), in which the fuzzy flip-flops with quasi sigmoidal *J-Q(t+1)* characteristics implement the neurons in a Multilayer Perceptron.

The paper of Ján Vaščák deals with the application of Fuzzy Cognitive Maps in combination with a graph search algorithm A^* for purposes of path planning for a vehicle in a traffic system with dynamic changes.

Our other paper with László Gál summarizes the bacterial type evolutionary algorithms used for fuzzy rule base identification. We propose here an improved version of the bacterial memetic algorithm.

Péter Keresztes and Timót Hídvégi present the most interesting parts of the design process of an emulated digital Cellular Neural Network (CNN) Universal Machine chip.

Krisztián Balázs investigates the De Morgan identities in fuzzy set theory, especially the possibility of non-dual behavior.

Szabolcs Nagy, Péter Baranyi and Péter Gáspár deal with rollover prevention to provide a heavy vehicle with the ability to resist overturning moments generated during cornering. They study a combined yaw-roll model including the roll dynamics of unsprung masses.

Our paper with Áron Ballagi and Tamás D. Gedeon proposes that communication among intelligent robots by intention guessing and fuzzy evaluation of the situation might lead to effective cooperation and the achievement of tasks that cannot be done without collaboration and communication.

The paper by Gyula Agárdy presents the application of fuzzy rule based systems in bridge management.

Our study with Katalin Tamás shows a way of defining fuzzy signature based models, and introduces generalized Mamdani-type inference on fuzzy signature based models. We have implemented a software capable of calculating a conclusion for a fuzzy signature observation.

At last, the paper written by Szilveszter Kovács gives a short survey on various Fuzzy Rule Interpolation (FRI) methods together with a simple demonstration of their application benefits.

We hope the Reader will study with interest the articles presented in this special issue.

Győr, December 2008

László T. Kóczy

János Botzheim

Guest Editors

AIBO Talking Procedure based on Incremental Learning Approach

Zlatko Fedor, Peter Sinčák

**Center for Intelligent Technologies,
Department of Cybernetics and Artificial
Intelligence, FEI TUKE Kosice
Home page: http://www.ai-cit.sk**

Abstract. The target of this project is to propose and implement incremental system for linguistic command recognition in multi agent system MASS, based on client-server architecture. Preprocessing is realized with the aid of Mel-frequency cepstral coefficients and classification is realized by modified MF Artmap. System allows remote parallel learning of various commands, their consecutive identification and robot dog AIBO control.

Keywords: recognition, words, MF Artmap, MFCC, multi-agent, client-server, incremental, system, MASS, AIBO

1. Project Definition and Task Determination

The goal of this project is to propose and implement incremental system for linguistic command recognition in multi-agent system MASS with Adaptative Resonance Theory (ART) like methods especially with modified MF Artmap and sound preprocessing which is realized with the aid of Mel-frequency cepstral coefficients ([1], 2006). Finally, the chosen methods in form of plugins are tested with this system.

2. The State of the Art in the Domain

If we want to solve some problems in real life with methods of artificial intelligence, we use very often recognition and classification. These concepts are very similar but there are slight differences between them. While in process of classification the number of classification classes is known, in recognition process these classes are being created during the recognition process. The concept of classification can be defined as follows: Incorporation of objects or events into specific classes by the decision rule. The objects which are familiar enough are incorporated into the same class. Generally the classification rule has some parameters which are changeable. This change of parameters is the training of the classification tool. In the domain of linguistic command recognition, the neural networks are the common classification tools and the recurrent types of them with adaptive resonance are the best choice in many cases.

3. Selected Methods and Approaches

Modified MF Artmap is derived from the existing MF Artmap ([8], 2002) model. It utilizes all advantages of original MF Artmap, for example speed of learning/classification and identification of unknown classes. Moreover some errors from this neural network have been removed.

First modification was the change of work with parameter R on comparative layer network. In the original network it performs check of distance from the central cluster for every dimension separately. Original network is using the same parameter R for all dimensions and clusters.

It brings the following disadvantages:

1. Clusters have very similar measurements and they can't have different size in different dimension.
2. Creation of extra clusters which are not needed.
3. The occurrence of unclassified inputs even if they belong to certain clusters.

These problems were solved as follows. Comparison distance from the centre cluster for every dimension is done with use of parameter R which is different for every dimension and cluster. Then with the creation of a new cluster, R parameters are assigned for every dimension. The parameters are from interval <0, 1>. Second modification is the change of update logic for parameter R. At first parameter q is updated by formula:

$$qn = qs + 1$$

where qs is the count of examples in cluster before addition of new example, qn is the count of examples in the updated cluster.

Next step is to update parameter X for every dimension. Update of this parameter is by original MF Artmap formula:

$$Xn = Xs + \frac{1}{qn} \cdot (Xs - X)$$

where Xn is the position of the new centre cluster in actual dimension, Xs is the value original cluster centre, X is position of the new example, qn is the count of examples in cluster.

Finally for every dimension of cluster parameter R is updated with of this formula:

$$Rn = Rs + \frac{sign \cdot \left\| Xn - X \right| - \left| Xs - X \right\|}{qn}$$

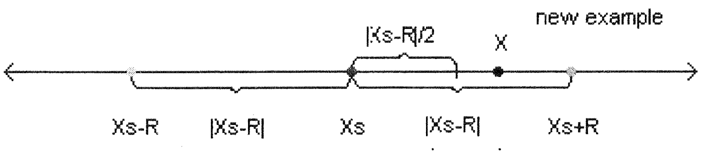


Fig. 1. Update of the dimension cluster

where Rn is new radius of the cluster for actual dimension, R is old cluster radius, qn is the count of examples in actual cluster, |Xn- X| is distance of sample from the centre cluster, |Xs- X| is distance of sample from the cluster center before change, parameter sign is described here by the formula:

$$sign = \begin{cases} -1, \dfrac{|Xs - R|}{2} > |Xs - X| \\ 1, otherwise \end{cases}$$

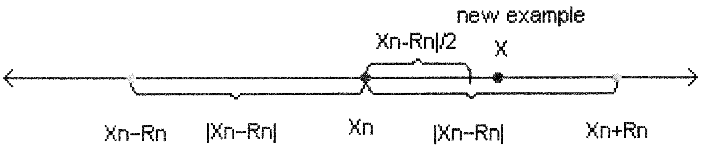


Fig. 2. Updating process for the modification of the cluster parameters

Next modification of the network was done to solve error situation in original network which occurs when new sample falls in the middle of some cluster but had another class. In this case, that cluster is deleted from the network.

4. Design and Implementation

Everything from the previous part was implemented as a plugin for MASS. This part will describe MASS and plugin types which are used in this system.

MASS could be described as multi-agent, incremental, plugin system with client-server architecture. With plugins for object recognition it is possible to learn various objects or to recognize them in parallel manner for many clients around the world. Gained knowledge will be stored on server which will host the object recognition setup in MASS.

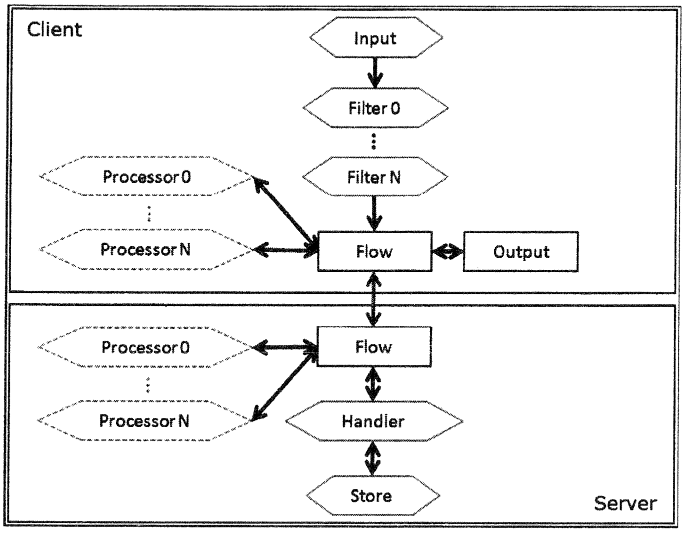


Fig. 3. MASS Plugin system

Fig. 1 describes the plugin system of MASS. Red plugin types are required and together they represent the smallest possible system. Green plugin types are optional. As you can see there can be more filter and processor plugins in the system and processor plugins can by placed whether on client or server side.

Flow plugin is special because the same plugin is placed on both sides and these sides are communicating together. Each plugin type will be briefly described in the sequel.

Flow is required part of the system. This plugin type is managing the client-server communication. If some user input is needed, form is included in the plugin.

Output is required part of the system. Its task is to provide and react on results provided by server. The reaction could be manipulation with some connected robot or device.

Input is optional part of the system. It provides input data from devices like webcams, microphones and sensors.

Filter is optional part of the system. It modifies its input, which can be from Input or other Filter plugin. The purpose of its use is similar to that of Filters known from image or audio processing.

Processor is optional part of the system. It can be placed whether on client or server side of the system. Processor should change the input data to data which can be used in classification, clustering or other types of Handler plugin tasks.

Handler is optional part of the system. This plugin should have all the functionality required for manipulation with data in database. Classification, clustering and other similar operations should be implemented in this type of plugin.

Store is optional part of the system. Here should be implemented everything related to data storage. This could be implemented all by authors or they can implement link to SQL or similar database system. Moreover also internet can be considered as some sort of database.

5. Experiments

This section presents experiments on robotic dog AIBO in Slovak language. Experiments are using aibo in "remote" regime, when robot could be controlled over wifi interface. Plugin InputOutputAudioAibo allows to read the audio data from the robot microphone and to send him commands that are dog performs.

5.1. Training

Experiments are using following verbal commands from four speakers. Commands were spoken by three men and one woman. Total word count for training was 165. Individual verbal commands were spoken directly to robot AIBO from approximately one meter distance without disturbing environment sound. In the next table is the count of recorded commands from individual speakers:

commands in Slovak language	count				
	speaker 1 (man)	speaker 2 (man)	speaker 3 (woman)	speaker 4 (man)	total count
sadni	5	4	5	3	17
ľahni	5	2	5	4	16
vstaň	4	3	3	5	17
tancuj	5	3	3	5	16
kopni	5	3	4	2	14
doprava	6	3	5	5	19
doľava	5	4	3	4	16
dopredu	5	3	4	4	16
dozadu	5	5	4	5	19
lez	5	3	3	4	15

5.2. Testing

The test set consists of 58 verbal commands. Speakers were the same from the training stage. Commands were spoken directly to aibo at approximately one meter distance without disturbing environment sound. Count of the commands are showed in the next table:

commands in Slovak language	count				
	speaker 1 (man)	speaker 2 (man)	speaker 3 (woman)	speaker 4 (man)	total count
sadni	1	1	0	3	5
ľahni	2	2	0	2	6
vstaň	2	1	2	2	7
tancuj	1	1	2	1	5
kopni	2	1	1	3	7
doprava	2	1	1	1	5
doľava	2	1	2	1	6
dopredu	1	1	1	2	5
dozadu	2	0	1	1	4
lez	3	1	2	2	8

5.3. Results

Parameter R for the neural network was 0.6. Next table shows the classification percentage of testing commands.

actual class	predicting class									
	Sadni	ľahni	vstaň	tancuj	kopni	doprava	doľava	dopredu	dozadu	lez
sadni	100	0	0	0	0	0	0	0	0	0
ľahni	16.7	83.3	0	0	0	0	0	0	0	0
vstaň	0	0	100	0	0	0	0	0	0	0
tancuj	0	0	0	100	0	0	0	0	0	0
kopni	0	0	0	14.3	85.7	0	0	0	0	0
doprava	0	0	0	0	0	60	40	0	0	0
doľava	0	0	0	0	0	33.3	66.7	0	0	0
dopredu	0	20	0	0	0	0	0	80	0	0
dozadu	0	0	0	0	0	0	0	0	100	0
lez	0	0	0	0	0	0	0	0	0	100

Final classification accuracy is 87.93%.

In the third experiment the cycle count was increased to 5. Results are in the next table.

actual class	predicting class										
	sadni	ľahni	vstaň	tancuj	kopni	doprava	doľava	dopredu	dozadu	lez	unknown class
sadni	100	0	0	0	0	0	0	0	0	0	0
ľahni	16.6	50	0	16.6	0	0	16.6	0	0	0	0
vstaň	0	0	100	0	0	0	0	0	0	0	0
tancuj	0	0	0	100	0	0	0	0	0	0	0
kopni	0	0	0	0	100	0	0	0	0	0	0
doprava	0	0	0	0	0	80	0	0	0	0	20
doľava	0	0	0	0	0	16.6	83.4	0	0	0	0
dopredu	0	0	0	0	0	0	0	80	0	0	0
dozadu	0	0	0	0	0	0	0	25	75	0	0
lez	0	0	0	0	0	0	0	0	0	100	0

Final classification accuracy was been 86.2%.

6. Contribution to the Results in the Domain

From the experiments you can see that similar words in way of pronunciation could confuse the network. One confusing example are words "doľava" and "doprava" which caused bad classification quite often. This special case could be solved by teaching only "ľava" and "prava". This also shows the robustness of given system.

7. Conclusion

This work is using modified version of MF Artmap neural network which reach better results in comparison with original MF Artmap network. It was showed in experiments with almost 88% of classification accuracy. System MASS allowed simple implementation of individual methods that were needed for the recognition in form of plugins.

Output of this work is modified MF Artmap network, plugins for recognition of isolated words for system MASS. After system startup of MASS, people for all over the world can teach system new words and improve the quality of recognition.

Next research could improve recognition with additional improvements of the neuronal network. Moreover, it could be convenient to implement that system will ask for class of the word which is not learned and it will learn it afterwards. It would be good to create filter that will be able to remove disturbing environment sounds and focus only on speaker voice.

Acknowledgement:

Research supported by the National Research and Development Project Grant 1/0885/08 "Learnable Systems based on Computational Intelligence" 2008-2010

References

[1] Psutka, J.: *Mluvíme s počítačem česky*. Academia, Praha (2006)

[2] Nuttall, A. H.: Some *Windows with Very Good Sidelobe Behavior*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol.ASSP-29, No.1, February, (1981). Dostupné na internete: http://en.wikipedia.org/wiki/Window_function#Hamming_window

[3] Černocký, J., Burget L: *Parametrizace reči*. FIT VUT Brno. Dostupné na internete: http://www.fit.vutbr.cz/~cernocky/speech/pred/11_priznaky/11_priznaky.pdf

[4] Olajec, J.: *Návrh rozpoznávania izolovaných čísloviek.*, 2004. Department of Telecommunications, EF ZU Zilina. Dostupné na internete: http://kt.uniza.sk/~olajec/publications/2004_Jan_Olajec_Diplomova_praca.pdf

[5] Psutka, J., Muller, L.: *Optimization of Same Parameters in the Speech Parameters in the Speech-Processing Module Developed for the Speaker Independent ASR System*. In: Proc. Of IIIS 2003, Orlando, USA, (2003), s. 414-418

[6] Sinčák, P., Andrejková, G.: *Neurónové siete*. Inžinierske aplikácie II. Dostupné na internete: http://www2.fiit.stuba.sk/~cernans/nn/nn_download/ Sincak_Andrejkova_vol_2.pdf

[7] Olajec, J., Jarina, R.: *Použitie metódy 3TDCM pri rozpoznávaní izolovaných slov neurónovou sieťou*, IEEE Vršov 2005, October 2005, Vršov, Cech Republic, ISBN 80-214-3008-7. Dostupné na internete: http://kt.uniza.sk/~olajec/publications/%5B02%5D_-_Vrsov_2005.pdf

[8] Hric M.: *Integrácia neurónových sietí typu ARTMAP s prvkami fuzzy systémov pre klasifikačné úlohy*. Dostupné na internete: http://www.ai-cit.sk/source/ publications/thesis/master_thesis/2000/hric/html/index.html

[9] Pai, H. F., Wang H. C.: *A study of the two-dimensional cepstrum approach for speech recognition*, Computer Speech and Language vol.6 (1992)

[10] Ariki, Y., Mizuta, S., Nagata, M., Sakai, T.: *Spoken-word recognition using dynamic features analysed by two-dimensional cepstrum*, IEE Proceedings, vol.136 (1989)

[11] Hudec, M.: *Prehľad problematiky*. Dostupné na internete: http://www.elajnus.sk/diplomovka/files/prehlad.pdf

[12] Biswas, S., Ahmad, S., Islam Mollat, M.K.: *Speaker Identification Using Cepstral Based Features and Discrete Hidden Markov Model*, Information and Communication Technology, (2007). ICICT apos;07

[13] Molnárová, M., Spalek, J.: *Fuzzy monitoring of the safety-related critical processes*, In: Híradástechnika 9/2001, Vol. LVI., ISSN 0018-2028, Budapest, pp. 21-24

[14] Molnárová, M., Spalek, J., Šurín, P.: *The Use of Fuzzy Logic for Safety-Related Decision*, In: IFAC Conference Control Systems Design, Bratislava, 18.-20. June (2000), pp. 548-553

[15] Hájek, P, Olej, V. : *Municipal Creditworthiness Modelling by NEURAL Networks*, Accepted to Acta Electrotechnika , Informatika TU Košice

Fuzzy Exponents for Heuristic Based Applications

P. Földesi[1], J. Botzheim[2], L.T. Kóczy[3,4]

[1]Department of Logistics and Forwarding, Széchenyi University H-9026, Győr, Egyetem tér1. Hungary, foldesi@sze.hu

[2]Department of Automation, Széchenyi University H-9026, Győr, Egyetem tér 1. Hungary, botzheim@sze.hu

[3]Faculty of Engineering Sciences, Széchenyi University H-9026, Győr, Egyetem tér 1. Hungary, koczy@sze.hu

[4]Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, H-1117, Budapest, Magyar tudósok krt. 2. Hungary koczy@tmit.bme.hu

Abstract: The fuzzy extensions for heuristic based optimizing algorithms often face the problem of increased number of calculations required to find the solutions. Even in linear cases the running time can be significantly larger than for crisp versions. In several management decision making processes the uncertainty of circumstances are represented by fuzzy sets and numbers, and non-linear features occur as well. In order to handle that kind of non-linearity, appropriate representation of the fuzzy power function is to be used that can keep the required computation time and resources at a reasonable level. In the paper different solutions are compared from practical points of views when the bacterial evolutionary algorithm is used for the approximation of the optimum. Suggestions for the representation of fuzzy exponents are made as well.

Keywords: fuzzy power function, non-linear optimization, bacterial evolutionary algorithm

1. Introduction

In heuristic based applications a key issue is the computational effort. The bacterial evolutionary algorithm, like other soft computing tools can provide a good compromise between the computational complexity and the accuracy of the solution. If we would like to achieve better results, then the parameters of the algorithm need to be increased, and therefore, more computation is needed. Our goal is to diminish the evaluation time of one solution, thus more evaluations can be performed within the same time as with the more complicated descriptions of fuzzy exponents. The fuzzy exponent approaches proposed in the literature [2,3] are using complicated shaped membership functions, thus they cannot be used in applications where lots of evaluations of these membership functions are necessary. Instead of these techniques, in our approaches, only the main characteristic points are considered in the description of fuzzy exponent numbers. Several approaches have been presented and it must be emphasized that different applications require special representation in order to keep the advantages of the given procedure [1,6,7], so in case of bacterial evolutionary algorithm a certain approach is to be followed [4,5]. In the paper different solutions are compared from practical points of views when the bacterial evolutionary algorithm is used for the approximation of the optimum. Suggestions for the representation of fuzzy exponents are made as well.

2. Bacterial evolutionary algorithm

Nature inspired some evolutionary optimization algorithms suitable for global optimization of even non-linear, high-dimensional, multi-modal, and discontinuous problems. The original genetic algorithm was based on the process of evolution of biological organisms. It uses three operators: reproduction, crossover and mutation. Later, new kind of evolutionary based techniques were proposed, which are imitating phenomena that can be found in nature. Bacterial Evolutionary Algorithm (BEA) [11] is one of these techniques. BEA uses two operators; the bacterial mutation and the gene transfer operation. These new operators are based on the microbial evolution phenomenon. Bacteria share chunks of their genes rather than perform a neat crossover in chromosomes. The bacterial mutation operation optimizes the chromosome of one bacterium; the gene transfer operation allows the transfer of information between the bacteria in the population. Each bacterium represents a solution for the original problem. BEA has been applied for wide range of problems, for instance optimizing the fuzzy rule bases [10,11].

The algorithm consists of three steps. First, an initial population has to be created randomly. Then, bacterial mutation and gene transfer are applied, until a stopping criterion is fulfilled. The evolution cycle is summarized as follows:

```
create initial population
do {
   apply bacterial mutation for each individual
   apply gene transfer in the population
} while stopping condition not fulfilled
return best bacterium
```

First, the initial (random) bacteria population is created. The population consists of N_{ind} bacteria (chromosomes). It is followed by the evolutionary cycle, which contains two operators. The bacterial mutation is applied to each chromosome one by one. First, N_{clones} copies (clones) of the bacterium are generated, then a certain segment of the chromosome is randomly selected and the parameters of this selected segment are randomly changed in each clone (mutation). Next all the clones and the original bacterium are evaluated and the best individual is selected. This individual transfers the mutated segment into the other individuals. This process continues until all of the segments of the chromosome have been mutated and tested. At the end of this process the clones are eliminated. In the next step the other operation, the gene transfer is applied, which allows the recombination of genetic information between two bacteria. First, the population must be divided into two halves. The better bacteria are called the superior half, the other bacteria are called the inferior half. One bacterium is randomly chosen from the superior half, this will be the source bacterium and another is randomly chosen from the inferior half, this will be the destination bacterium. A segment from the source bacterium is chosen randomly and this segment will overwrite a segment of the destination bacterium or it will be added to the destination bacterium. This process is repeated for N_{inf} times. The stopping condition is usually given by a predefined maximum generation number (N_{gen}). When N_{gen} is achieved then the algorithm ends otherwise it continues with the bacterial mutation step.

The basic algorithm has four parameters: the number of generations (N_{gen}), the number of bacteria in the population (N_{ind}), the number of clones in the bacterial mutation (N_{clones}), and the number of infections (N_{inf}) in the gene transfer operation.

3. Fuzzy power function and exponents

The question is how to represent the fuzziness of exponents so that the crisp algorithms can be kept. If we use more bacteria in the population, or more clones in the bacterial mutation, or the number of generations is increasing, then although the accuracy of the solution becomes higher, we need more evaluations in the bacterial operations.

In our paper fuzzy solutions for unvaried function $y = b + aX^{\beta}$ (where a, b, $\beta \in \mathbf{R}$) are investigated. An obvious solution is to weight the functions and sum up three weighted functions with weighted exponents (see Fig 1):

- a "central" function, where $\beta_2 = \beta_C$
- a "left-side" function where exponent is calculated by using α-cuts $\beta_1 = \beta_C - (1-\alpha)(\beta_C - \beta_L)$
- a "right-side" function, where the exponent is $\beta_3 = \beta_C + (1-\alpha)(\beta_R - \beta_C)$

The weights of the functions are calculated in proportion to $\mu(\beta_i)$ membership function values based on similar considerations. Thus the function weights are:

$1/(2\alpha + 1)$ for the "central" function

And

$\alpha/(2\alpha + 1)$ for the "left-side" and "right-side" functions.

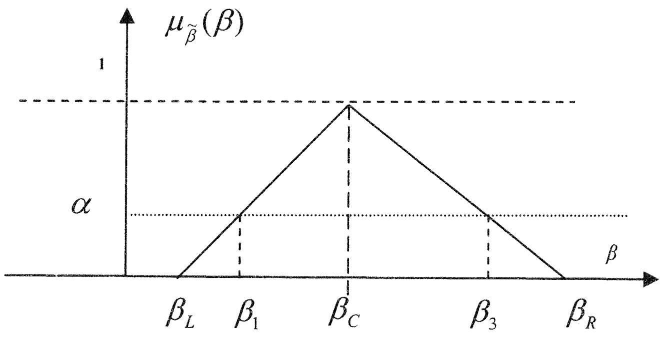


Fig. 1 Weights and exponents based on α-cuts

$$\mu_{\tilde{\beta}}(\beta)=\begin{cases}\dfrac{\beta-\beta_L}{\beta_C-\beta_L} & \beta_L \le \beta \le \beta_C \\ \dfrac{\beta_R-\beta}{\beta_R-\beta_C} & \beta_C \le \beta \le \beta_R \\ 0 & otherwise\end{cases} \tag{1}$$

Thus $\tilde{\beta}=(\beta_L,\beta_C,\beta_R)$, and its supporting interval is [β_L , β_R].

Then the fuzzy solution for unvaried function $y=b+aX^{\beta}$ (where a, b, $\beta \in \mathbf{R}$) is

$$y=\frac{1}{2\alpha+1}(b+ax^{\beta_C})+\frac{\alpha}{2\alpha+1}(b+ax^{\beta_C-(1-\alpha)(\beta_C-\beta_L)})+\frac{\alpha}{2\alpha+1}(b+ax^{\beta_C+(1-\alpha)(\beta_R-\beta_C)}) \tag{2}$$

Re-arranging (2) we obtain:

$$y=b+\frac{1}{2\alpha+1}a\,(x^{\beta_C}+\alpha\,x^{\beta_C-(1-\alpha)(\beta_C-\beta_L)}+\alpha\,x^{\beta_C+(1-\alpha)(\beta_R-\beta_C)}) \tag{3}$$

We can consider any *m=2k+1* functions *k=1,2,3, …* (see Fig. 2).

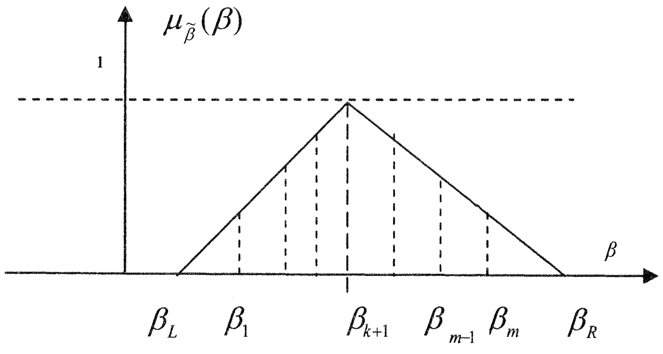


Fig. 2 Exponents for m functions

so we can write the following

$$y=\sum_{j=1}^{m}\frac{\mu(\beta_j)}{\sum_{j=1}^{m}\mu(\beta_j)}(b+a\,x^{\beta_j})=\frac{\sum_{j=1}^{m}\mu(\beta_j)\,(b+a\,x^{\beta_j})}{\sum_{j=1}^{m}\mu(\beta_j)} \tag{4}$$

For any shape of membership function we can write:

$$y=\frac{\int_0^{\infty}\mu(\beta)(b+a\,x^{\beta})d\beta}{\int_0^{\infty}\mu(\beta)d\beta} \tag{5}$$

Using (3) when the value of α → 0 the solution of bacterial evolutionary algorithm can be instable. The reason of this phenomenon is that when α → 0 the coefficients of the left side and right side → 0 as well, that is, the "importance of fuzziness" is getting smaller. Therefore we propose:

1/(3-2α) for the "central" function

And

1-α /(3-2α) for the "left-side" and "right-side" functions (see Fig. 3)

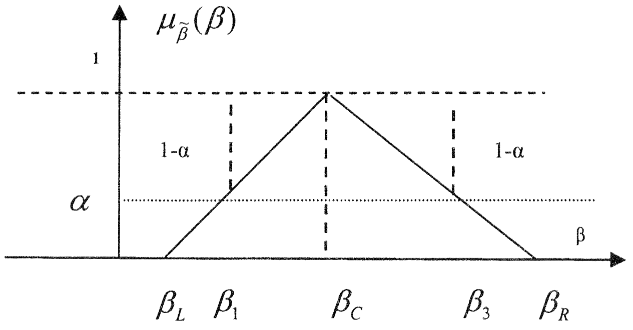


Fig. 3 Weights and exponents based on 1- (α-cuts)

Thus instead of (3) we obtain:

$$y=\frac{1}{3-2\alpha}(b+ax^{\beta_C})+\frac{1-\alpha}{3-2\alpha}(b+ax^{\beta_C-(1-\alpha)(\beta_C-\beta_L)})+\frac{1-\alpha}{3-2\alpha}(b+ax^{\beta_C+(1-\alpha)(\beta_R-\beta_C)}) \quad (6)$$

The general formula for any *m=2k+1* points can be given, where *k=1,2,3...* is the number of α-cuts (see Fig 4):

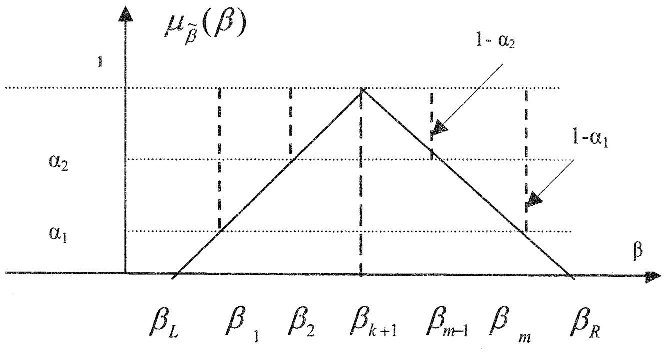


Fig. 4 Exponents for m functions based on 1-(α-cuts)

$$y=\frac{1}{1+m-\sum_{j=1}^{m}\mu(\beta_j)}(b+a\,x^{\beta_{k+1}})+\sum_{j=1}^{m}\frac{1-\mu(\beta_j)}{1+m-\sum_{j=1}^{m}\mu(\beta_j)}(b+a\,x^{\beta_j}) \quad (7)$$

Note, that $\mu(\beta_{k+1})=1$, thus $1-\mu(\beta_{k+1})=0$ and $\mu(\beta_j)=\mu(\beta_{m+1-j})$

The continuous formula is:

$$y = \frac{\int_{\beta_L}^{\beta_R} (1-\mu(\beta))(b + a\, x^{\beta})\,d\beta}{(\beta_R - \beta_L) - \int_{\beta_L}^{\beta_R} \mu(\beta)\,d\beta} \tag{8}$$

In case of triangular fuzzy numbers

$$\int_{\beta_L}^{\beta_R} \mu(\beta)\,d\beta = (\beta_R - \beta_L)/2$$

so (8) can be recast as:

$$y = \frac{2}{\beta_R - \beta_L} \int_{\beta_L}^{\beta_R} (1-\mu(\beta))(b + a\, x^{\beta})\,d\beta \tag{9}$$

The results are biased since the right side, the greater exponent values have the same coefficient, so certain transformations seem to be reasonable. One option is to use a slope instead of α-cuts:

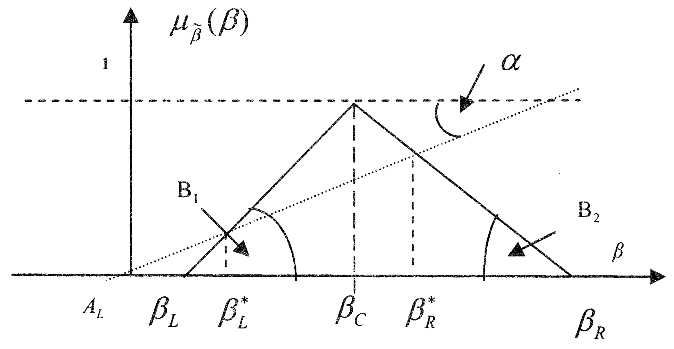


Fig. 5 Asymmetric representation of exponents based on slopes

Where $A_L = \beta_L - \gamma(\beta_R - \beta_L)$

and $\gamma \geq 0$, $0 \leq \alpha \leq arctg \dfrac{1}{\beta_C - \beta_L + \gamma(\beta_R - \beta_L)}$

Then $$\beta_L^* = \frac{\beta_L(tg\alpha + \gamma tg\alpha - tg\mathrm{B}_1) - \gamma\beta_R tg\alpha}{tg\beta - tg\mathrm{B}_1} \tag{10}$$

and its coefficient

$$\frac{1-(\beta_L^*-\beta_L)tg\mathrm{B}_1}{3-(\beta_L^*-\beta_L)tg\mathrm{B}_1-(\beta_R-\beta_R^*)tg\mathrm{B}_2} \tag{11}$$

For practical reasons this version is not too "attractive" considering the required computation time and resources. In order to eliminate that problem we can re-adjust the asymmetry by using asymmetric exponents and coefficients:

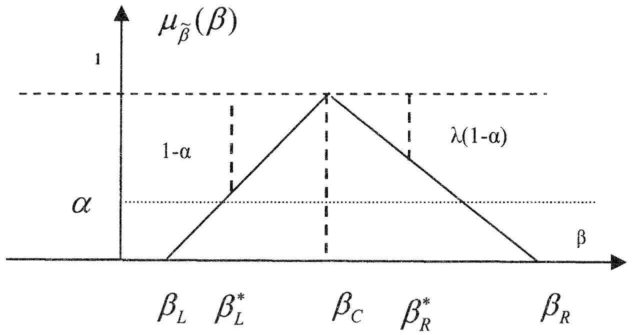


Fig. 6 Asymmetric representation of exponents based on different 1-(α-cuts)

Then instead of (6) we obtain

$$y=\frac{1}{1+(1+\lambda)(1-\alpha)}(b+ax^{\beta_C})+\frac{1-\alpha}{1+(1+\lambda)(1-\alpha)}(b+ax^{\beta_C-(1-\alpha)(\beta_C-\beta_L)})+\frac{(1-\alpha)\lambda}{1+(1+\lambda)(1-\alpha)}(b+ax^{\beta_C+(1-\alpha)\lambda(\beta_P-\beta_C)})$$

where $0\le\lambda\le1$ (12)

The general formula for any *m=2k+1* points can be given, where *k=1,2,3..* is the number of α-cuts

$$y=\frac{1}{2+k(1+\lambda)-\sum_{j=1}^{m}\mu(\beta_j)\lambda_j}(b+a\,x^{\beta_{k-1}})+\sum_{j=1}^{m}\frac{(1-\mu(\beta_j))\lambda_j}{2+k(1+\lambda)-\sum_{j=1}^{m}\mu(\beta_j)\lambda_j}(b+a\,x^{\beta_j}) \tag{13}$$

where $\lambda_j=\begin{cases}1 & if\ j=1,2,\dots,k+1\\ \lambda & if\ j=k+2,\dots,m\end{cases}$

The continuous formula is:

$$y=\frac{\int_{\beta_L}^{\beta_R}(1-\mu(\beta))(b+a\,x^{\beta})\lambda(\beta)d\beta}{(\beta_R-\beta_L)-\int_{\beta_L}^{\beta_R}\mu(\beta)\lambda(\beta)d\beta} \tag{14}$$

Where $\lambda(\beta)=\begin{cases}1 & \beta\le\beta_C\\ \lambda & \beta\ge\beta_C\end{cases}$

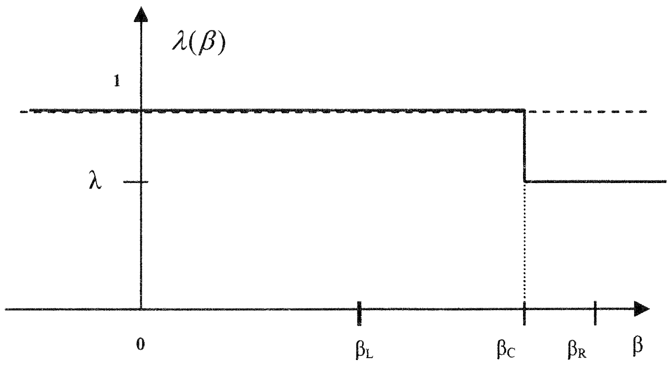


Fig. 7 Characteristic function for λ(β)

4. Practical application

4.1. Conditions and targets

For designing and developing products/services it is vital to know the relevancy of the performance generated by each technical attribute and how they can increase customer satisfaction. Improving the parameters of technical attributes requires financial resources, and the budgets are generally limited. Thus the optimum target is to achieve maximum customer satisfaction within given financial limits. Kano's quality model [9] classifies the relationships between customer satisfaction and attribute-level performance and indicates that some of the attributes have a non-linear relationship to satisfaction [8], rather power-function should be used. For the customers' subjective evaluation these relationships are not deterministic and are uncertain.

The benefit of fuzzy extension can be measured by the advantage we obtain analyzing the outputs. Difference between overall satisfaction values was considered, but what more important is the structure of technical attributions has to be examined. The aim is to allocate the limited resources subject to the maximum profit requirement. The customers' assessment of technical attributes is very uncertain especially at the beginning of product life-cycle so in Kano's model the exponents of satisfaction functions cannot be considered as deterministic values. For practical reasons a simple parametric representation must be used, in order to keep the required computation time and resources at a reasonable level.

4.2. Formulating the problem

The general target is to achieve the maximum economic result with the minimum use of resources, that is to maximize customer satisfaction with the minimum cost. The task can be mathematically formulated by maximizing overall satisfaction (S) not exceeding given cost limit (C)

Let $S_i(x_i) = b_i + a_i x_i^{\beta_i}$ $\quad i= 1,2 \ldots.n$ (15)

be the customer satisfaction generated by technical attribute x_i

$0 < x_i << \infty$ is a real number variable
$a_i > 0$ is a real constant
$\beta_i > 0$ is a real constant
b_i is a constant such that $sgn(b_i) = sgn(\beta_i - 1)$

further $C_i(x_i) = f_i + v_i x_i$ $\quad i=1,2,\ldots.n$ (16)

be the cost of manufacturing technical attribute at level x_i

$f_i \geq 0$, $v_i \geq 0$, $x_i \geq 0$ are real constants

Let $S = \sum_{i=1}^{n} (b_i + a_i x_i^{\beta_i})$ be the overall satisfaction (17)

and $C = \sum_{i=1}^{n} (f_i + v_i x_i)$ be the total cost. (18)

Then the general formula is:

Let $\sum_{i=1}^{n} S_i(x_i) \rightarrow$max, subject to $\sum_{i=1}^{n} C_i(x_i) \leq C_o$ (19)

where C_0 is a given constant.

An efficient solution in the practice is when the membership of each function is represented by the exponent β. If β is considered as a fuzzy number then the features of each technical attribute are given by the shape of the membership function. The main features of each set are:

- $0 < \beta < 1$ (degressive)
- $\beta=1$ (linear)
- $\beta > 1$ (progressive).

In our application the optimal values of these x_i variables need to be found. It is done by the bacterial evolutionary algorithm in which one individual is an $(x_1, x_2, \ldots, x_n)$ vector. The parameters of the bacterial evolutionary algorithm are the number of generations (N_{gen}), the number of bacteria in the population (N_{ind}), the number of clones in the bacterial mutation (N_{clones}), and the number of infections (N_{inf}) in the gene transfer operation. For the fuzzy model we use the following data and fuzzy exponents:

$S_1(x_1) = 10 + 0.1\, x_1^{1.8}$ $C_1(x_1) = 40 + 15\, x_1$
$S_2(x_2) = 5 + 0.15\, x_2^{1.8}$ $C_2(x_2) = 20 + 20\, x_2$
$S_3(x_3) = 0 + 0.45\, x_3^{1}$ $C_3(x_3) = 10 + 2\, x_3$
$S_4(x_4) = -5 + 0.15\, x_4^{0.5}$ $C_4(x_4) = 20 + 0.2\, x_4$
$S_5(x_5) = -10 + 0.35\, x_5^{0.5}$ $C_5(x_5) = 40 + 0.5\, x_5$
$C_0 = 1250$

$\beta_1 = (1.7,\ 1.8,\ 2.2)$
$\beta_2 = (1.6,\ 1.8,\ 2.1)$
$\beta_3 = (0.7,\ 1.0,\ 1.2)$
$\beta_4 = (0.4,\ 0.5,\ 0.8)$
$\beta_5 = (0.4,\ 0.5,\ 0.7)$

The results are shown in Table 1. and Table 2.

Table 1. Results of fuzzy approximation according to (3)

α	$\sum S_i$	x_1	x_2	x_3	x_4	x_5
0.01	256	0	0	559	3	2.66
0.1	301	74.6	0	0	1.1	0.5
0.5	304	74.5	0	0	3	3
0.9	249	0	0	559	3	2.7
[a]1.0	252	0	0	558	6	5.5

[a]Case of $\alpha=1$ means the crisp solution. We also get back the original results in case of $\alpha=0$, since the weights of the functions are 0

Table 2. Results of fuzzy approximation according to (6)

α	$\sum S_i$	x_1	x_2	x_3	x_4	x_5
0.01	561	74,6	0	0	0.43	0.4
0.1	492	74.6	0	0	0.5	0.4
0.2	429	74.6	0	0	0.2	0.2
0.3	378	74,6	0	0	0.4	0.4
0.4	337	74,6	0	0	0.7	0.3
0.5	304	74,6	0	0	0.5	0.5
0.6	279	74.5	0	0	1	0.6
0.7	253	0	0	559	3	2.7
0.8	251	0	0	559	3	2.7
0.9	251	0	0	559	3	2.7
[a]1.0	252	0	0	558	6	5.5

[a]Case of $\alpha=1$ means the crisp solution.

In the numerical example the fuzzy solution is significantly different from the crisp (deterministic) version, not only in terms of total satisfaction but – what is more important – in terms of technical attribute levels. In both case when $\alpha=0.9$ the x_i values are practically equal to the original solution, but at $\alpha=0.1$ the set of x_i-s transformed and attractive (progressive) technical attributes seem to be more important.

5. Conclusions

Soft computing applications set different requirements regarding the representation of uncertain, fuzzy values. These requirements are based on the nature of uncertainty and fuzziness, and the characteristics and features of applied algorithms must be assessed as well. There are several methods that can be a theoretic foundation of parametric representation, but in case of power function and fuzzy exponents the asymmetric features of function values must be handled as well, since for practical reasons (computation time and resources) the continuous formulas cannot be used efficiently. In this paper a simple solution for readjusting the asymmetry is also presented.

Acknowledgment

This paper was supported by Széchenyi István University Main Research Direction Grants 2008, and a National Scientific Research Fund Grant OTKA T048832.

References

[1] Buckley, J. J., Feuring, T.: *Linear and non-linear fuzzy regression : Evolutionary algorithm solutions,* Fuzzy Sets and Systems 112 (2000) pp. 381-394

[2] Zhong, Q., Yue, Z., Guangyuan, W.: Fuzzy random variable-valued exponential function, logarithmic function and power function, Fuzzy Sets and Systems 99 (1998) pp. 311-324

[3] Stefanini, L., Sorini, L., Guerra, M. L.: *Parametric representations of fuzzy numbers and application to fuzzy calculus,* Fuzzy Sets and Systems 157 (2006) pp. 2423-2455

[4] Földesi, P., Kóczy, L.T., Botzheim, J.: *Fuzzy extension for Kano's model using bacterial evolutionary algorithm*, Proceedings of 3rd International Symposium on Computational Intelligence and Intelligent Informatics, Agadir, Marocco (2007), pp. 147-152

[5] Földesi, P., Kóczy, L.T., Botzheim, J.: *Fuzzy solution for non-linear quality models*, Proceedings of 12th International Conference on Intelligent Engineering Systems, Miami, (2008) pp. 269-276

[6] Liu, S. T.: *Geometric programming with fuzzy parameters in engineering optimization,* International Journal of Approximate Reasoning 46 (2007) pp. 484-498

[7] Klir, G. J.: *Fuzzy arithmetic with requisite constrains*, Fuzzy Sets and Systems 91 (1997) pp. 165-175

[8] Matzler, K., Bailom, F., Hinterhuber, H.H., Renzl, B., Pichler, J.: The asymmetric relationship between attribute-level performance and overall customer satisfaction: a reconsideration of the importance-performance analysis, Industrial Marketing Management 33 (2004) pp. 271-277

[9] Kano, N., Seraku, N., Takahashi, F., Tsuji, S.: *Attractive quality and must-be quality. Hinshitsu*, The Journal of Japanese Society for Quality Control (1984) pp. 39-48

[10] Botzheim, J., Hámori, B., Kóczy, L.T., Ruano, A.E.: *Bacterial algorithm applied for fuzzy rule extraction*, in Proceedings of the International Conference on

Information Processing and Management of Uncertainty in Knowledge-based Systems, Annecy, France, (2002) pp. 1021-1026

[11] Nawa, N. E., Furuhashi, T.: *Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm*, IEEE Tr. Fuzzy Systems 7 (1999) pp. 608-616

A Logic Model for Rule-Based Expert Systems

József Sziray

Department of Informatics, Széchenyi University
Egyetem tér 1. 9026 Győr, Hungary
E-mail: sziray@sze.hu

Abstract: This paper deals with the calculations in the reasoning process of rule-based expert systems, where inference chains are applied. It presents a logic model for representing the rules and the rule base of a given system. Also, the fact base of the same expert system is involved in the logic model. The proposed equivalent representation manifests itself in a logic network. After that, a four-valued logic algebra is introduced. This algebra is used for the calculations where forward chaining is carried out. Next, the notion of line-value justification is described. This operation is applied in the backward chaining process, also on the base of the previously introduced four-valued logic. The paper describes two exact algorithms which serve for the forward and backward chaining processes. These algorithms can be implemented by a computer program, resulting in an efficient inference engine of an expert system. The achieved result enhances the reliability and usability of the intelligent software systems which is extremely important in embedded environments.

Keywords: Expert system, rule base, inference chains, computational complexity, multi-valued logic.

1. Introduction

The application area of embedded systems and the related economical and reliability requirements imply a specific hardware-software structure that is significantly different from the resources available in modern high-end systems. The relatively low processing performance, small memory space and the safety prescriptions have resulted in various architectural properties and programming solutions [1]. In case of real-time safety-critical systems the reaction time for the external events is a key issue [2]. It means that the speed of the calculations is a critical factor. On the other hand, the same applies to the memory consumption.

In many cases, artificial intelligence is realized within the frames of expert systems. This approach has gained a wide-spread use in controlling railway stations, dangerous chemical processes, power stations, airplane flights, medical systems, etc. These applications are equally related to safety-oriented systems.

As known, the most common form of storing knowledge in expert systems is the use of rules. It means that the knowledge base (long-term memory) consists of rules and facts. The other component of such an expert system is the inference engine which is the most important factor for a successful operation. An inference engine usually works in a fixed manner, for example, it could be designed as either data driven (i.e., forward reasoning or forward chaining) or goal driven (i.e., backward reasoning or backward chaining), however, most of the modern systems may well use both ways of reasoning [3]-[6].

The major concern related to the inference processes is their excessive computational amount. The algorithmic complexity derives from the fact that the task to be solved belongs to the so-called NP-complete problems. As known, NP-complete problems have a computational complexity for which there exists no upper bound by a finite-degree polynomial of the problem size. It means actually that the number of the computational steps is finite, but unpredictable [6]-[9]. Here the problem size can be expressed by the number of rules in the knowledge base. Due to the described features of the computations, the execution speed of the software is a crucial factor. This feature concerns especially the embedded real-time systems, where the response time must always be kept within a previously specified limit.

The paper deals with the calculations in the reasoning process of rule-based expert systems, where inference chains are applied. It presents a logic model for representing the rules and the rule base of a given system. Also, the fact base of the same expert system is involved in the logic model. The proposed equivalent representation manifests itself in a logic network. After that, a four-valued logic algebra is introduced. This algebra is used for the calculations where forward chaining is carried out. Next, the notion of line-value justification is described. This operation is used in the backward chaining process, also on the base of the previously introduced four-valued logic. The paper describes two exact algorithms which serve for the forward and backward chaining processes. These algorithms can be implemented by a computer program, resulting in an efficient inference engine of an expert system.

2. Fundamental Concepts

In a rule-based system, any rule consists of two parts: the IF part, called the antecedent (premise or condition) and the THEN part, called the consequent (conclusion or action). The basic syntax of a rule is:

IF <antecedent> THEN <consequent>.

In general, a rule can have multiple antecedents joined by the keywords AND (conjunction), OR (disjunction), or a combination of both. Negation of an antecedent is also allowed. In this case the NOT operator is used. For example,

IF the spill is liquid
AND the spill pH < 6
AND the spill smell is vinegar
THEN the spill material is acetic acid.

Forward chaining is an inference method where rules are matched against facts to establish new facts, finally reaching a conclusion. In case of backward chaining the system starts with what it wants to prove, and tries to establish the facts it needs to prove the initial fact. The components of the reasoning process that are applied, constitute the so-called inference chain.

The knowledge base consists of the set of rules (rule base), and the set of facts (fact base), where the rule base is permanent, while the fact base contains an initial set of facts depending on the actual task to be solved, and it changes in accordance with the concrete reasoning process.

The existent expert systems build up the knowledge base in a usual data-base structure, and their inference engine applies an exhaustive search through all the rules during each cycle. The aim of the search is to find the appropriate rules for which the antecedents or the consequents satisfy the actual conditions. As a consequence of this process, systems with a large set of rules (over 100 rules) can be slow, and thus they may be unsuitable for real-time applications, especially in the field of embedded systems [4].

In the following a novel knowledge representation based on Boolean algebra and logic networks will be presented. On this base, a four-valued logic system is introduced. This new model results in a significantly more efficient inference processing than the classical one. The computational improvement is estimated to be at least two orders of magnitude, which is due to the small memory usage and fast operations in the logic domain.

3. The Use of Boolean Algebra and Logic Networks

The relations of Boolean algebra can also be used for the rule-based systems. As it is well-known, Boolean logic involves two values: 0 (false) and 1 (true), where the following three basic operations are used: logic AND (denoted by the multiplication point ($\cdot$), logic OR (denoted by the addition sign (+), and logic NOT (denoted by an apostrophe succeeding the actual variable). For instance, A' means the negation of A.

It can be easily seen that the logic conditions within the rules can directly be substituted by the corresponding Boolean operations and logic gates [10]. As an example let us consider the following set of rules, where the facts are denoted by capital letters:

IF C AND D THEN L,

IF NOT E THEN K,

IF L OR K THEN P,

IF E AND M THEN Q.

The Boolean description of the above rules is the following:

$L = C \cdot D,$

$K = E',$

$P = L + K,$

$Q = E \cdot M.$

These four rules can be represented by four logic gates: two AND gates, one NOT gate, and one OR gate. Now, if we connect the inputs and outputs of these gates in accordance with the identical letters, the logic network of Figure 1 will be obtained.

It should be noted here that in case of a simple direct rule, for example,

IF U THEN V,

its corresponding Boolean form will be

$V = U.$

This relation is represented by a YES gate which does not modify its input value.

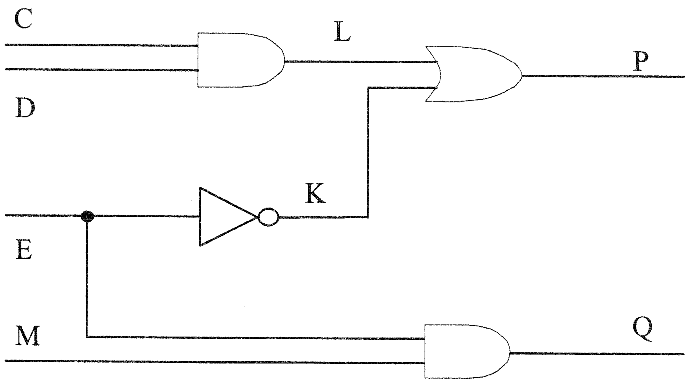


Figure 1. The logic net work for the rule base

4. The Use of a Four-Valued Logic System

4.1. The truth tables

As known, the original Boolean algebra is based on a two-valued logic, i.e., on *0* and *1*. These are called determined values as well. If a fact is true in the inference process, then its logic variable will have the value 1, if it is false then its value is 0. However, as far as the general algebraic treatment of rule bases is concerned, it requires more than these two values. It can be proved that the number of necessary and sufficient values is four [11], [12]. It means that in addition to 0 and 1, two more values are to be involved. These are as follows:

1) The indifferent or don't care logic value: *d.* It is interpreted in such a way that the network line which carries this value can take on either 0 or 1 freely, without influencing the computational results.
2) The unknown logic value: *u.* In this case we have not any knowledge about the concrete logic value (0, 1 or d) of the network line carrying u.

The treatment of the four values can be extended to the basic Boolean operations. This extension is summarized in the truth tables of Table 1, below:

AND	0	1	d	u
0	0	0	0	0
1	0	1	d	u
d	0	d	d	u
u	0	u	u	u

OR	0	1	d	u
0	0	1	d	u
1	1	1	1	1
d	d	1	d	u
u	u	1	u	u

NOT	
0	1
1	0
d	d
u	u

Table 1. Truth tables of the four-valued logic system

It should be remarked that there are other noteworthy logic systems with four values, e.g., those proposed in [13] and [14]. In these systems (and also others), 0, 1, "divergent" and "meaningless" are used. The basic deviation is that the values in the other systems are interpreted and applied for a differing purpose.

Next we are going to show how the given truth tables are used for forward and backward chaining. To reach this goal, consider the rule base above and the logic network belonging to it (see Figure 1). Let the initial set of facts be as follows:

$$T_0 = \{A, B, C, D, E, G, H\}.$$

4.2. The forward chaining procedure

In our representation, the forward chaining is performed in the following way:

Step 1: C = 1 and D = 1, since they both are in the fact base, which results in L = 1, so L is placed in the fact base.
Step 2: E = 1, because E is in the fact base, from which it follows that K = 0, but L = 1 alone implies P = 1, so P is placed in the fact base.
Step 3: The fact base does not contain M. In our logic system it can be interpreted as M = u. Though E = 1, due to the unknown value of M, this is not sufficient to imply Q = 1. It means that Q = u, thus Q cannot be placed in the fact base.

Here the final conclusion of the forward chaining was that fact P is true alone.

The above computational procedure can also be called forward tracing of the logic values. It means that we calculate the output values of the logic gates with knowledge of the gate-input values. As a matter of fact, this kind of tracing values is nothing else than logic simulation, which is a really fast process on computers.

4.3. The backward chaining procedure

The same way as before, the backward chaining procedure involves backward tracing of the logic values through the network. Now the input values of a gate have to be determined with knowledge of the actual gate-output value. In this case the goal is to justify that a primary output value is 1, i.e., a selected fact is true. It requires a successive decision process which is also called line-value justification [11], [12], [15]. As known, line-value justification is a procedure with the aim of successively assigning input values to the logic elements in such a way that they are consistent with each previously assigned value. (This concept is an auxiliary calculation process for justifying an initial set of logic values in a network, first applied in the so-called D-algorithm, for two-valued logic [16].)

The backward tracing of the logic values can also be performed in accordance with the four-valued truth table. However, this principle differs in some points from the forward tracing. In case of forward tracing, the output value of a gate is to be calculated with knowledge of the input values at the gate. If the inputs are given then they determine the output unambiguously. On the other hand, for a given output value at a gate not only one input combination can be assigned, there may be more than one possible choices. If two-input gates are considered, the possible choices are summarized in Figure 2 and Figure 3. If the number of inputs at a gate were more than two, it would increase the number of choices, but would not cause any difference in principle.

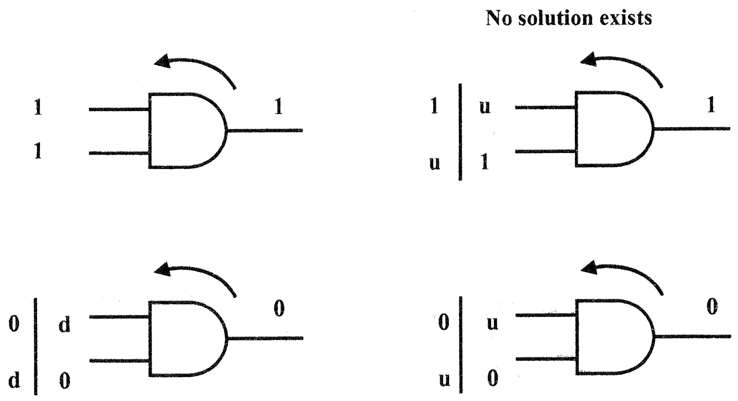


Figure 2. Backward tracing choices for an AND gate

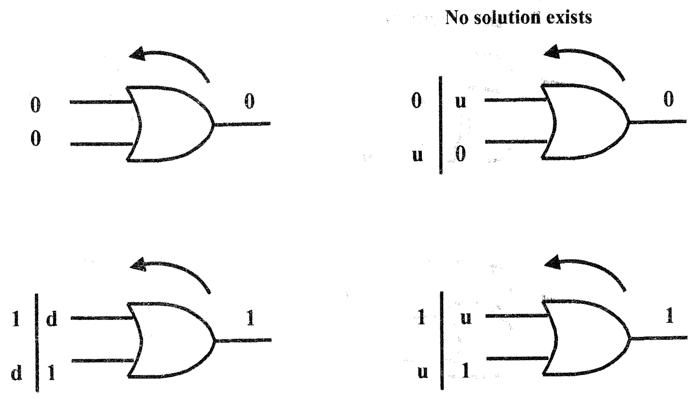


Figure 3. Backward tracing choices for an OR gate

When performing this process the following viewpoints have to be taken into consideration:

- Only the determined logic values, *0* and *1*, have to be traced back, i.e., these values are to be justified at the gate inputs. The value of *d* needs no justification, so it is unnecessary to trace it back. The output value of *u* is justified only by the input values of *u*.
- Since *d* does not require justification, it is worth assigning the minimum number of determined values to the gate inputs, while leaving the others at the value of *d*.
- In this logic system, the determined values and *u* are consistent only with *d*. This fact is to be taken into consideration when a network line has already a previously assigned value, and another value is required at the same line. Whenever a contradiction, i.e., inconsistency occurs, we have to make a new choice or change the last possible decision.

In our example (Figure 1), the computations proceed as follows:

Step 1: The proof of **P = 1:** At first let K = 1 and L = d, which are the minimally necessary assignments. Now from K = 1 it follows that E = 0, which is a contradiction, for E is in the fact base, so E = 1 holds.

Step 2: We have to modify our previous decision: Now let L = 1 and K = d. In this way L = 1 is justified by C = 1 and D = 1, without any contradiction.

Step 3: It is unnecessary to trace back the value K = d, since the indifferent value does not require justification. So the proof of P has been finished.

Step 4: The proof of **Q = 1:** This condition requires that both inputs to the AND gate be 1, i.e., E = 1 and M = 1. Since E is a member of the fact base, E = 1 holds. However, M is missing from the fact base, which means that M = u. In this case it is impossible to justify (prove) that Q = 1.

5. Concluding Remarks

This paper has presented a logic model which is directly applicable for inference chains in expert systems. Both forward and backward reasoning can be performed on the base of the model. In comparison with the conventionally organized knowledge bases, the calculations using this four-valued logic can advantageously be organized and carried out in embedded computing systems due to the following reasons:

- The storage requirement of the four logic values at the network lines is negligible: only two bits are necessary and sufficient for coding them.
- Computations among logic values are *ab ovo* fast and efficient. This fact manifests itself especially when bit-level implementation is applied.
- The data-base structure of a logic network is comparatively simple, and requires minimal memory space. Only the gate types and the input-output connections of the gates are to be encoded and stored. The forward and backward tracing are carried out directly on this network structure.

References

[1] Massa, A. J.: *Embedded Software Development with eCos*, Prentice-Hall, Inc., USA (2003)

[2] Storey, N.: *Safety-Critical Computer Systems*, Addison-Wesley-Longman, Inc., New York, (1996)

[3] Waterman, D. A.: *A Guide to Expert Systems*, Addison-Wesley Publishing Company, USA (1986)

[4] Negnevitsky, M.: *Artificial Intelligence: A Guide to Intelligent Systems*, Addison-Wesley Publishing Company, Great Britain (2002)

[5] Chen, Zh.: *Computational Intelligence for Decision Support*, CRC Press LLC, USA (2000)

[6] Sziray, J.: *The Basic Principles of Expert Systems*, (In Hungarian), Universitas Ltd., Győr (2006)

[7] Lewis, H. R., Papadimitriou, Ch. H.: *Elements of the Theory of Computation*, Prentice-Hall, Inc., USA (1998)

[8] Hopcroft, J. E., Motwani, R.,. Ullman, J. D: *Introduction to Automata Theory, Languages, and Computation,* Second Edition, Addison-Wesley Publishing Company, USA (2001)

[9] Cormen, T. H., Leiserson, Ch. E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, McGraw-Hill Publishing Company, USA (2001)

[10] Uyemura, J. P.: *A First Course in Digital Systems Design: An Integrated Approach*, Brooks-Cole Publishing Company, USA (2000)

[11] Abramovici, M., Breuer, M. A., Friedman, A. D.: *Digital Systems Testing and Testable Design*, Computer Science Press, USA (1990)

[12] Sziray, J.: *Test Calculation for Logic Networks by Composite Justification*, Digital Processes, Vol. 5, No. 1-2, Great Britain (1979) pp. 3-15

[13] Belnap Jr., N. D.: *A Useful Four-Valued Logic*, *Modern Uses of Multiple-Valued Logic*, Fifth International Symposium, Indiana University, Bloomington, Ind., USA (1975) pp. 5-37

[14] Tarasov, V. E.: *Quantum Computer with Mixed States and Four-Valued Logic*, J. Phys. A. Math. Gen. 35, (2002) pp. 5207-5235

[15] Sziray, J.: *Test Calculation for Logic and Timing Faults*, IEEE International Workshop on RTL and High Level Testing, (WRTLT-2004), Osaka, Japan, (2004) pp. 45-50

[16] Roth, J. P.: *Diagnosis of Automata Failures: a Calculation and a Method*, IBM Journal of Research and Development, Vol. 10, USA, (1966) pp. 278-291

Analyzing Fuzzy Flip-Flops Based on Various Fuzzy Operations

Rita Lovassy [1,2], László T. Kóczy [1,3], and László Gál[1,4]

[1] **Faculty of Engineering Sciences, Széchenyi István University**
9026, Győr, Egyetem tér 1. Hungary
Email: lovassy.rita@kvk.bmf.hu, koczy@sze.hu, gallaci@ttmk.nyme.hu

[2] **Inst. of Microelectronics and Technology,**
Kandó Kálmán Faculty of Electrical Engineering, Budapest Tech,
Budapest, Hungary
Email: lovassy.rita@kvk.bmf.hu

[3] **Dept. of Telecommunication and Media Informatics,**
Budapest University of Technology and Economics,
H-1117 Budapest, Magyar tudósok krt. 2. Hungary
Email: koczy@tmit.bme.hu

[4] **Department of Technology, Informatics and Economy**
University of West Hungary
H-9700, Szombathely, Károlyi G. tér 4. Hungary
e-mail: gallaci@ttmk.nyme.hu

Abstract: This paper concerns the role that fuzzy operations play in the study of behavior of fuzzy J-K and D flip-flops (F^3). We define various types of F^{3}s based on well known operators, presenting their characteristic equations, illustrating their behavior by their respective graphs belonging to various typical values of parameters. Connecting the inputs of the fuzzy J-K flip-flop in a particular way, namely, by applying an additional inverter in the connection of the input J to K ($K=1-J$), a fuzzy D flip-flop is obtained. When input K is connected to the complemented output ($K=1-Q$), or in the case of $K=1-J$, the J-$Q(t+1)$characteristics of the F^{3}s derived from the Yager, Dombi, Hamacher, Frank, Dubois-Prade and Fodor t-norms present more or less sigmoidal behavior. Two different interpretations of fuzzy D flip-flops are also presented. We pointed out the strong influence of the idempotence axiom in D F^3's behavior. A method for constructing Multilayer Perceptron Neural Networks (MLP NN) with the aid of fuzzy systems, particularly by deploying fuzzy flip-flops as neurons is proposed.

Keywords: t-norm, t-conorm, fuzzy J-K flip-flop, fuzzy D flip-flop, Multilayer Perceptron (MLP) constructed from F^3 neurons, Fuzzy Flip-Flop Neural Network (FNN)

1. Introduction

In fuzzy set theory the study of triangular operators has been going on for a long time. The three basic crisp (Boolean) logical operations (namely, complementation – negation, intersection – conjunction and union - disjunction), well-defined on traditional sets, can be generalized in many ways using fuzzy sets. When Zadeh first introduced the concept of fuzzy sets [15] he proposed operators for set complement, intersection and union. These operators have been referred to classic, standard or Zadeh-type ones. The generalized class of intersections was shown to satisfy the axiomatic properties of t-norms, while unions were proven to be t-conorms (s-norms).

The fuzzy literature offers a large variety of triangular operators; researchers still propose again and again new fuzzy operations to be used in a given field. Obviously, the performance of fuzzy systems depends from the choice of different triangular operators. Despite the variety of available fuzzy set operators, however, the classic triple of complement, intersection and union still bear particular significance, especially in the practical applications. The big challenge for fuzzy researchers is to fit the fuzzy sets into the context of applications.

The paper is structured into five sections. After the introduction, in Section 2, we present the concept of a single fuzzy J-K flip-flop, using the fundamental equation as it was proposed in [12].

In Section 3, a comparative study of several types of fuzzy J-K flip-flops based on various norms has been made and it was shown that, broadly, they may be classified into two types, one of which present quasi S-shape *J-Q(t+1)* characteristics and the rest with non-sigmoidal character. Comparison between fuzzy J-K flip-flop with feedback, fuzzy D flip-flop and a different interpretation to define fuzzy D flip-flop is presented. Section 4 is devoted to the investigation of the F^3 based neurons and the Multilayer Perceptrons (MLP) [10] constructed from them. We proposed the Fuzzy Flip-Flop Neural Network (FNN) architecture. We show that it can be use for approximating various simple transcendental functions, such as a simple sine wave, a complex trigonometric function with one variable, another trigonometric function with two variables, a rational function with two variables, and a benchmark pH problem model. Comparison between different types of FNNs and the target *tansig* (hyperbolic tangent sigmoid transfer function) characteristics NN are presented in Section 5.

2. The Concept of Fuzzy J-K Flip-Flop

The fuzzy J-K flip-flop is an extended form of binary J-K flip-flop. In this approach the truth table for the J-K flip-flop is fuzzified, where the binary NOT, AND and OR operations are substituted by their fuzzy counterparts, i.e. fuzzy negation, t-norm, and co-norm respectively. The next state *Q(t+1)* of a J-K flip-flop is characterized as a function of both the present state *Q(t)* and the two present inputs *J(t)* and *K(t)*. For simplicity *(t)* is omitted in the next. The so called fundamental equation of J-K type fuzzy flip-flop [12] is

$$Q(t+1) = (J \vee \neg K) \wedge (J \vee Q) \wedge (\neg K \vee \neg Q) \tag{1}$$

where $\neg, \wedge, \vee$ denote fuzzy operations (e.g. $\neg K = 1 - K$). As a matter of course, it is possible to substitute the standard operations by any other reasonable fuzzy operation triplet (e.g. De-Morgan triplet), thus obtaining a multitude of various fuzzy flip-flop (F^3) pairs.

In [9] we studied the behavior of F^3 based on various fuzzy operations.

In the next Section we will give an overview of the different type J-K F^3s, based on familiar norms well known from the literature, namely the standard (min-max), Yager, Dombi, Hamacher (including algebraic), Frank, Dubois – Prade and Schweizer – Sklar ones, using the standard complementation in every case. After introducing their characteristic equations we will illustrate their behavior by the graphs belonging to the next states of fuzzy flip-flops for typical values of *Q, J* and *K*.

Some researchers tried to relax the constraint of associativity for fuzzy connectives. In [3] a pair of non-associative (non-dual) operations for a new class of fuzzy flip-flops was proposed. The behavior of the "Fodor type fuzzy flip-flop" developed from a modified version of the operations was also evaluated for comparison.

3. J-K F^3s Based on Various Fuzzy Connectives

This section provides a comprehensive overview of the behavior of fuzzy J-K flip-flops based on various fuzzy operations. A set of ten t-norms, combined with the standard negation, was analyzed to investigate, whether and to what degree they present more or less sigmoidal (S-shaped) *J-Q(t+1)* characteristics in particular cases, when *K=1-Q*, *K=1-J*, with fixed value of *Q*. Only a few t-norm pairs present non-sigmoidal behavior, with piecewise linear characteristics and several breakpoints. One of them (the algebraic norm pair) having the advantage of the hardware implementation of F^3. Circuits based on algebraic norms are presented earlier in [11]. The implementation was done by using fuzzy gate circuits.

3.1. Fuzzy J-K Flip-Flops Based on Some Classes of Fuzzy Set Unions and Intersections

In his very first paper on fuzzy sets Zadeh proposed the standard (min-max) t-operators on fuzzy sets [15]. Using standard negation (2), they are as follows:

$$c(a) = 1 - a \tag{2}$$

$$i_S(a,b) = \min(a,b) \tag{3}$$

$$u_S(a,b) = \max(a,b) \tag{4}$$

In this case, equation (1) can be expressed as

$$Q(t+1)=\min(\max(J,(1-K)),\max(J,Q),\max((1-K),(1-Q))) \quad (5)$$

the characteristical equation of the standard type fuzzy J-K flip-flop.

Using the algebraic norms

$$i_A(a,b)=ab \quad (6)$$

$$u_A(a,b)=a+b-ab \quad (7)$$

The fundamental equation of the algebraic type fuzzy flip-flop [11] can be rewritten in the form

$$Q(t+1)=J+Q-JQ-KQ \quad (8)$$

Lukasiewicz norms [7] and the corresponding *Q(t+1)* definition is presented

$$i_L(a,b)=\max(a+b-1,0) \text{ and } u_L(a,b)=\min(a+b,1) \quad (9)$$

$$Q(t+1)=\max(\min(J+(1-K),1)+\min(J+Q,1)+\min((1-K)+(1-Q),1)-1,0) \quad (10)$$

Min and max are often selected as the t-norm/s-norm pair. This choice is mainly due to the simplicity of the calculations.

Yager, in [14], proposed an infinite family of possible fuzzy operation pairs. The intersection of two fuzzy sets *a* and *b* applying the Yager t-norm has the expression

$$i_w(a,b)=1-\min\left[1,((1-a)^w+(1-b)^w)^{1/w}\right] \text{ for } a,b\in[0,1] \quad (11)$$

where values of parameter *w* lie within the open interval $(0,\infty)$. By the way for $w=1$ it gives the Łukasiewicz t-norm. For simplicity we use the following denotation $i_w(a,b)=a\ \ i_w\ b$.

The dual expression of t-conorm is defined by

$$u_w(a,b)=\min\left[1,(a^w+b^w)^{1/w}\right], \quad (12)$$

for *w* as before. Similarly to (12) $u_w(a,b)=a\ \ u_w\ b$.

Using such as triplet, the maxterm form in the unified equation (1) can be rewritten as

$$Q(t+1)=\left(J\ u_w\left(1-K\right)\right)\ i_w\ \left(J\ u_w\ Q\right)\ i_w\ \left(\left(1-K\right)\ u_w\left(1-Q\right)\right) \tag{13}$$

Several values of parameter w in the Yager-norm were considered, in an effort to tune the *J-Q(t+1)* characteristics of the corresponding F^3.

The pair of Dombi-class operators (similarly, a De-Morgan triplet) are defined as follows [7]:

$$i_\alpha(a,b)=\frac{1}{1+\left[\left(1/a-1\right)^\alpha+\left(1/b-1\right)^\alpha\right]^{1/\alpha}};u_\alpha(a,b)=\frac{1}{1+\left[\left(1/a-1\right)^{-\alpha}+\left(1/b-1\right)^{-\alpha}\right]^{-1/\alpha}} \tag{14}$$

where $i_\alpha(a,b)=a\ \ i_\alpha\ \ b$ and $u_\alpha(a,b)=a\ \ u_\alpha\ \ b$.

The unified equation of the next state can be expressed as

$$Q(t+1)=\left(J\ u_\alpha\left(1-K\right)\right)\ i_\alpha\left(J\ u_\alpha\ Q\right)\ i_\alpha\ \left(\left(1-K\right)\ u_\alpha\left(1-Q\right)\right) \tag{15}$$

Parameter α lies within the open interval $(0,\infty)$. If $J=0$, $K=0$ or $Q=0$ (14) results in division by 0 the expressions are extended to their respective limit values. Both the Yager and the Dombi operators are classic (monotonic, commutative, associative and limit preserving) t-norms and co-norms. The character of standard, algebraic, Yager and Dombi t-norms for a selected parameter value is illustrated in Figures 1-4.

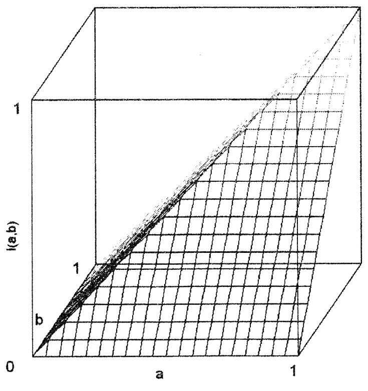


Figure 1. Standard t-norm

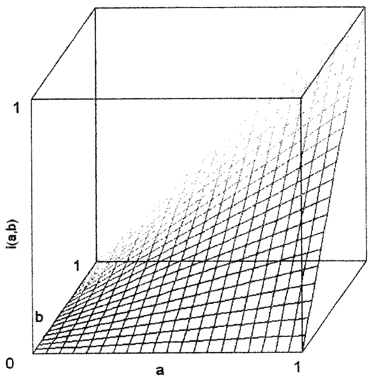


Figure 2. Algebraic t-norm

Hamacher t-norms are the following [7] for $v \in (0,\infty)$

$$i_H(a,b) = \frac{ab}{v+(1-v)(a+b-ab)}, \text{ and } \quad u_H(a,b) = \frac{a+b-(2-v)ab}{1-(1-v)ab} \tag{16}$$

The definition of Frank operators [7] for $s \in (0,\infty)$

$$i_F(a,b) = \log_s\left[1+\frac{(s^a-1)(s^b-1)}{s-1}\right]; \; u_F(a,b) = 1-\log_s\left[1+\frac{(s^{1-a}-1)(s^{1-b}-1)}{s-1}\right] \tag{17}$$

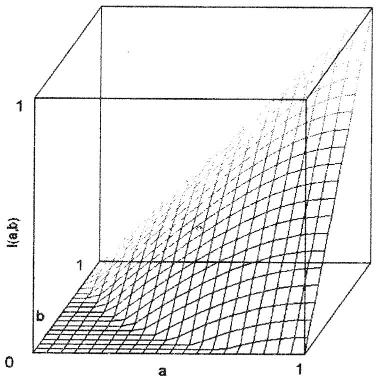


Figure 3. Yager t-norm w=2

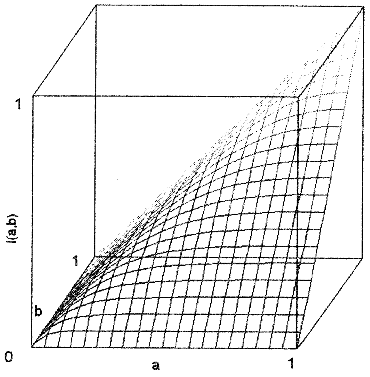


Figure 4. Dombi t-norm α = 2

The Dubois and Prade [10] operators are for $d \in (0,1)$

$$i_{D-P}(a,b) = \frac{ab}{\max(a,b,d)}; \; u_{D-P}(a,b) = \frac{a+b-ab-\min(a,b,1-d)}{\max(1-a,1-b,d)} \tag{18}$$

Schweizer and Sklar investigated the following class of t-operators [13]

$$i_{S-S}(a,b) = \max(0,a^{-p}+b^{-p}-1)^{-1/p}; u_{S-S}(a,b) = 1-\max(0,(1-a)^{-p}+(1-b)^{-p}-1)^{1/p}$$
$$p \in (-\infty,\infty) \tag{19}$$

In [3] a pair of non-associative (non-dual) operations for a new class of fuzzy flip-flops was proposed. The operations proposed combined the standard and Łukasiewicz norms by the arithmetic mean and resulted into the following operations:

$$i_F(a,b) = \frac{i_L(a,b)+i_S(a,b)}{2} \text{ and } u_F(a,b) = \frac{u_L(a,b)+u_S(a,b)}{2} \tag{20}$$

In a similar way, using the respective expressions for fuzzy set intersection and union, we give in [8] the unified equation of the J-K F^3 in this case.

We will refer to this new type of F^3 as Fodor type fuzzy flip-flops (because of the first author J. Fodor) by F^4.

$$Q(t+1) = u_L\left[\frac{i_L+i_S}{2}(J,1-Q),\frac{i_L+i_S}{2}(1-K,Q)\right] = i_L\left[\frac{u_L+u_S}{2}(J,Q),\frac{u_L+u_S}{2}(1-K,1-Q)\right] \quad (21)$$

3.2. Fuzzy J-K flip-flop, K=1-Q (fuzzy J-K flip-flop with feedback)

The next figures depict the behavior by the graphs belonging to the next states of different type fuzzy J-K flip-flops for various typical values of *Q*, *J* and *K*, in the particular case, when *K=1-Q*. Figure 5 and Figure 6 bring examples for non-sigmoidal F^3s (min-max and algebraic types). Using the parameterized families of Yager, Dombi, Hamacher, Frank, and Dubois-Prade norms for typical parameter values, we obtained more or less S-shaped *J-Q(t+1)* characteristics. The 2D figures and the sections of the 3D surface are approximately sigmoidal as it is shown in Figures 7-12. Figures 13 and 14 depict the behavior of the fuzzy J-K flip-flop based on the Schweizer-Sklar and Fodor norms. There are obviously several breakpoints and lines in the surface because of the max and min operations in the formulas. Nevertheless the sections of the surface in F^4 are again more or less sigmoidal as it is shown for some typical values of *Q* and *J*.

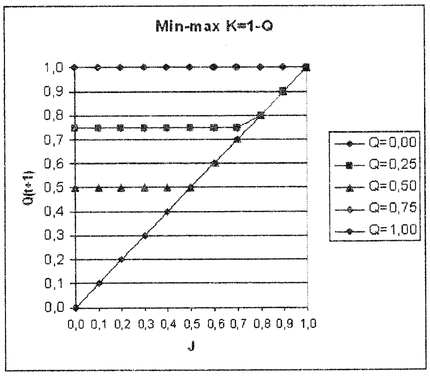


Figure 5
J-Q(t+1)characteristics of standard type F^3

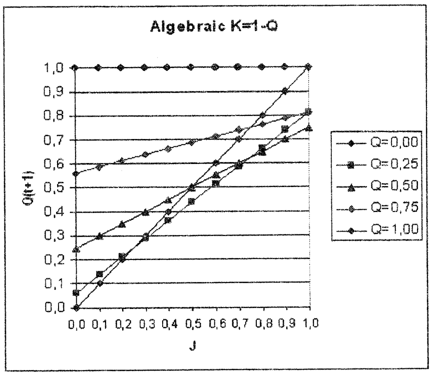


Figure 6
J-Q(t+1)characteristics of algebraic type F^3

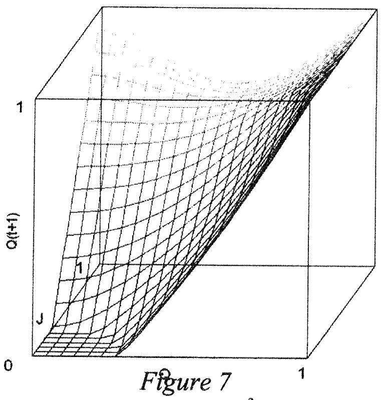


Figure 7
Yager type J-K F^3 α=2

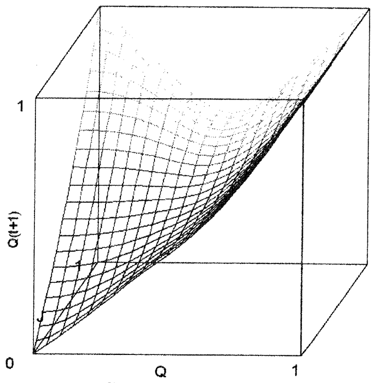


Figure 8
Dombi type J-K F^3 α=2

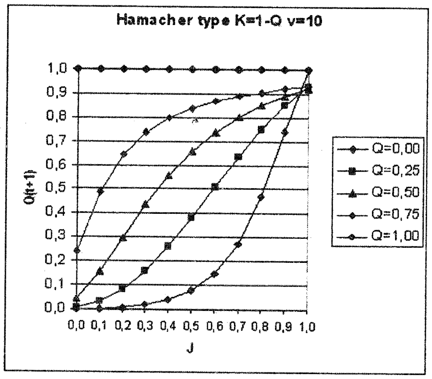


Figure 9
J-Q(t+1)characteristics of Hamacher type F^3

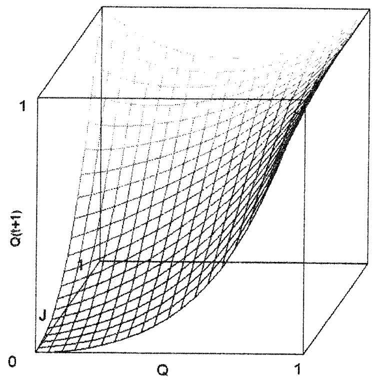


Figure 10
Hamacher type J-K F^3

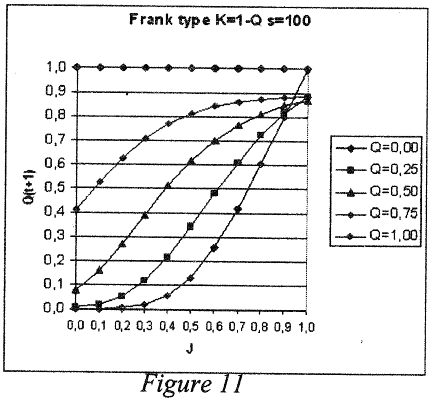


Figure 11
J-Q(t+1)characteristics of Frank type F^3

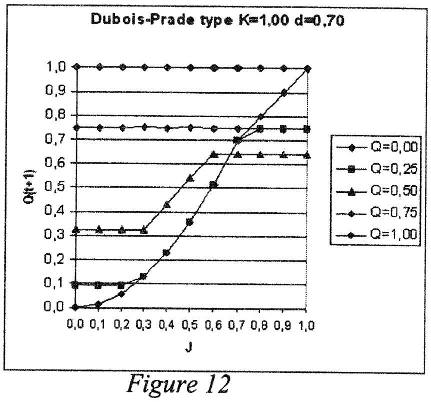


Figure 12
J-Q(t+1)characteristics of Dubois-Prade type F^3

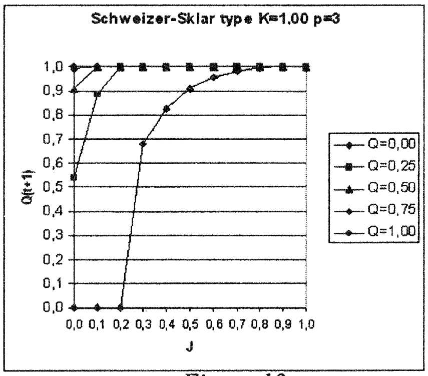


Figure 13
J-Q(t+1)characteristics of Schweizer-Sklar type F^3

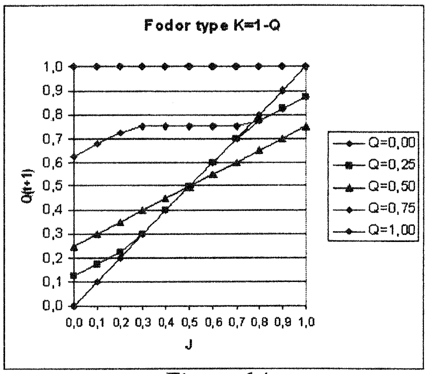


Figure 14
J-Q(t+1)characteristics of Fodor type F^3

3.3. Fuzzy J-K flip-flop, K=1-J (fuzzy D flip-flop)

Connecting the inputs of the fuzzy J-K flip-flop in a particular way, namely, by applying an inverter in the connection of the input J to K, case of $K=1-J$, a fuzzy D flip-flop is obtained. Substituting $\neg K = J$ in equation (1) and let $D=J$, the fundamental equation of fuzzy D flip-flop will be

$$Q(t+1) = (D \vee D) \wedge (D \vee Q) \wedge (D \vee \neg Q) \tag{22}$$

Figures 15-18 show the behavior of the fuzzy D flip-flop introduced above, substituting to equation (22) the standard, algebraic, Yager and Dombi norms. For a well selected parameter values (i.e. w=2 and α=2) and Q value, the J-$Q(t+1)$ characteristics present nice quasi sigmoidal behavior.

As an alternative approach, Choi and Tipnis [1] proposed an equation which exhibits the characteristics of a fuzzy D flip-flop, as follows

$$Q(t+1) = (D) \wedge (D \vee Q) \wedge (\neg Q \vee D) \tag{23}$$

We will refer to this new type of fuzzy D flip-flop as Choi type fuzzy D flip-flop (because of the first author B. Choi). Comparing the characteristical equation of the fuzzy D flip-flop (22), with expression (23), there is an essential difference between the two fuzzy flip-flops. Substituting $D=J=1-K$, the two formulas differ in the first member.

$D = D \vee D$ holds only in the exceptional case, when the t-conorm is idempotent. Idempotence for T and S means that [2]

$T(x,x) = x$ and $S(x,x) = x$ for all $x \in [0,1]$;

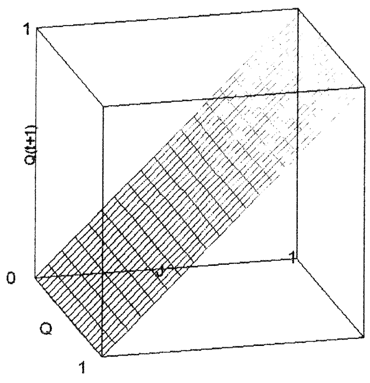


Figure 15
Standard type D F^3

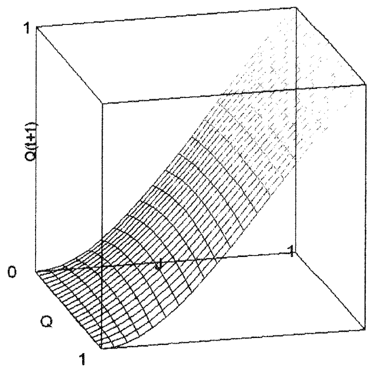


Figure 16
Algebraic type D F^3

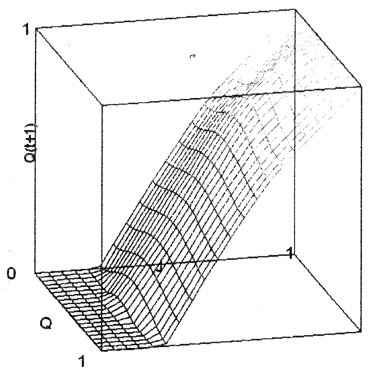


Figure 17
Yager type D F^3

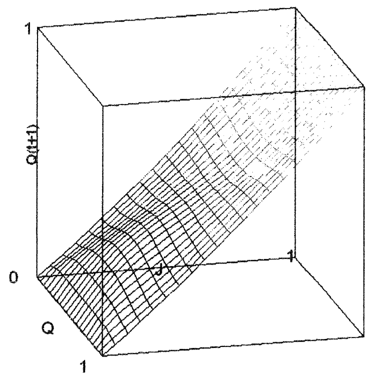


Figure 18
Dombi type D F^3

It can be proved [4] that t-norm T is idempotent iff T = min, and t-conorm S is idempotent iff S = max.
For example, using the algebraic norm

$$u_A(a,a) = a + a - a \cdot a = 2 \cdot a - a^2 = a \tag{24}$$

is true only in the borderline cases, i.e. when $a = 0$, or $a = 1$. It is surprising how much the satisfaction of idempotence influences the behavior of the fuzzy D flip-flops.

Although, the *J-Q(t+1)* Choi fuzzy D flip-flop characteristics for standard, algebraic, Yager and Dombi norms (Figures 19-22) also present approximately sigmoidal behavior. Comparing Figures 15-18 and 19-22 belonging to the two types of fuzzy D flip-flop with the same norms, it can be seen that, for the same value of Q, the curvature differs, which fact leads to a rather different behavior in the applications.

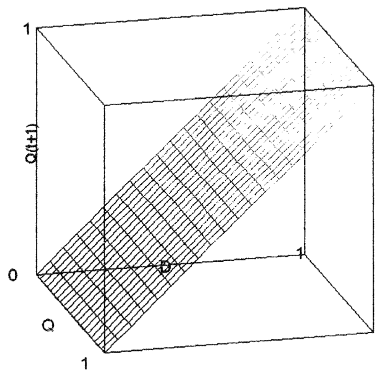


Figure 19
Standard type Choi D F^3

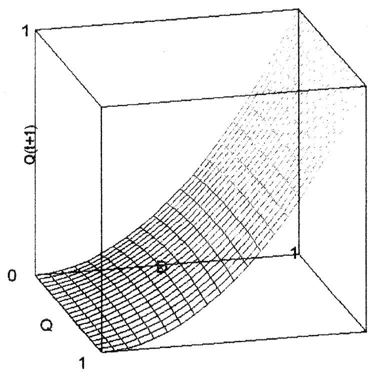


Figure 20
Algebraic type Choi D F^3

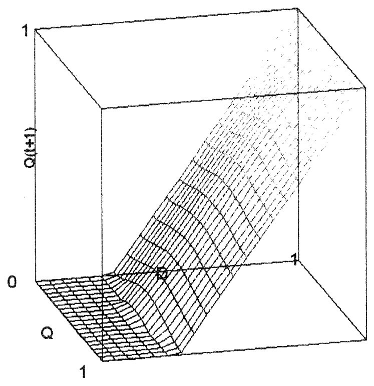


Figure 21
Yager type Choi D F^3

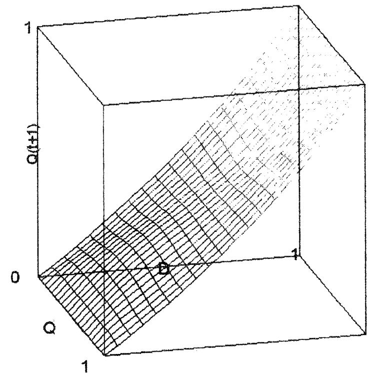


Figure 22
Dombi type Choi D F^3

4. The Fuzzy Flip-Flop Based Neurons

Next, a fuzzy network is proposed, in which an artificial neural network-like approach is designed to construct the knowledge base of an expert system.

We study the effect of applying some well know t-norms in the investigation of the F^3 based neurons and the MLPs constructed from them. An interesting aspect of these F^3s is that they have a certain convergent behavior when their input *J* is excited repeatedly. This convergent behavior guarantees the learning property of the networks constructed this way.

In our approach the weighted input values are connected to input J of the fuzzy flip-flop based on a pair of t-norm and t-conorm, having quasi-sigmoidal transfer characteristics.

The output signal is then computed as the weighted sum of the input signals, transformed by the transfer function [5].

In this concept, K=1-Q (feedback J-K F^3), or K=1-J (D F^3) is proposed. When input K of the F^3 is connected with output $\overline{Q}$, or when input K is connected with J, an elementary fuzzy sequential unit with just one input is obtained. Now J can be considered as an equivalent of the traditional input of the neuron. The behavior of Choi type fuzzy D flip-flop was also evaluated for comparison.

From the neural networks perspective (regarding to the ability to use the learning and adaptation mechanisms used with classic neuron models), suitable t-norms may be deployable for defining fuzzy neurons.

4.1. Fuzzy Flip-Flop Network

A very commonly used architecture of neural network is the multilayer feed forward network, which allows signals to flow from the input units to the output units, in a forward direction. In general, two trainable layer networks with sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer are universal approximators [6].

The model for the neural system now proposed is based on two hidden layers constituted from fuzzy flip-flop neurons. Networks now proposed are sensitive to the number of neurons in their hidden layers. Too few neurons can lead to underfitting, too many neurons can cause similarly undesired overfitting. The functions to be approximated are represented by a set of input/output pairs. All the input and output signals are distributed in the unit interval. During network training, the weights and thresholds are first initialized to small, random values.

5. Function Approximation by Multilayer Networks

5.1. Single sine wave (various norms)

A fuzzy flip-flop based neural network, with a transfer function using algebraic, Yager, Dombi and Fodor operators in the hidden layers furthermore a linear transfer function in the output layer, was used to approximate a single period of the sine wave. The number of neurons was chosen after experimenting with different size hidden layers. Smaller neuron numbers in the hidden layer result in worse approximation properties, while increasing the neuron number results in better performance, but longer simulation time. The training was performed for different size hidden layers and finally a 1-4-4-1 FNN was proposed as good and fast enough.

Different random initial weights were used and the network was trained with Levenberg-Marquardt algorithm with 100 maximum numbers of epochs as more or less sufficient.

In our present experiments we forced $Q = 0.32$, because this value ensured rather good learning abilities. We suppose however that flexible Q values might lead to even better learning and approximation properties in the future.

The expression of the function to be approximated was:

$$y = \sin(c_1 * x)/2 + 0.5, \tag{25}$$

where the input vector x generated a sinusoidal output y. The value of constant c_1 was chosen 0.07, to keep the wavelet in the unit interval. The parameter of Dombi operators was fixed $\alpha=2$ while the Yager F^3 was set $w=2$. Both fixed parameter values provided good learning and convergent properties. Figure 23 presents the graphs of the simulations for fuzzy J-K flip-flop based neural network. It can be observed that the algebraic F^3 provides a fuzzy neuron with rather bad learning ability. The Fodor F^3 is much better but it is still clearly deviating from the target function.

Table 1 summarizes the 100 runs average approximation goodness, by indicating the Mean Squared Error (MSE) of the training and validation values for each of the *tansig*, algebraic, Yager, Dombi and Fodor types of FNNs. Comparing the minimum, median (median value of the array), mean and standard deviation (StdDev) values, the Dombi and Yager type neural networks performed best, thus they can be considered as rather good function approximators. The worst results were produced obviously by the networks based on Fodor operators and especially algebraic F^3s. Although, Fodor fuzzy flip-flops presented favorable mathematical properties, they had not very good approximation behavior.

Figure 24 compares the behavior of the fuzzy J-K flip-flop with feedback, fuzzy D flip-flop and Choi type fuzzy D flip-flop based NN. The fuzzy flip-flops are based on algebraic norms. Table 2 concludes the different MSE of the training and validation values regarding to the above mentioned cases, complete with the target values.

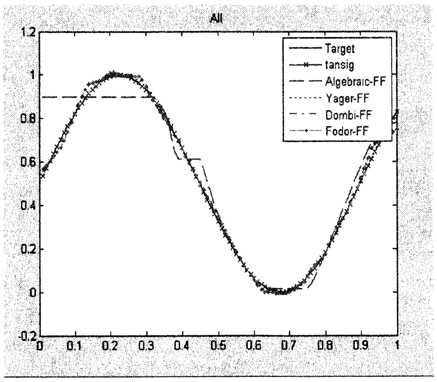


Figure 23
Simulation results

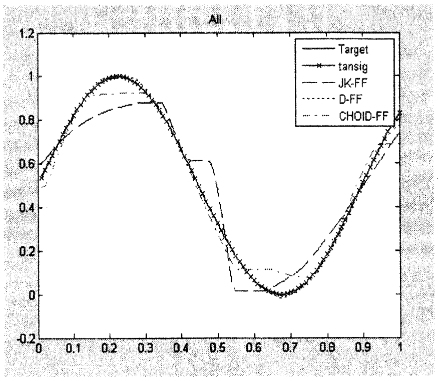


Figure 24
Simulation results

TABLE 1. Single sine wave

F^3 Neuron Type	MSE Training				MSE Validation			
	Min	Median	Mean	StdDev.	Min	Median	Mean	StdDev.
tansig (target)	9.3×10^{-14}	3.7×10^{-8}	5.4×10^{-3}	2.1×10^{-2}	9.2×10^{-14}	1.7×10^{-7}	5.2×10^{-3}	2.1×10^{-2}
Algebraic	9.1×10^{-3}	5.2×10^{-2}	5.8×10^{-2}	2.7×10^{-2}	8.9×10^{-3}	5.9×10^{-2}	6.4×10^{-2}	2.8×10^{-2}
Yager	6.7×10^{-7}	3.7×10^{-2}	5.6×10^{-2}	5.1×10^{-2}	9.2×10^{-7}	4.1×10^{-2}	5.7×10^{-2}	5.3×10^{-2}
Dombi	2.1×10^{-8}	4.6×10^{-2}	6.5×10^{-2}	5.1×10^{-2}	4.7×10^{-8}	6.2×10^{-2}	6.5×10^{-2}	4.9×10^{-2}
Fodor	9.7×10^{-4}	4.1×10^{-2}	5.4×10^{-2}	3.7×10^{-2}	6.6×10^{-4}	4.7×10^{-2}	6.0×10^{-2}	4.1×10^{-2}

TABLE 2. Single sine wave

F^3 Neuron Type	MSE Training				MSE Validation			
	Min	Median	Mean	StdDev.	Min	Median	Mean	StdDev.
tansig (target)	1.2×10^{-9}	2.8×10^{-8}	1.2×10^{-3}	5.6×10^{-3}	3.0×10^{-9}	1.1×10^{-7}	2.2×10^{-3}	1.5×10^{-2}
JK-FF	9.2×10^{-3}	6.5×10^{-2}	7.6×10^{-2}	3.9×10^{-2}	4.8×10^{-3}	6.6×10^{-2}	7.8×10^{-2}	4.9×10^{-2}
D-FF	6.6×10^{-5}	1.2×10^{-2}	2.8×10^{-2}	3.3×10^{-2}	1.2×10^{-4}	1.5×10^{-2}	3.8×10^{-2}	3.6×10^{-2}
CHOID-FF	5.3×10^{-3}	5.1×10^{-2}	6.1×10^{-2}	3.4×10^{-2}	4.8×10^{-3}	5.3×10^{-2}	6.3×10^{-2}	4.1×10^{-2}

5.2. Two superimposed sine waves with different period lengths (various norms)

When instead of a single sine wave a more complex wave form was used, in order to obtain the same results we increased the neuron numbers in the hidden layers to 8 neurons in each. We proposed a 1-8-8-1 F^3 based neural network to approximate a combination of two sine wave forms with different period lengths described with the equation

$$y = \sin(c_1 * x) * \sin(c_2 * x)/2 + 0.5. \tag{26}$$

The values of constants c_1 and c_2 were selected to produce a frequency proportion of the two components 1:0.35. Same as in subsection 5.1 we compared the network function approximation capability in the above mentioned cases as is shown in Figures 25, 26.

It is interesting that according to the numerical illustrations, the average of 100 runs mean squared error of training and validation values (Tables 3, 4), the sequence is again the same as it was in the case of the single sine wave. Our hypothesis is that among these four Dombi neuron is the best and the Yager neuron is not much worse.

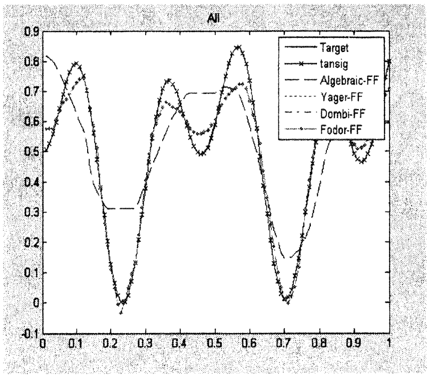


Figure 25
Simulation results

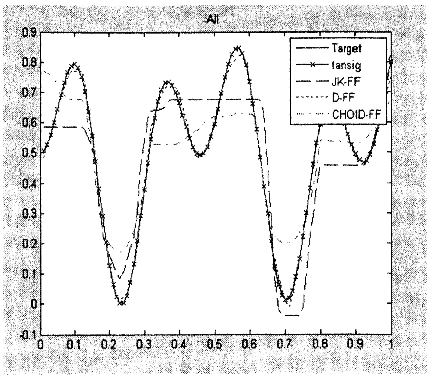


Figure 26
Simulation results

TABLE 3. Two sine waves

F^3 Neuron Type	MSE Training				MSE Validation			
	Min	Median	Mean	StdDev.	Min	Median	Mean	StdDev.
tansig (target)	6.8x10^{-8}	1.6x10^{-6}	1.2x10^{-4}	3.9x10^{-4}	3.2x10^{-7}	6.2x10^{-5}	6.3x10^{-4}	2.2x10^{-3}
Algebraic	2.1x10^{-2}	4.6x10^{-2}	4.8x10^{-2}	1.8x10^{-2}	1.9x10^{-2}	5.2x10^{-2}	5.3x10^{-2}	2.3x10^{-2}
Yager	3.8x10^{-6}	6.1x10^{-3}	2.1x10^{-2}	2.2x10^{-2}	4.9x10^{-5}	8.4x10^{-3}	2.4x10^{-2}	2.6x10^{-2}
Dombi	2.3x10^{-7}	3.4x10^{-2}	3.1x10^{-2}	2.3x10^{-2}	2.2x10^{-6}	3.2x10^{-2}	3.2x10^{-2}	2.5x10^{-2}
Fodor	4.1x10^{-3}	4.2x10^{-2}	4.2x10^{-2}	2.6x10^{-2}	6.2x10^{-3}	4.2x10^{-2}	4.4x10^{-2}	2.5x10^{-2}

TABLE 4. Two sine waves

F^3 Neuron Type	MSE Training				MSE Validation			
	Min	Median	Mean	StdDev.	Min	Median	Mean	StdDev.
tansig (target)	1.3910^{-8}	1.6x10^{-6}	5.7x10^{-4}	3.4x10^{-3}	4.8x10^{-7}	2.9x10^{-5}	1.4x10^{-3}	4.1x10^{-3}
JK-FF	1.1x10^{-2}	4.7x10^{-2}	4.9x10^{-2}	1.7x10^{-2}	1.7x10^{-2}	5.1x10^{-2}	5.5x10^{-2}	2.2x10^{-2}
D-FF	1.7x10^{-4}	1.5x10^{-2}	2.3x10^{-2}	2.1x10^{-2}	2.1x10^{-4}	1.5x10^{-2}	2.8x10^{-2}	3.4x10^{-2}
CHOID-FF	1.3x10^{-2}	5.3x10^{-2}	5.1x10^{-2}	1.9x10^{-2}	1.5x10^{-2}	5.3x10^{-2}	5.6x10^{-2}	2.5x10^{-2}

5.3. Two - input trigonometrical function

From another point of view, we compare the Yager type FNN with the Dombi one, whose characteristic equation was mentioned in Section 2. We approximated the following two dimensional function composed from sin and cos components, using a 1-20-20-1 feedforward neural network structure.

$$y = \sin(c_1 * x_1)^5 * \cos(c_2 * x_2)^3 /2+0.5. \tag{27}$$

The 3D scenes using Yager and Dombi operators are depicted in Figure 27 and 28 respectively. The parameter of Dombi operators was fixed α=2 and the Yager F^3 parameter was set to w=2. Both fixed parameter values provided good learning and convergence properties. In particular the parameters of the family of Yager and Dombi norms were optimized for this purpose.

Comparing the minimum mean squared errors, the Dombi and Yager type neural networks can be considered as rather good function approximators. The MSEs appearing at the top of the graphs are instantaneous values, illustrating very well the 20 runs average approximation goodness.

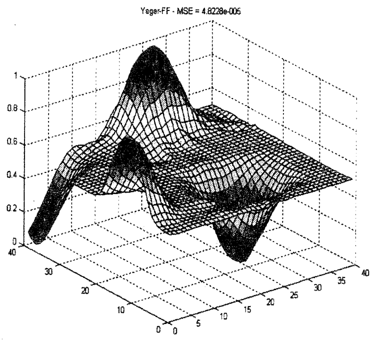


Figure 27
Simulation results

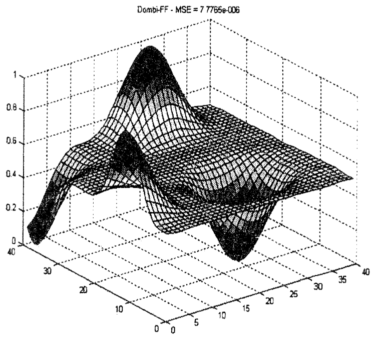


Figure 28
Simulation results

5.4. Two dimensional polynomial input function

A simple two dimensional polynomial input function was used for evaluating and comparing the approximation properties of the proposed F^3 based neural networks.

$$y = (x_1 - x_2)/(x_1 + x_2)/2+0.5. \tag{28}$$

The combination of the multi-dimensional linear function and the one-dimensional quasi-sigmoid function gave the characteristic sigmoid cliff response.

The network with two hidden layers combined a number of response surfaces together, through repeated linear combination and non-linear activation functions. Figures 29 and 30 illustrate typical response surfaces of two input and a single output units. From these scenes, comparing the MSEs, it is not difficult to ascertain that the best average performance is given again by the Dombi F^3 based neural network which is followed by the Yager one.

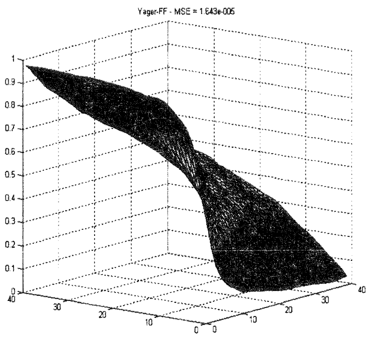


Figure 29
Simulation results

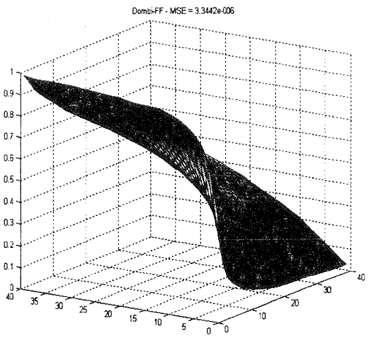


Figure 30
Simulation results

In the future we plan to do simulations with a wide range of different functions and patterns to confirm our hypothesis. It may be worth while comparing a multitude of Dombi type F^3s when parameters α are assuming their whole range.

5.5. One - input benchmark model (pH problem)

Finally, the performance of our fuzzy flip-flop based neural network was tested on one-input benchmark model the so called pH problem. The test points consist of a 101 input/output data pairs with very uneven distribution:

Domain: [0.034914, 0.743401]
Range: [0.0001, 1.0000]
No data in (0.19, 0.38); (0.39, 0.59); etc.

In this case we compared the performance of the *tansig* F^3 neuron type with the Dombi one. Figure 31 shows that in the domain with just a few data points, in the middle area there are outlayer points, thus the curve belonging to *tansig* is deviating from the target and produce overfitting. At the same time, the Dombi one follows very nice by test points interpolating everywhere uniformly well.

It is somewhat surprising, that comparing the minimum MSE values (Table 5) belonging to this two cases the results are quite different, there the *tansig* based approximation still outperforms the Dombi F^3 network.

TABLE 5. pH problem

F^3 Neuron Type		*tansig*	Dombi
MSE Training	Minimum	$9.67x10^{-10}$	$5.76x10^{-8}$
	Median	$1.28x10^{-7}$	$2.78x10^{-6}$
	Mean	$1.54x10^{-6}$	$3.37x10^{-4}$
	StdDev.	$1.01x10^{-5}$	$1.97x10^{-3}$
MSE Validation	Minimum	$2.24x10^{-7}$	$4.51x10^{-7}$
	Median	$4.15x10^{-4}$	$2.17x10^{-5}$
	Mean	$5.30x10^{-3}$	$5.79x10^{-4}$
	StdDev.	$1.97x10^{-2}$	$2.59x10^{-3}$

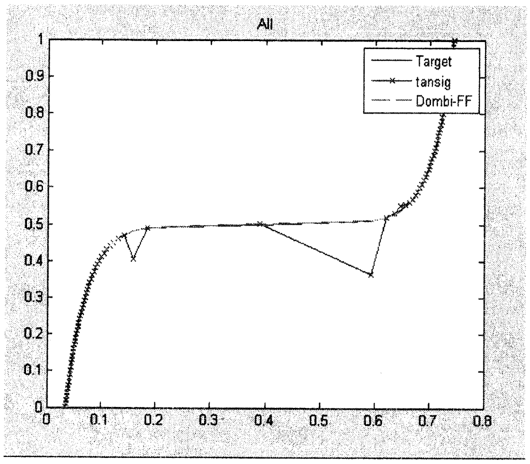


Figure 31
Simulation results

6. Conclusions

The concepts of fuzzy J-K and D flip-flops based on various t-norms have been presented. The unified equations describing these flip-flops and the characteristics were given. A fuzzy neural network (FNN) was proposed, in which the F^3s, with quasi sigmoidal *J-Q(t+1)* characteristics, substituted the traditional neurons. We tuned this neural network to perform as a function approximator based on a combination of trigonometric and polynomial test functions. We compared the performance of various type FNNs based on fuzzy J-K, D and Choi type D flip-flops, additionally using algebraic, Yager, Dombi and Fodor norms. The results were promising, the proposed fuzzy D flip-flop based neural network was found to be superior to the other approaches in approximating test functions.

References

[1] Choi, B., Tipnis, K.: *New Components for Building Fuzzy Logic Circuits*, Proc. Of the 4th Int. Conf. on Fuzzy Systems and Knowledge Discovery, vol. 2, (2007) pp. 586-590

[2] Czogała, E., Leski, J.: *Fuzzy and Neuro-fuzzy Intelligent Systems*, Physica-Verlag, Springer Verlag, (2000)

[3] Fodor, J. C., Kóczy, L. T.: *Some remarks on fuzzy flip-flops*, In: L. T. Kóczy, K. Hirota, eds., Proc. of the Joint Hungarian-Japanese Symposium on Fuzzy Systems and Applications, Technical University, Budapest (1991) pp. 60-63

[4] Fodor, J. C., Rubens, M.: *Fuzzy preference modelling and Multicriteria Decision Support*, Kluwer Academic Pub., (1994)

[5] Hagan, M., Demuth, H.: *Neural Networks for Control*, Invited Tutorial, American Control Conference, San Diego (1999) pp. 1642-1656

[6] Hornik, K. M., Stinchcombe, M., White, H.: *Multilayer feedfordward nerworks are universal approximators*, Neural Networks, Vol.2, No.5 (1989) pp. 359-366

[7] Klir, G.J., Yuan, B.: *Fuzzy sets and fuzzy logic: Theory and applications*, Prentice Hall, Upper Saddle River, NJ, (1995) pp. 50-88

[8] Kóczy, L. T., Lovassy, R.: *Fuzzy Flip-Flops Revisited*, Proc. IFSA World Congress, Cancun, Mexico, (2007) pp. 643-652

[9] Lovassy, R., Kóczy, L. T.: *Comparison of Elementary Fuzzy Sequential Digital Units Based on Various Popular T-norms and Co-norms*, Proc. 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, Timişoara (2006) pp. 164-181

[10] Lovassy, R., Kóczy, L. T., Gál, L.: *Multilayer Perceptron Implemented by Fuzzy Flip-Flops*, IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong (2008) pp. 1683-1688

[11] Ozawa, K., Hirota, K., Kóczy, L. T., Omori, K.: *Algebraic fuzzy flip-flop circuits*, Fuzzy Sets and Systems 39/2, North Holland (1991) pp. 215-226

[12] Ozawa, K., Hirota, K., Kóczy, L. T.: *Fuzzy flip-flop*, In: M. J. Patyra, D. M. Mlynek, eds., Fuzzy Logic. Implementation and Applications, Wiley, Chichester (1996) pp. 197-236

[13] Schweizer, B., Sklar, A.: *Associative functions and statistical triangle inequalities*, Publicationes Mathematicae Debrecen, 8, (1961) pp. 169-186

[14] Yager, R. R.: *On a general class of fuzzy connectives*, Fuzzy Sets and Systems, vol.4 (1980) pp. 235-242

[15] Zadeh, L. A.: *Fuzzy Sets*, Information and Control 8, (1965) pp. 338-353

Fuzzy Cognitive Maps in Path Planning

Ján Vaščák

Centre for Intelligent Technologies, Technical University in Košice
Letná 9, 042 00 Košice, Slovakia
jan.vascak@tuke.sk

Abstract: This paper deals with application of Fuzzy Cognitive Maps (FCM) in combination with a graph search algorithm A^* for purposes of path planning for a vehicle in a traffic system with dynamic changes. In this paper a simulation of a parking problem is shown. The term parking problem is reserved for searching a suitable parking place and path planning. The purpose of this paper is to show possibilities of using FCM in a dynamic environment, too.

Keywords: algorithm A^*, fuzzy cognitive map, graph search algorithms, path planning

1. Introduction

Let us consider a situation on a parking place surrounded by lots of shops with moving cars trying to park on diverse parts to have the shortest distance to a chosen shop or a full-automated production line where individual workplaces are interconnected by robotic cars.

In both cases path planning is necessary partly to achieve the goal (shortest distance to a shop or achieving given workplace) and partly to avoid traffic jam or crash. In this paper we would like to focus on using artificial intelligence means as graph search algorithms, namely A^* algorithm, and Fuzzy Cognitive Maps (FCM) for this kind of problems and to prove their effectiveness on a parking problem.

The main objective of this approach is to verify suitability of FCM in their utilization for path planning purposes because there are still only very few sources (e.g. [10]) for doing comparison this proposed approach to other ones based on FCM. Huge number of possible path planning approaches and specific proposed experiment also do not enable to do mutual comparison because they take into account diverse aspects of a given problem. The main motivation for using FCM is their comprehensibility for a human. An overview about path planning methods can be found in [2, 8].

At first, we will shortly describe both used means and especially FCM as relatively unknown knowledge representation form for fuzzy systems. Then we will characterize the parking problem and show necessary modifications of A* algorithm using FCM for purposes of estimating search cost in A*. After that a simulation example with dynamic traffic will be shown with some concluding remarks.

2. Description of Used Methods

In next sections the means from different parts of artificial intelligence will be described. The more known graph search algorithms represent symbolic approach to problem solving, where a problem is described by symbolic relations between notions. FCM are a part of computational intelligence and use numeric expressions to describe the meaning of a given notion. It seems to be reasonable to interconnect these two approaches to get description power in notions as well as in their mutual relations. However, it is necessary to remark that connecting symbolic and numeric (sub-symbolic) means of artificial intelligence is a relatively rare case. Therefore it was one another reason to explore this alternative.

2.1. Graph Search Algorithm A*

Let us have a graph consisting of nodes x interconnected by edges. A goal of graph search algorithms is to find the shortest way in a graph (e.g. a graph of a parking place). The general scheme of these algorithms can be described as follows [8]:

```
1.  Insert starting node $x_s$ into a queue $Q$ and denote it as visited.
2.  While $Q$ is not empty do:
3.      Pick out a node $x$ in a way from $Q$.
4.      If $x$ is a goal node $x_g$ then
5.          return SUCCESS and finish the algorithm
6.      else
7.          determine all descendants $x'$ of $x$ ($x$ and $x'$ are connected by edges).
8.      For all $x'$ do:
9.          If $x'$ was not yet visited then
10.             insert $x'$ into $Q$
11.             mark $x'$ as visited
12.         else
13.             determine what to do with $x'$.
14. Return FAILURE and finish the algorithm.
```

Based on how the queue Q is sorted, i.e. which order nodes x are picked out in, we can define various searching algorithms. The fundamental ones there are forward searching to breath and to depth as well as their backward modification.

A more sophisticated method how to manipulate with Q is Dijkstra's algorithm [4]. It is based on cost calculation of paths. The edges e of the graph are assigned cost values $l(e)$. The order of elements x in Q is determined according to a *cost function* $C(x)$, which represents path costs from starting node x_s to the node x. $C(x)$ is a summation of edge costs $l(e)$ creating the path. In other words, for a descendant node x' we get:

$$C(x') = C(x) + l(e), \tag{1}$$

where $l(e)$ is the cost of the connection from x to its descendant x'. In accordance to values $C(x)$ the sequence of x in Q will be ordered as non-decreasing. In the case of repeated visiting a node x' and calculating a lower value $C(x')$ the *what to do* function will replace the old cost value by the new one and will initiate a new reordering the queue Q (see the line 13). It can be proven if there exist some solutions then Dijkstra's algorithm will find the optimal one [3].

The A* algorithm is an extension of Dijkstra's algorithm with the aim to minimize the number of explored nodes. Let us consider a path from x_s to x_g via x. The equation (1) can be generalized to a form:

$$C(x_g) = C(x) + H(x), \tag{2}$$

where $C(x)$ is the cost from x_s to x and $H(x)$ the cost from x to x_g. $C(x_g)$ is then a total cost from x_s to x_g via x. During searching we can incrementally calculate $C(x)$ but usually we do not know the cost of remaining path to the goal in advance. We can try to estimate the value of $H(x)$ by a heuristic, i.e. we will get $\hat{H}(x)$. Again it can be proven if we underestimate the value of $H(x)$ then the A* algorithm will find the optimal path [5]. Of course, if the estimation $\hat{H}(x)$ equals $H(x)$ then the algorithm will explore the minimal number of nodes (minimal computational efforts). The lower estimation the greater number of nodes will be explored. If $\hat{H}(x) = 0$ then the A* algorithm will degenerate to Dijkstra's algorithm. Hence the only functional difference between these two approaches is that A* will use instead of (1) the relation $C(x') + \hat{H}(x')$ for ordering Q.

2.2. Fuzzy Cognitive Maps

In general a Cognitive Map (CM) is an oriented graph where its nodes represent notions and edges causal relations. Mostly, notions are states or conditions and edges are actions or transfer functions, which transform a state in a node to another one in another node. For instance, we can also rewrite the general scheme of searching algorithms to such a form as shown in the fig. 1.

CM is able to describe complex dynamic systems. It is possible to investigate e.g. cycles, collisions, etc. Besides it is possible to define strengths of relations, too. Originally they were represented by three values -1, 0 and 1. Another advantage is its human-friendly knowledge representation and ability to describe various types of relations (more detailed e.g. [9]).

FCM represents an extension of CM and was proposed by Kosko in 1986 [7]. The extension is based on strength values that are from an interval [-1; 1] as well as the nodes can be represented by membership functions. Strengths or rather weights after their combining correspond to rule weights in rule based systems.

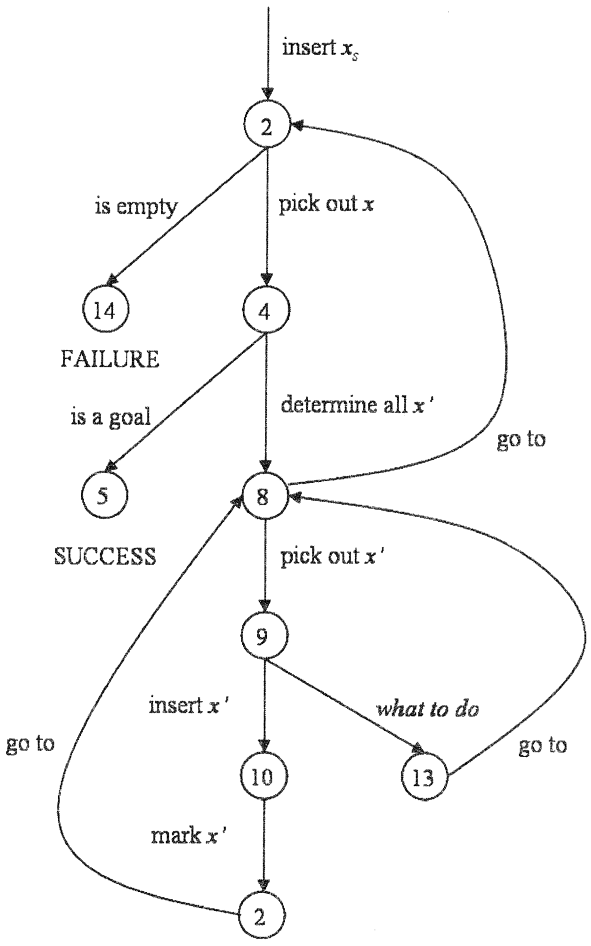


Figure 1: A CM representing the general scheme of searching algorithms; numbers in nodes correspond to line numbers of the scheme.

There are two basic formal definitions of FCM [1, 13]. Further, the definition by Chen [1] will be used where FCM is defined as a 3-tuple:

$$FCM = (C, E, \alpha, \beta) \tag{3}$$

where:

C – finite set of cognitive units (nodes) described by their states $C = \{C_1, C_2, \ldots, C_n\}$

E – finite set of oriented connections (edges) between nodes $E = \{e_{11}, e_{12}, \ldots, e_{nn}\}$

α – mapping $\alpha : C \rightarrow A$ on interval [-1; 1]

β – mapping $\beta : E \rightarrow B$ on interval [-1; 1]

In other words, α is a membership function placed in a node and the result is a grade of membership for values entering such a node. Similarly, β does the same but it is placed on a connection, i.e. it is its weight represented as a membership function. Further, we will use β only in form of a crisp number.

For computational representation of FCM a transition matrix is used. For an example in fig. 2 it will look as:

$$E = \begin{bmatrix} 0 & -1 & 0 & -1 & 1 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

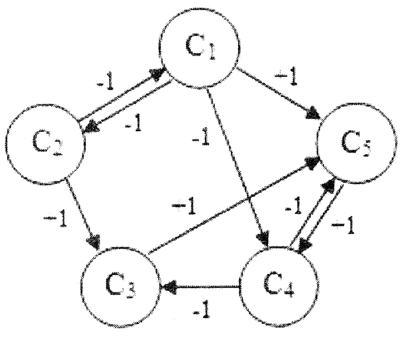


Figure 2: An example of FCM with crisp edges.

Cognitive units are in each time step in a certain state. Using transition matrix we can compute their states for next time step and thus repeatedly for further steps. Similarly as for differential equations we can draw phase portraits. To preserve values in prescribed limits a boundary function L is used, too. So we can compute the states for $t+1$ as follows [1]:

$$C(t+1) = L(C(t).E). \tag{5}$$

As seen in fig. 2 FCM are in comparison to fuzzy rule based systems their extension. Fuzzy rules are totally independent. Their consequents do not have any mutual influence, which is possible only in simpler decision cases. Rule based systems do not enable e.g. creating implication chains or representation of temporal information. From this point of view they are only a very special and restrained case of FCM, which can be depicted in fig. 3 comparing a fuzzy rule with n inputs $x_1,\ldots,x_n$, (one output LU) and its redrawing into FCM. Combining strengths $\beta_1,\ldots,\beta_n$ we will get the rule weight w.

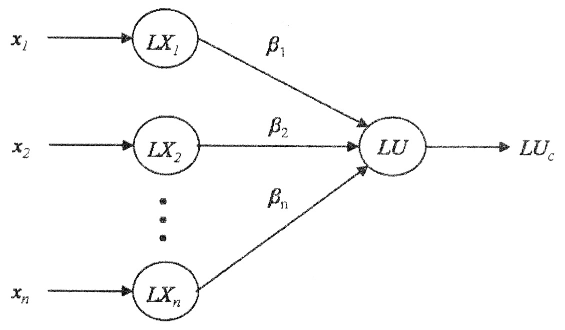


Figure 3: A fuzzy rule with n inputs transformed to FCM.

3. Path Planning in the Parking Problem

The parking problem puts some limitations on making plans. They are caused partly by traffic rules and partly by risk of mutual crashes. Therefore before using described means their modification is necessary to match these restrictions. In addition, as experience from performed experiments, a traffic simulation system for computation of time schedules (timetables) for each car is very useful for finding optimal (or almost optimal) solutions.

3.1. Description of the Parking Problem

Under the name parking problem several mutually connected aspects can be considered in literature. In this case we take into account the problem of path planning, i.e. finding a sequence of edges leading from the starting position (entrance to a parking place) to the aimed (chosen) position. Let us suppose a parking place depicted in fig. 4. There are depicted parking boxes (solid drawing) and routes with crossings (dashed drawing). Occupied boxes are denoted by X. Further, it is surrounded by various shops and customers try to park their cars as close to their chosen shops as possible. However, there is still one fact more it is necessary to take into consideration. It is the so-called *path cost* (the more expensive the less suitable). The traffic situation can be so complicated it would be better to park a little farther than to risk a traffic jam, etc. It would be better at first to consider several potential paths leading to a reasonable surrounding of a shop and then to compute their costs. After that considering distance of a parking box and cost of getting there the best path will be chosen. This is a typical task for each driver coming to a larger parking place. The aim of this research is to propose convenient means for constructing a consultation system, which would be able to navigate drivers in environment of large parking places or traffic systems of bigger downtown areas.

We suppose the form of the parking place as well as information of occupied parking boxes are known (simply realizable by camera systems). Such a parking place can be transformed into an oriented evaluated graph because we know lengths of routes and will be able to calculate path costs, too (see next section). In that case the path finding and planning problem can be solved as graph search using algorithms like A^*.

3.2. Graph Constructing and Path Planning

For path planning at least these criteria are necessary to be taken into account and thereby summarizing the path cost:

- path length,
- number of turnings (the less the better),
- traffic cost (waiting).

Path length and number of turnings on crossings can be calculated from the scheme of a parking place very easily but traffic cost requires taking into account traffic rules, especially relations between main and minor routes and rules concerning turnings on crossings. From this reason it is necessary to know path plans of other cars, which enable to compute waiting. Therefore it is suitable to create a system for traffic simulation whose output is a timetable describing on what place and at what time a car will be. For a certain time *t* we can get an overview about cars positions.

Further, there are still two other limitations – prohibition of a backward turning on a crossing and the so-called *P-problem* (see fig. 5).

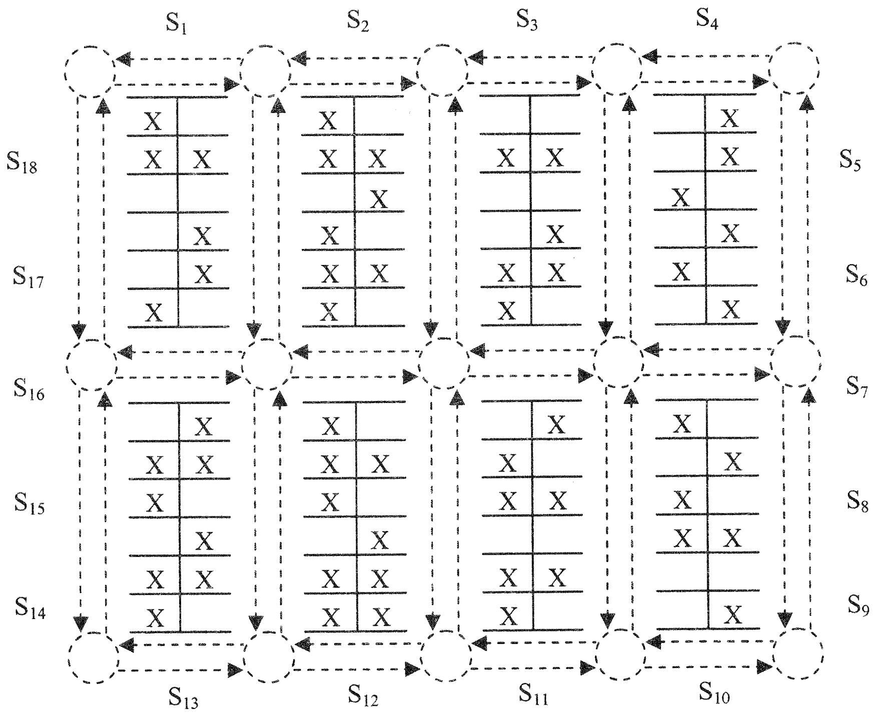


Figure 4: Example of a parking place with some occupied boxes denoted by X with allowed routes and crossings (dashed drawing); S_i – surrounding shops.

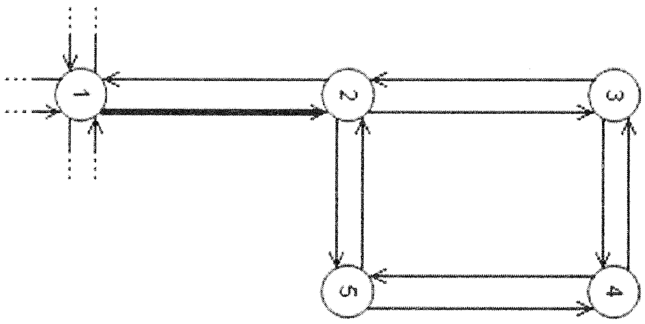


Figure 5: Example of a P-problem; crossings are numbered.

If a car is moving from node 1 to 2 then it will be prohibited directly to turn back, i.e. to move 2→1. However, the A^* algorithm will expand the node 2 (x) to nodes 3, 5 and also 1 (x'). Therefore, it is necessary to insert into the line 7 a condition for expanding nodes x': *if x' is different from the ancestor of x.*

Usual reasoning how to construct a search graph of the parking place is to use crossings as nodes and routes as its edges. However, the P-problem causes the algorithm A^* will not be able to decide between routes 1→2→3→4→5→2→1 and 1→2→5→4→3→2→1 (fig. 5). The algorithm will empty the queue Q and after that it will return failure and finish without finding a path. To prevent this situation crossings (fig. 4) will not be regarded as nodes of the graph but routes. In our case in fig. 4 there are 44 routes (edges) so the graph will have 44 nodes and their edges will represent applications of traffic rules between given nodes. In other words, edges of such a modified graph will represent traffic restraints or conditions for getting from one node to the other one. Path costs will serve as edge evaluations.

All three mentioned criteria are defined on diverse dimensions. Therefore, it will be necessary to merge them into a variable named as *path cost*. Its values will help the algorithm in deciding what path will be preferred. For this purpose a simple FCM will be used (see fig. 6). Weights w_1, w_2 and w_3 determine the influence of chosen criteria. A user can set up them by his needs. In such a manner we can modify strategies for choice of a parking box depended on various circumstances. The output from the node *path costs* is the edge value.

The system for traffic simulation is quite complicated and some unexpected events may occur, which cause change of path plans. However, considering the property of A^* in (2) that the hypothesis $H(x)$ needs to be underestimated some additional delays will not affect the functionality of A^*. Only the resulting path plan needs not be temporally optimal because from the mathematical point of view we will get a sub-graph of the graph describing real waiting. In the worst case we can omit the whole traffic simulation system choosing $w_3=0$ in fig. 6, too. Other calculations in this FCM are the same as in conventional rule based fuzzy inference systems.

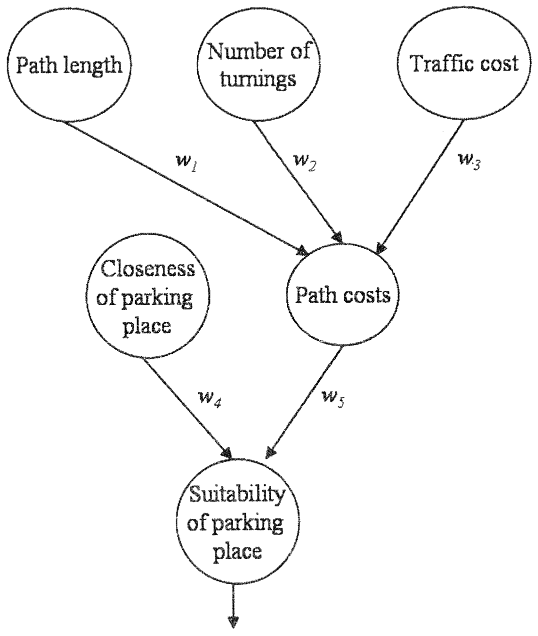


Figure 6: FCM for calculation of values for graph edges.

Beside path costs also the first (in the introduction) mentioned criterion – distance, i.e. closeness of a parking box is taken into account in the same way as path cost and the final result is the suitability of chosen parking box.

3.3. Traffic Simulation System

Although the existence of a traffic simulation system is not requisite for the function of path planning but it can very significantly optimize found solutions. It serves for calculation of traffic cost, i.e. waiting, which is necessary to continue the chosen path.

We suppose the path plans of all cars are known. On their base as well as on the information about the parking place form and traffic rules we can construct the so-called *temporal-spatial plan* for entire traffic, i.e. beside the spatial information (form of a path) also temporal information (when and where) will be added. Calculating the temporal part represents actually computation of waiting. For this purpose it is necessary to incorporate traffic and crash preventing rules into the calculation algorithm, i.e. rules for main and minor roads and prohibition to put two cars on the same place at the same time. Finally, it is also necessary to take into account the capacities of edges (roads). In such a manner we will compute timetables for each car, i.e. we will know where a car will be at a certain time. The set of these timetables creates the temporal-spatial plan (see fig. 7).

cars (vehicles) ↓ \ time →					t				
v1	12.03	12.02	12.02	12.01	32.07	32.06	32.05	32.04	...
v2	15.08	15.07	15.06	15.05	15.04	15.03	15.02	15.01	...
v3	22.04	22.04	22.04	22.03	22.03	22.02	22.01	15.08	...
v4	31.06	31.05	31.04	31.03	31.02	31.01	33.08	33.07	...
v5	12.04	12.03	12.03	12.02	12.01	25.04	25.03	25.02	...
v6	18.05	18.04	18.03	18.02	18.02	18.01	17.05	17.04	...
⋮									

Figure 7: An example of a temporal-spatial plan.

Horizontal axis represents growing time for each car. The rows are actually timetables for individual cars and the column in the time *t* represents the overall situation on the parking place, i.e. the positions of the cars. Items of the plan are encoded car positions, e.g. the car *v2* will be in the time *t* on the edge number *15* in its part *4* (*15.04*). The number of parts for a given edge defines its capacity, too. The difference between total number of time steps and steps of free (not restricted) movements from a starting point to a goal gives the value of waiting.

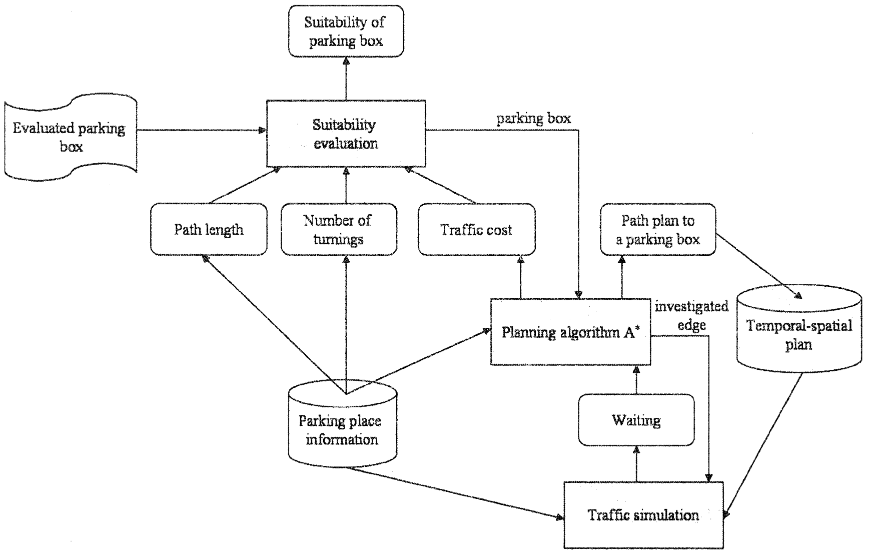


Figure 8: Structure of the path planning system.

Traffic simulation requires creating a centralized database, which will contain all timetables and thereby the entire planning system must be centralized. In the fig. 8 we can see the overall structure of the path planning system. The three gray shaded blocks are algorithmic parts of the planning system. For suitability evaluation a FCM from fig. 6 is used and waiting is the result of the traffic simulation system. Necessary data for evaluation of a chosen parking box (band-shaped block), which initializes the computation, are stored in two databases (grey hatched blocks). The database of parking place is static. There are stored geometric data about its form, boxes and paths (roads). This database will be changed only if some construction changes are done. The database of the temporal-spatial plan is dynamic and it must be updated every time step when new calculations are performed. Static and dynamic natures of these data are the main reason why they are divided into two distinct databases. Oval blocks describe parts of information, which enter or result from modules of the path planning system. Parking boxes that are evaluated are determined by the *closeness* function (fig. 6). The whole computational cycle is initialized extra for each chosen parking box to be evaluated. Thereby we will get suitability evaluations for each chosen parking box and that one with the best suitability will be definitely offered.

The overall function of the path planning system is following:

1. At first, it chooses several potential free parking boxes in surroundings of a preferred shop (given by the *closeness* function).
2. The suitability of each chosen parking box is computed, i.e.:
   a) The A^* will find optimal paths to these boxes.
   b) Continuing calculation in fig. 6 we will get suitability for each of chosen boxes.
3. The box with the best suitability will be offered.
4. Finally, the timetable of the proposed path plan will be inserted into the temporal-spatial plan.

4. Experiments

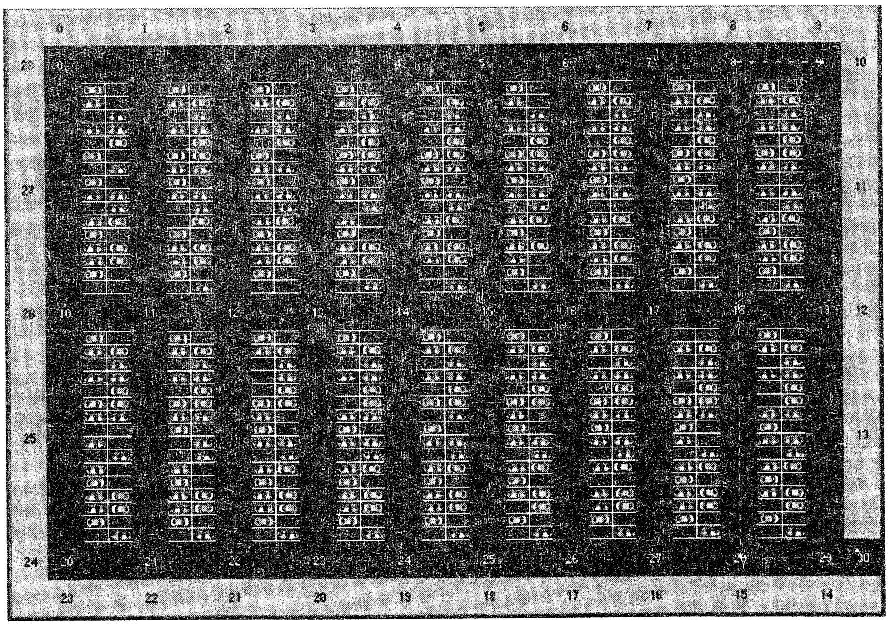


Figure 9: Simulation window of the proposed path planning system with depicted traffic.

In [12] a system for path planning based on A* searching and using FCM for evaluating edges was proposed (see fig. 9). The figure shows a car (marked as red), which the path is calculated for and a dashed line is the proposed path. The system offers animated simulation so it is possible to observe traffic in every time step. A user has possibility to adjust two kinds of parameters: weights w and membership functions. Experiments showed it is advantageous to use so many criteria as possible. Their higher number decreases the number of possible solutions so the number of expanded nodes and computational complexity will be lesser. However, if the criteria are too strict the algorithm will try to expand more nodes but without any better result. On contrary, if criteria are too weak there is a risk the algorithm will find a solution very quickly but not an optimal one. From this reason it is very reasonable to propose a traffic simulation system, which would be able at least roughly to simulate traffic load and waiting.

Further part of experiments dealt mainly with setting up the membership function of *closeness*. Its form and parameters determine the surrounding, which potential parking boxes are searched in. It is necessary to set up this range not too strictly because only few boxes can be found (extremely none). Experiments validated also the need to take into account traffic load. In many experiments close parking boxes were charged with high traffic delays and it was advantageous rather to park on a little distant place than to wait in a long crowd.

5. Conclusions

The proposed planning system has two main advantages. Firstly, it is possible easily to modify it by customer's efforts. For example, for a disabled customer the criterion of closeness will be much more important than waiting. It is necessary only to change weights and partially some membership functions (first of all the *closeness*) in FCM. Secondly, the system described by FCM is very comprehensive. Relations between criteria are clear. It is very easy to add further additional criteria, etc. The system is thanks to the condition (2) able to work also with imprecise information about traffic delays with satisfactory results (although not optimal). In addition, the total number of parameters is small and their setting up is relatively easy and quick, which is a special user-friendly aspect of used means. Finally, if the search criteria are properly set up then the A* will enable quick finding an optimal solution. Experience has shown that setting up criteria is not a difficult problem and it takes only a few modification steps.

For future research there are two perspective approaches. Firstly, to 'fuzzify' the traffic simulation system, which would be able in certain measure to absorb some unexpected events and the system could become more robust. Secondly, to propose adaptive mechanisms for a self-tuning FCM like described for instance in [11, 6].

References

[1] Chen, S. M.: *Cognitive-map-based Decision Analysis Based on NPN Logics*, Fuzzy Sets and Systems vol. 71 (2), (1995) pp. 155-163

[2] Collective, *Research and task analysis of telematic planning*, research report PELOTE, IST-2001-38873, (2003) pp. 29

[3] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.: *Introduction to Algorithms*, (2nd Ed.), MIT Press, Cambridge, USA (2001)

[4] Dijkstra, E. W.: *A note on two problems in connection with graphs*, Numerische Mathematik, No. 1, (1959) pp. 269-271

[5] Fikes, R. E., Nilsson, N. J.: *STRIPS: A new approach to the application of theorem proving*, Artificial Intelligence Journal, No. 2, (1971) pp. 189-208

[6] Khan, M. S. et al. *A Methodology for Developing Adaptive Fuzzy Cognitive Maps for Decision Support*, Journal of Advanced Computational Intelligence vol. 4, No. 6, (2000) pp. 403-407

[7] Kosko, B.: *Fuzzy cognitive maps*, International Journal of Man-Machine Studies vol. 24 (1), (1986) pp. 65-75

[8] LaValle, S. M.: *Planning Algorithms*, Cambridge University Press, United Kingdom, available at http://planning.cs.uiuc.edu/, (2006).

[9] Mls, K.: *Hybrid Fuzzy Cognitive Systems*, (in Czech), III. Slovak-Czech Seminar Cognition, Artificial Life and Computational Intelligence, Stará Lesná, elfa, s.r.o., Slovakia, (2003) pp. 233-238

[10] Motlagh, O., Hong, T., Ismail, N.: An Intelligent Mobility Aid System for Visually Impaired Persons According to the Blinds Fuzzy Cognitive Map (FCM) in Local Path Planning, The 2nd National Intelligent Systems And Information Technology Symposium (ISITS'07), ITMA -UPM, Malaysia, (2007) pp. 152-157

[11] Preitl, St. Precup, R. E., Fodor J., Bede, B: *Feedback Tuning in Fuzzy Control Systems. Theory and Applications*, Acta Polytechnica Hungarica (Budapest Tech Polytechnical Institution), vol. 3, No. 3, (2006) pp. 81-96

[12] Rutrich, M.: "Path Planning in Traffic Environment", MSc. thesis in Slovak, Technical University in Kosice, Slovakia, (2007)

[13] Stach, W., Kurgan, L., Pedrycz, W., Reformat, M.: "Genetic learning of fuzzy cognitive maps", Fuzzy Sets and Systems, vol. 153, (2005) pp. 371-401

Advanced Bacterial Memetic Algorithms

L. Gál [1,2], L. T. Kóczy [1,3]

**[1]Faculty of Engineering Sciences, Széchenyi István University,
H-9026, Győr, Egyetem tér 1., Hungary
Email: gallaci@ttmk.nyme.hu, koczy@sze.hu**

**[2]Department of Technology, Informatics and Economy
University of West Hungary, H-9700, Szombathely, Károlyi G. tér 4. Hungary
e-mail: gallaci@ttmk.nyme.hu**

**[3]Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics,
H-1117, Budapest, Magyar tudósok krt. 2. Hungary
koczy@tmit.bme.hu**

Abstract: *Fuzzy systems* have been successfully used in the area of controllers for a long time. The first appearance of these controllers was in 1974 by Mamdani and Assilian [18]. The main problem in the usage of *Mamdani-type inference system* and other fuzzy logic based controllers is how to gain the *fuzzy rule base* (FRB) what the inference system based on.

Nawa, Hashiyama, Furuhashi and Uchikawa proposed a novel type of *Genetic Algorithm* called *Pseudo-Bacterial Genetic Algorithm* (PBGA) for *fuzzy rule base* (FRB) extraction from input-output data (1995, 1997) [21]. Furthermore, Nawa and Furuhashi improved the PBGA concerning the relationship among the individuals (the rules) and proposed a new method for automatic FRB identification named *Bacterial Evolutionary Algorithm* (BEA) (1999) [22].

Botzheim, Cabrita, Kóczy and Ruano have applied another technique to gain FRBs, they have used a local search method, the *Levenberg-Marquardt algorithm* (LM) [3] [2].

The most significant improvement in the speed of convergence and the quality of the model achieved by the FRB identification process was the idea of combining the global and local search methods. Botzheim, Cabrita, Kóczy and Ruano proposed a novel method called *Bacterial Memetic Algorithm* (BMA, 2005) [1], [4].

Although *Bacterial Memetic Algorithm* provides a very good speed of convergence towards the optimal rule base there are likely some points of the algorithm where the performance could be increased. Gál, Botzheim, Kóczy and Ruano proposed new elements (*Swap, Merge*) for *knot order violation handling* in *Levenberg-Marquardt* method used in BMA (*Improved Bacterial*

Memetic Algorithm, IBMA, 2008) [6]. Another improvement of the BMA is the *Bacterial Memetic Algorithm with the modified operation execution order* (BMAM, Gál, Botzheim and Kóczy, 2008) [7]. This new approach proposed using the Levenberg-Marquardt method more efficiently. Combining the improvements in IBMA and BMAM is beneficial (Gál, Botzheim and Kóczy, 2008) [8]. This advanced version of the *Bacterial Memetic Algorithm* used for FRB extraction named *Modified Bacterial Memetic Algorithm*.
This paper summarizes the *bacterial type evolutionary algorithms used for FRB identification.*

Keywords: *fuzzy systems, Mamdani-type inference system, Bacterial Memetic Algorithm (BMA), Levenberg-Marquardt method (LM), Modified BMA (MBMA)*

1. Introduction

The *bacterial type evolutionary algorithms* have been first developed and applied for identifying *fuzzy rule bases* automatically.

In the course of the function of *fuzzy controllers* the input data is processed by the *inference system* supported by the so called *fuzzy rule base*. The *fuzzy rule base* consists of one or more *fuzzy rules*. One of such a *rule* holds the expected output for a certain input or inputs (in multidimensional case). Commonly, in case of *fuzzy systems* the input and output data are not *crisp values* but *fuzzy values* determined by a kind of *fuzzy membership function*. These *membership functions* can be fairly various; however, in the practice the most commonly used ones are the *triangular shaped* and the *trapezoidal shaped fuzzy membership functions*.

Commonly, in case of describing *triangular shaped membership functions* it is enough to specify two parameters (isosceles triangle): the position of the top (a) and the length of the base (b) of the triangle (Fig. 1).

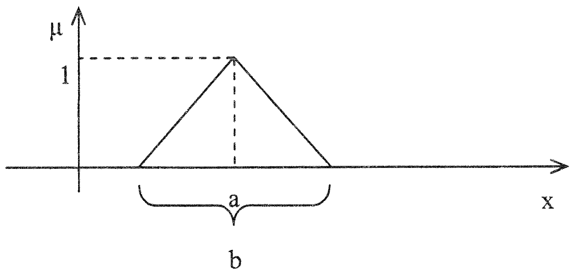


Figure 1. Triangular shaped fuzzy membership function

Trapezoidal shaped fuzzy membership functions offer more potential than the *triangular shaped* ones. These are widely used and are general enough from a mathematical point of view. Commonly, in case of describing *trapezoidal shaped fuzzy membership functions* four parameters are specified. These are the positions of the four *breakpoints* (a, b, c, d) of the trapezoid (Fig. 2).

In the case of using trapezoidal shaped *fuzzy* membership functions the four breakpoints that define the shape of each trapezoid must satisfy the following constraint:

$$a \leq b \leq c \leq d. \tag{1}$$

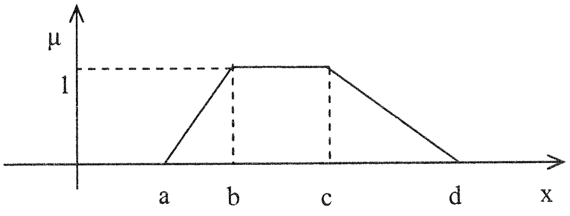


Figure 2. Trapezoidal shaped fuzzy membership function

In case of *Mamdani-type inference systems* working with *trapezoidal shaped fuzzy membership functions*, one rule consists of N_{Inputs} *antecedents* and one *consequent* (where N_{Inputs} is the dimension of the input). Accordingly, since these are all trapezoids with four breakpoints, one rule can be defined by $(N_{Inputs}+1) \cdot 4$ parameters. As the *fuzzy rule base* contains N_{Fuzzy_rules} rules (N_{Fuzzy_rules} is the number of fuzzy rules of the rule base), the rule base can be defined by $N_{Fuzzy\_rules} \cdot (N_{Inputs} + 1) \cdot 4$ parameters. If a rule base is built up of 5 rules and the number of input variables is 6, then the full number of the parameters needed to define the rule base is 140.

The task is to find the *fuzzy rule base* which one fits for the functioning of a given system best. In the case above it means determining and tuning of 140 parameters.

One of the serious problems of fuzzy rule base modeling is how to find the optimal or quasi-optimal rule base for a certain system when no human expert is available to gain the rules. In this case we need a method for identifying the fuzzy rule base automatically.

One can find several approaches in the literature for fuzzy model identification (e.g. [23]). Some of them determine the rules and the corresponding linguistic terms based on fuzzy clustering (e.g. the method proposed in [14] or ACP in [10]). Another group of methods (e.g. RBE-DSS and RBE-SI [11]) start with two initial rules that describe the maximum and minimum of the output and extend the rule base iteratively in course of the tuning. Most of the methods mentioned above are also able to identify fuzzy models with low complexity by generating sparse rule bases. These systems use *fuzzy rule interpolation* (FRI) based reasoning (e.g. [15], [16], and [12]). An application oriented aspect of the FRI emerges in "FIVE" (Fuzzy Interpolation based on Vague Environment, originally introduced in [16], [17]), where for the sake of reasoning speed and direct real-time applicability the fuzziness of fuzzy partitions replaced by the concept of Vague Environment and hence the fuzzy interpolation to crisp one. Recently a freely available comprehensive FRI toolbox [13] and an FRI oriented web site (*fri.gamf.hu*) were appeared for aiding and guiding the future FRI applications.

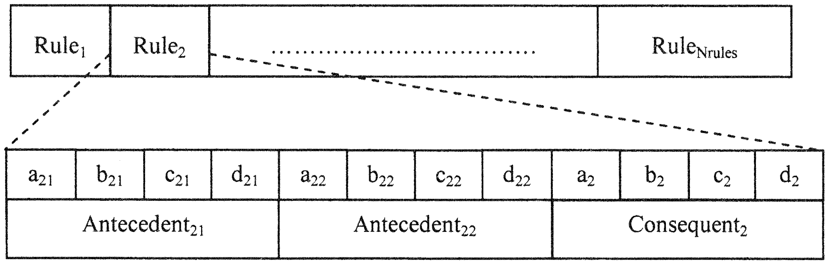
Various evolutionary approaches have been proposed for fuzzy rule base extraction from input-output data such as the *Pseudo-Bacterial Genetic Algorithm* (PBGA) [21], the *Bacterial Evolutionary Algorithm* (BEA) [22], the *Bacterial Memetic Algorithm* (BMA) [1], the *Improved Bacterial Memetic Algorithm* (IBMA) [6], the *BMA with the Modified Operator Execution Order* (BMAM) [7] and the *Modified Bacterial Memetic Algorithm* (MBMA) [8]. All these have turned out to be helpful with the construction of such fuzzy rule base models; however, their respective optimality has been different in each case. This paper summarizes the *bacterial type evolutionary algorithms used for fuzzy rule base identification.*

2. Pseudo-Bacterial Genetic Algorithm (PBGA)

The original genetic algorithm (GA) was developed by Holland [9] and was based on the process of evolution of biological organisms. These processes can be easily applied in optimization problems where one individual corresponds to one solution of the problem.

Nawa, Hashiyama, Furuhashi and Uchikawa proposed a novel type of *Genetic Algorithm* called *Pseudo-Bacterial Genetic Algorithm* (PBGA) for fuzzy rule base extraction (1995, 1997) [21]. The *Pseudo-Bacterial Genetic Algorithm* is a special kind of *Genetic Algorithm* [9]. Its core contains a new genetic operation called *bacterial mutation*, which is inspired by the biological bacterial cell model, so this method mimics the microbial evolution phenomenon. Its basic idea is to improve the parts of *chromosomes* contained in each *bacterium*.

Bacteria can transfer *genes* to other *bacteria*. This mechanism is used in the *bacterial mutation*. For the *bacterial algorithm*, the first step is to determine how the problem can be encoded in a *bacterium* (*chromosome*). Our task is to find the optimal *fuzzy rule base* for a pattern set. Thus, the parameters of the *fuzzy rules* must be encoded in the *bacterium*. The parameters of the rules are the breakpoints of the trapezoids, thus, a bacterium will contain these breakpoints. For example, the encoding method of a fuzzy system with two inputs and one output can be seen in Fig. 3.



Rule_1
Rule_2
$\text{Rule}_{\text{Nrules}}$
a_{21} b_{21} c_{21} d_{21} a_{22} b_{22} c_{22} d_{22} a_2 b_2 c_2 d_2
Antecedent_{21}
Antecedent_{22}
Consequent_2

Figure 3. Encoding of the fuzzy rules

The next step is to optimize the parameters. Therefore a procedure is working on changing the parameters, testing the model obtained by this way and selecting the best models. In the course of testing the input-output data used for training are compared to the input and the output of the model (SSE, MSE, BIC). The smaller error, the better performance of the model.

The flowchart of the *Pseudo-Bacterial Genetic Algorithm* can be seen in Fig. 4, and its main steps are described below:

- Create the initial population: N_{Ind} individuals are randomly created and evaluated. (N_{Ind} is the number of individuals in the population.) Each individual contains N_{Fuzzy_rules} fuzzy rules encoded in the chromosome (N_{Fuzzy_rules} is the number of fuzzy rules of the desired model).
- Apply the *bacterial mutation* to each individual
    - Each individual is selected one by one.
    - N_{Clones} copies of the selected individual are created ("clones").
    - Choose the same part or parts randomly from the clones and mutate it (except one single clone that remains unchanged during this mutation cycle).
    - Select the best clone and transfer its mutated part or parts to the other clones.
    - Repeat the part choosing-mutation-selection-transfer cycle until all the parts are mutated and tested exactly once.
    - The best individual is remaining in the population, all other clones are deleted.
    - This process is repeated until all the individuals have gone through the bacterial mutation.
- Apply conventional genetic operations (selection, reproduction and crossover).
- Repeat the procedure above from the *bacterial mutation* step until a certain termination criterion is satisfied (e.g. maximum number of generations).

The algorithm works efficiently in environments where there are weak relationships between the parameters encoded in the chromosome. *Fuzzy rule bases* have this property, so PBGA has been successfully applied for obtaining quasi-optimal rules of fuzzy systems based on input-output training sets. PBGA performs well, converges fast towards the optimal rule base and simple to implement.

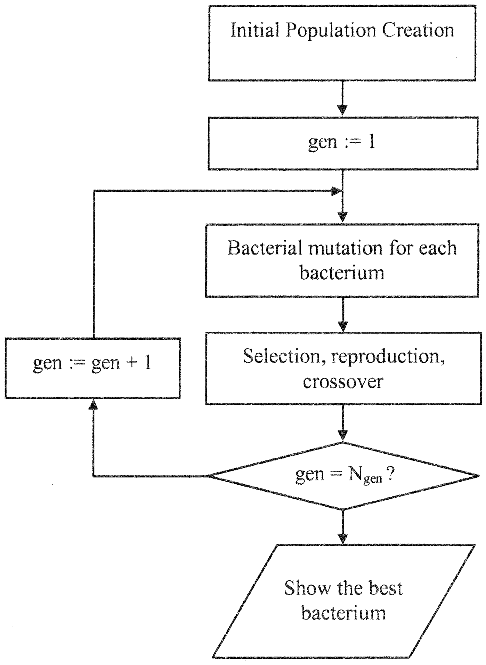

Initial Population Creation
gen := 1
Bacterial mutation for each bacterium
Selection, reproduction, crossover
gen := gen + 1
gen = N_gen ?
Show the best bacterium

Figure 4. Flowchart of the PBGA

3. Bacterial Evolutionary Algorithm (BEA)

Bacterial Evolutionary Algorithm (BEA) is based on the PBGA supported by a new genetic operation called *gene transfer operation* [22]. This new operation establishes relationships among the individuals of the population. It can also be used for decreasing or increasing the number of the rules in a fuzzy rule base.

The main steps of the *gene transfer operation* are:

- Sort the population according to the fitness values and divide it in two halves. The half that contains the better individuals is called superior half while the other half is the inferior half.
- Choose one individual (the "source chromosome") from the *superior half* and another one (the "destination chromosome") from the *inferior half.*
- Transfer a part from the source chromosome to the destination chromosome (select the part randomly or by a predefined criterion).
- Repeat the steps above N_{Inf} times (N_{Inf} is the number of "infections" to occur in one generation.)

The *gene transfer operation* can be used in place of *selection, reproduction, crossover* in the algorithm described in Fig. 4. The flowchart of the *Bacterial Evolutionary* Algorithm can be seen in Fig. 5.

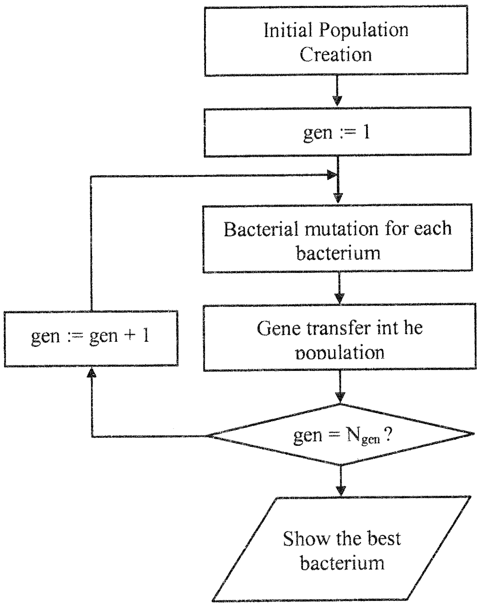

Initial Population Creation
gen := 1
Bacterial mutation for each bacterium
Gene transfer int he population
gen := gen + 1
gen = N_{gen} ?
Show the best bacterium

Figure 5. Flowchart of the BEA

4. Bacterial Memetic Algorithm (BMA)

Memetic Algorithms combine evolutionary and local search methods (P. Moscato, 1989) [20]. The evolutionary part is able to find the global optimum region, but is not suitable to find the local minimum in practice. The gradient based part is able to reach the local optimum, but is very sensitive to the initial position in the search space and is unable to avoid the local optimum. Combining global and local search is expected to be beneficial.

Bacterial Memetic Algorithm (BMA) is a very recent approach. It combines the *Bacterial Evolutionary Algorithm* and the *Levenberg-Marquardt* method. It can be used for fuzzy rule base identification because the derivatives for the Jacobian matrix can be computed for the general trapezoidal fuzzy membership functions (with COG defuzzification) [1], [2], [5]. It provides significant improvements both in terms of the speed of convergence and in the quality of the model achieved in FRB identification.

4.1. Levenberg-Marquardt method (LM)

The *Levenberg-Marquardt* (LM) method [19] is a gradient based iterative procedure. It is used for least squares curve fitting for a given set of empirical data ($\underline{x}_i$, t_i). Its main equation is

$$\left(\underline{\underline{J}}^T \underline{\underline{J}} + \lambda \underline{\underline{I}}\right)\underline{s} = -\underline{\underline{J}}^T \left[\underline{y}\left(\underline{p}\right) - \underline{t}\right] \tag{2}$$

where $\underline{p}$ is the parameter vector to be optimized, $\underline{t}$ is the target vector, $\underline{y}$ is the output vector produced by the model, $\underline{\underline{J}}$ is the Jacobian of $\underline{y}$ at $\underline{p}$, and $\underline{s}$ is the update vector to $\underline{p}$. The dumping parameter λ controls the direction and the size of the step that will be taken.

The equation above can be recast as

$$\underline{s} = -\begin{bmatrix} \underline{\underline{J}} \\ \sqrt{\lambda}\,\underline{\underline{I}} \end{bmatrix}^+ \begin{bmatrix} \underline{y}\left(\underline{p}\right) - \underline{t} \\ \underline{0} \end{bmatrix} \tag{3}$$

The operator $^+$ denotes the Moore-Penrose pseudoinverse.

After solving the equation above in the kth iteration the update vector $\underline{s}$ is applied to optimize the parameter vector $\underline{p}$ in the following way:

$$\underline{p}[k] = \underline{p}[k-1] + \underline{s}[k] \tag{4}$$

In case of FRB optimization the parameter vector contains the parameters of one FRB (or one chromosome).

The LM method can be used for fuzzy rule extraction directly [2], but combining it with the BEA provides definitely better results.

4.2. The Bacterial Memetic Algorithm

The algorithm is based on the operations of the PGBA (*bacterial mutation*), BEA (*gene transfer*) and the *Levenberg-Marquardt* method. It is much more successful in FRB identification than its predecessors.

The flowchart of the *Bacterial Memetic Algorithm* can be seen in Fig. 6, and its main steps are described below:

- Create the initial population.
- Apply the *bacterial mutation* to each individual.
- Apply the *Levenberg-Marquardt* method to each individual (e.g. 10 iterations per individual per generation).
- Apply the *gene transfer* operation N_{Inf} times per generation.
- Repeat the procedure above from the *bacterial mutation* step until a certain termination criterion is satisfied.

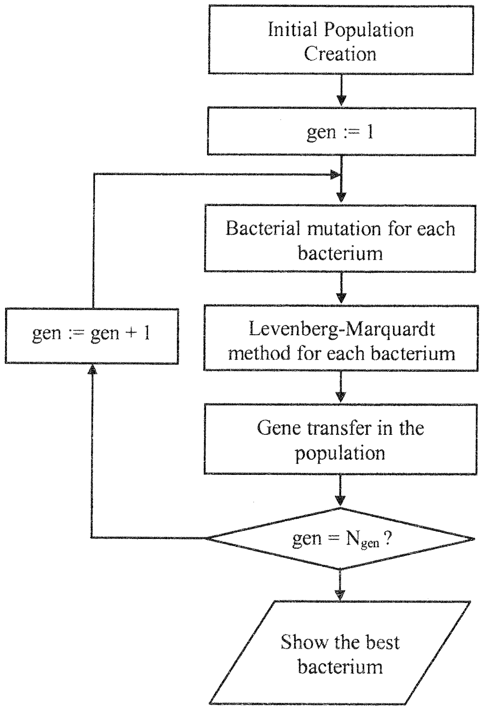

Initial Population Creation
gen := 1
Bacterial mutation for each bacterium
Levenberg-Marquardt method for each bacterium
Gene transfer in the population
gen = N_gen ?
gen := gen + 1
Show the best bacterium

Figure 6. Flowchart of the BMA

In the BMA the LM procedure has to be modified. In case of trapezoidal shaped fuzzy membership function the parameter vector contains the four breakpoints (a, b, c, d or K_1, K_2, K_3, K_4) for each trapezoid. Applying the update vector calculated by the LM method some breakpoints of the trapezoids may be swapped. It happens not too often but it may cause serious problem as abnormal trapezoids may be obtained (Fig. 7). In case the order of the breakpoints of a trapezoid does not satisfy the $K_1 \leq K_2 \leq K_3 \leq K_4$ constraint, then the membership function defined by the four breakpoints cannot be interpreted as a fuzzy membership function.

In the BMA in case of *knot order violation* (KOV) an *update vector reduction factor* is applied (g) [3]. This factor is a number between 0 and 1, it limits the magnitude of the update computed by LM for that pair of points which causes the damage of the knot order. It can be calculated as follows:

$$g = \frac{K_{i+1}[k-1] - K_i[k-1]}{2(s_i[k] - s_{i+1}[k])} \tag{5}$$

where $K_i[k\text{-}1]$ is the i^{th} breakpoint of the trapezoid before the k^{th} iteration (at the beginning of the current LM iteration), and $s_i[k]$ is the current LM update for the i^{th}

breakpoint. After calculating factor g the adjusted position of the breakpoints can be computed as (K'):

$$K'_i[k] := K_i[k-1] + g \cdot s_i[k],$$
$$K'_{i+1}[k] := K_{i+1}[k-1] + g \cdot s_{i+1}[k]. \tag{6}$$

5. Improved Bacterial Memetic Algorithm (IBMA)

Although *Bacterial Memetic Algorithm* provides a very good speed of convergence towards the optimal rule base there are likely some points of the algorithm where the performance could be increased. One of these points concerns the *knot order violation handling.*

The original BMA handles this problem by computing and applying the *update vector reduction factor*. The drawback of the above method is that in case of knot order's damage the full power of the LM method cannot be utilized because it limits the magnitude of the update (approx. to the half of the allowed value), and this method should be integrated into the LM procedure much deeper.

Gál, Botzheim, Kóczy and Ruano proposed new elements (*Swap, Merge*) for KOV handling in LM used in BMA (2008) [6]. The algorithm containing a new KOV handling technique (*Swap)* rather than the *update vector reduction factor* is simpler and a slightly more powerful than the BMA. It is called *Improved Bacterial Memetic Algorithm* (IBMA).

5.1. Improvements in knot order violation handling

The two alternate methods for KOV handling are:

a. Merge of the violating knots into a single knot. (Merge)

b. Swap of the knots that are in the wrong order. (Swap)

We found that both of these methods perform slightly better than the original one used in the BMA (especially the method *swap*), besides they are easier to implement and to integrate in the BMA.

The main point of the KOV handling method *swap* is in the case of the knot order violation the rate of the shift of both of the two breakpoints that have been computed by the LM method has to be applied as much as it can be done; however without the formation of trapezoids with vertical edges, and in such a manner that the algorithm can be applied *after* the update part of the LM algorithm. Corresponding to these, the method is to *swap* the two violating knots, so the formation of abnormal trapezoids or trapezoids with vertical edges can always be avoided (Fig. 7).

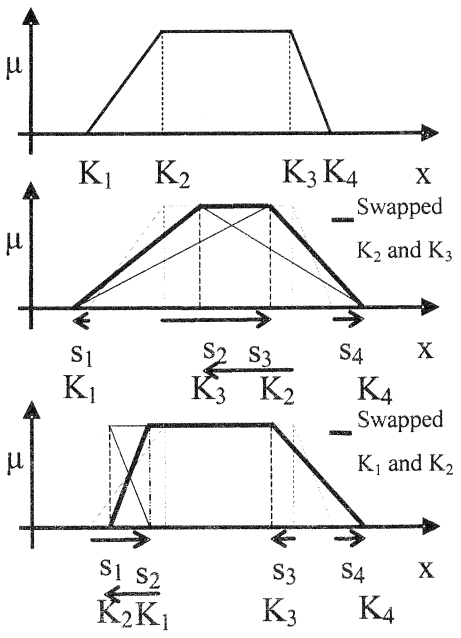


Figure. 7. KOVH method Swap

6. BMA with the modified operator execution order (BMAM)

Another improvement of the BMA is the *Bacterial Memetic Algorithm with the modified operation execution order* (BMAM) (Gál, Botzheim and Kóczy, 2008) [7]. This new approach exploits the Levenberg-Marquardt method more efficiently.

The BMA integrates its two components, the BEA and the LM method in the following way:

1. *Bacterial Mutation operation* for each individual,
2. *Levenberg-Marquardt* method for each individual,
3. *Gene Transfer operation* for a partial population.

This way the LM method is nested into the BEA, so that local search is done for every global search cycle.

Instead of applying the LM cycle *after* the *bacterial mutation* as a separate step, the modified algorithm executes several LM cycles *during* the *bacterial mutation* after *each mutational* step.

The *bacterial mutation* operation changes one or more breakpoints of the trapezoidal shaped fuzzy membership functions of a fuzzy rule base randomly, and then it tests whether the rule base obtained by this way performs better than the previous rule base or the rule bases that have been changed concurrently this way in the other so called clones. The mutation test cycle is repeated until all the parameters of the rule base have gone through the bacterial mutation.

In the mutational cycle it is possible to gain a rule base that has an instantaneous fitness value that is worse than the one in the previous or the concurrent rule bases. However, it is potentially better than those, because it is located in such a region of the search space which has a better local optimum than the other rule bases do. Corresponding to this, if some Levenberg-Marquardt iterations are executed after each bacterial mutational step, the test step is able to choose some potentially valued clones that could be lost otherwise.

In the *Bacterial Memetic Algorithm with the modified operation execution order* (BMAM), after *each mutational step* of *every single bacterial mutation iteration* several LM iterations are done. Several tests have shown it is enough to run just 3 to 5 of LM iterations per mutation to improve the performance of the whole algorithm. The usual test phase of the *bacterial mutation operation* follows after the LM iterations, and then, after the complete bacterial mutation follows the LM method that is used in the original BMA, where more, e.g. 10 iterational steps, are done with all the individuals of the population towards reaching of the local optimum. After all this the *gene transfer operation* is done.

The flowchart of the *Bacterial Memetic Algorithm* can be seen in Fig. 8. In the BMAM method the order of the steps is as follows:

1. Apply the *modified bacterial mutation operation* for each individual:
    - Each individual is selected one by one.
    - N_{Clones} copies of the selected individual are created ("clones").
    - Choose the same part or parts randomly from the clones and mutate it (except one single clone that remains unchanged during this mutation cycle).
    - *Run some Levenberg-Marquardt iterations (3 – 5).*
    - *Select the best clone and transfer its all parts to the other clones.*
    - Repeat the part choosing-mutation-*LM*-selection-transfer cycle until all the parts are mutated, *improved* and tested.
    - The best individual is remaining in the population, all other clones are deleted.
    - This process is repeated until all the individuals have gone through the *modified bacterial mutation.*
2. *Levenberg-Marquardt* method for each individual,
3. *Gene transfer operation* for a partial population.

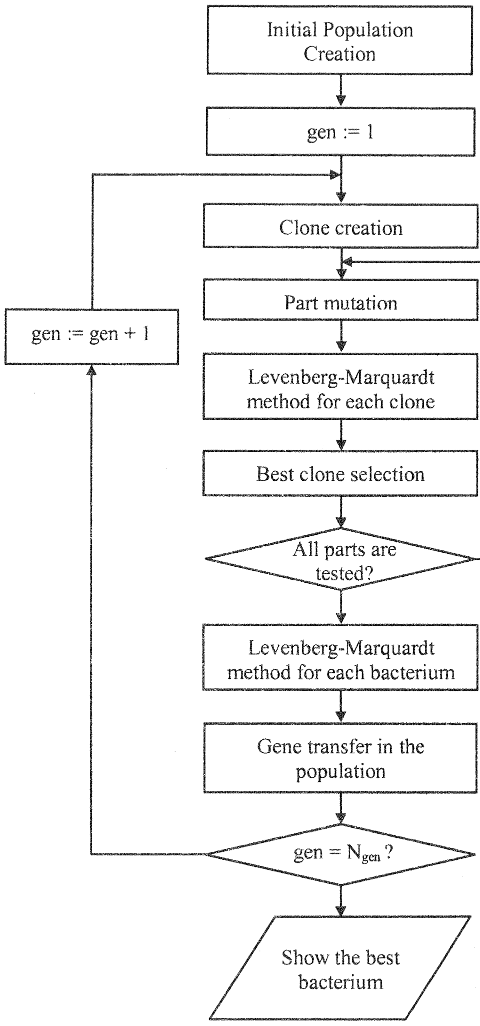

Initial Population Creation
gen := 1
Clone creation
Part mutation
gen := gen + 1
Levenberg-Marquardt method for each clone
Best clone selection
All parts are tested?
Levenberg-Marquardt method for each bacterium
Gene transfer in the population
gen = N_gen ?
Show the best bacterium

Figure 8. Flowchart of the BMAM

7. Modified Bacterial Memetic Algorithm (MBMA)

Although IBMA and BMAM perform better than the original BMA they behave in different manner in different circumstances. IBMA performed better in case of more complex fuzzy rule base while BMAM performed better in case of less complex fuzzy rule base.

Our recent work has pointed out that combining the improvements in IBMA and BMAM is beneficial (Gál, Botzheim and Kóczy, 2008) [8]. We presented a novel, improved version of the *Bacterial Memetic Algorithm* used for fuzzy rule base extraction named *Modified Bacterial Memetic Algorithm*. We modified the original

BMA in two parts. The first one is the *knot order violation handling* concerning the Levenberg-Marquardt method incorporated into the BMA, while the second one is the *operator execution order.*

The detailed steps of the MBMA are described below (Fig. 9):

1. Create the initial population: N_{Ind} individuals are randomly created and evaluated. (N_{Ind} is the number of individuals in the population.) Each individual contains N_{Fuzzy_rules} fuzzy rules encoded in the chromosome (N_{Fuzzy_rules} is the number of fuzzy rules of the desired model).
2. Apply the *Modified Bacterial Mutation operation* for each individual:
    - Each individual is selected one by one.
    - N_{Clones} copies of the selected individual are created ("clones").
    - Choose the same part or parts randomly from the clones and mutate it (except one single clone that remains unchanged during this mutation cycle).
    - *Run some Levenberg-Marquardt iterations (3–5)*
        - *Use method Swap for handling the knot order violations after each LM update.*
    - *Select the best clone and transfer its all parts to the other clones.*
    - Repeat the part choosing-mutation-*LM*-selection-transfer cycle until all the parts are mutated, *improved* and tested.
    - The best individual is remaining in the population, all other clones are deleted.
    - This process is repeated until all the individuals have gone through the *modified bacterial mutation.*
3. Apply the *Levenberg-Marquardt* method to each individual (e.g. 10 iterations per individual per generation).
    - *Use method Swap for handling the knot order violations after each LM update.*
4. Apply the *gene transfer operation* N_{Inf} times per generation:
    - Sort the population according to the fitness values and divide it in two halves. The half that contains the better individuals is called superior half while the other half is the inferior half.
    - Choose one individual (the "source chromosome") from the superior half and another one (the "destination chromosome") from the inferior half.
    - Transfer a part from the source chromosome to the destination chromosome (select the part randomly or by a predefined criterion).
    - Repeat the steps above N_{Inf} times (N_{Inf} is the number of "infections" in one generation.)

5. Repeat the procedure above from the *modified bacterial mutation* step until a certain termination criterion is satisfied (e.g. maximum number of generations).

8. Conclusions

In this paper we summarized the *bacterial type* evolutionary algorithms used for *fuzzy rule base identification.*

The *Pseudo-Bacterial Genetic Algorithm* (PBGA) offers a very simple but powerful way to extract quasi-optimal fuzzy rule bases from input-output data.

The *Bacterial Evolutionary Algorithm* (BEA) is based on PBGA and its new operator establishes relationships among the individuals and is able to change the number of the rules in the FRB.

The *Bacterial Memetic Algorithm* (BMA) combines the evolutionary approach and a local search method. It is much more successful in FRB identification than its predecessors.

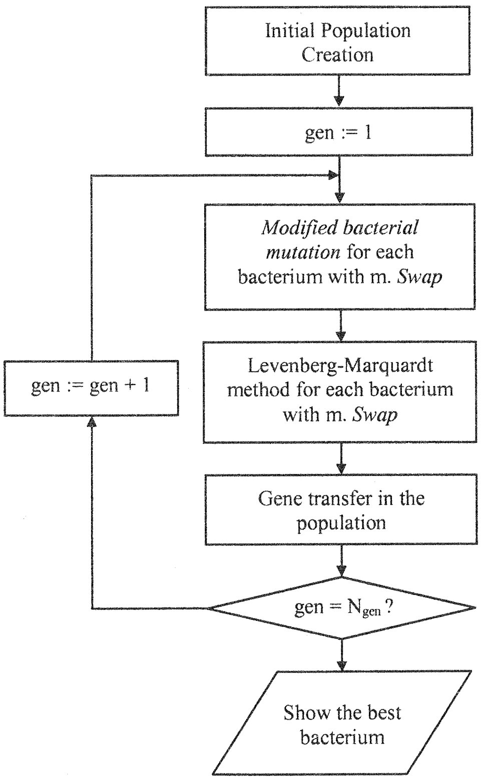

Initial Population Creation
gen := 1
Modified bacterial mutation for each bacterium with m. *Swap*
Levenberg-Marquardt method for each bacterium with m. *Swap*
Gene transfer in the population
gen = N_{gen} ?
gen := gen + 1
Show the best bacterium

Figure 9. Flowchart of the MBMA

The *Improved BMA* (IBMA) has an alternate *knot order violation handling* technique and provides improved performance rather in the case of more complex *fuzzy rule base.*

The *BMA with the modified operator execution order* (BMAM) exploits the *Levenberg-Marquardt* method (LM) more efficiently and provides improved performance rather in the case of less complex *fuzzy rule base.*

With combining the improvements of IBMA and BMAM the benefits of both methods can be utilized, because the first method increases the speed of convergence rather for higher complexity fuzzy rule bases, while the second one does the same for rule bases with lower complexity.

Previous work has confirmed that the latest version of the BMA can improve the performance of the BMA notably (up to 55 percent) in the simulated cases. While in case of very simple problem the improvement is minimal, that it is getting higher as the complexity of the fuzzy rule base increases.

Acknowledgments

This paper was supported by the Széchenyi University Main Research Direction Grant 2007, and a National Scientific Research Fund Grant OTKA T048832.

Special acknowledgment to Dr. János Botzheim (Széchenyi István University, Győr, Hungary). This paper is partly based on research done in collaboration with him.

References

[1] Botzheim, J., Cabrita, C., Kóczy, L. T., Ruano, A. E.: *Fuzzy rule extraction by bacterial memetic algorithm*, IFSA 2005, Beijing, China, (2005) pp.1563-1568

[2] Botzheim, J., Cabrita, C., Kóczy, L. T., Ruano, A. E.: *Estimating Fuzzy Membership Functions Parameters by the Levenberg-Marquardt Algorithm*, FUZZ-IEEE (2004), Budapest, Hungary pp. 1667-1672

[3] Botzheim, J., Kóczy, L. T., Ruano, A. E.: *Extension of the Levenberg-Marquardt algorithm for the extraction of trapezoidal and general piecewise linear fuzzy rules*, IEEE World Congress on Computational Intelligence, Honolulu, (2002) pp. 815-819

[4] Botzheim, J.,: *Intelligens számítástechnikai modellek identifikációja evolúciós és gradiens alapú tanuló algoritmusokkal* (in Hungarian). Ph.D. thesis, Budapest University of Technology and Economics, Faculty of Electrical Engineering and Informatics, Budapest, (2008)

[5] Cabrita, C., Botzheim, J., Gedeon, T. D., Ruano, A. E., Kóczy, L. T., Fonseca, C.: *Bacterial Memetic Algorithm for Fuzzy Rule Base Optimization*, World Automation Congress, WAC '06 (2006)

[6] Gál, L., Botzheim, J., Kóczy, L. T., Ruano, A. E.: *Fuzzy Rule Base Extraction by the Improved Bacterial Memetic Algorithm*," 6th International Symposium on Applied Machine Intelligence and Informatics Herl'any, Slovakia, January 21-22, (2008) pp. 49-53

[7] Gál, L., Botzheim, J., Kóczy, L. T.: *Improvements to the Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction*, Computational Intelligence for Measurement Systems and Applications, CIMSA 2008, Istanbul, Turkey, (2008) pp. 38-43

[8] Gál, L., Botzheim, J., Kóczy, L. T.: *Modified Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction*, 5th International Conference on Soft Computing as Transdisciplinary Science and Technology, CSTST 2008, Paris, France, (2008)

[9] Holland, J. H.: *Adaptation in Nature and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT Press, Cambridge, MA, (1992)

[10] Johanyák, Zs. Cs.: *Fuzzy rule interpolation methods and automatic system generation based on sample data*, (in hungarian), PhD thesis, University of Miskolc, (2007)

[11] Johanyák, Zs. Cs., Kovács, Sz.: *Sparse Fuzzy System Generation by Rule Base Extension*, Proceedings of the 11th IEEE International Conference of Intelligent Engineering Systems (IEEE INES 2007), Budapest, Hungary, (2007) pp. 99-104

[12] Johanyák, Zs. Cs., Kovács Sz.: *Fuzzy Rule Interpolation Based on Polar Cuts*, Computational Intelligence, Theory and Applications, Springer Berlin Heidelberg, (2006) pp. 499-511

[13] Johanyák, Zs. Cs., Tikk, D., Kovács, Sz., Wong, K. W.: *Fuzzy Rule Interpolation Matlab Toolbox - FRI Toolbox*, Proceedings of the IEEE World Congress on Computational Intelligence (WCCI'06), 15th Int. Conf. on Fuzzy Systems (FUZZ-IEEE'06), Vancouver, BC, Canada, Omnipress. ISBN 0-7803-9489-5, (2006) pp. 1427-1433

[14] Klawonn, F. and Kruse, R.: *Constructing a fuzzy controller from data*, Fuzzy Sets and Systems, Vol. 85, Issue 2, (1997) pp. 177-193

[15] Kóczy, L. T., Hirota, K.: *Size reduction by interpolation in fuzzy rule bases*, in IEEE Transactions on System, Man and Cybernetics, Vol. 27, (1997) pp. 14-25

[16] Kovács, Sz.: *Extending the Fuzzy Rule Interpolation "FIVE" by Fuzzy Observation*, Advances in Soft Computing, Computational Intelligence, Theory and Applications, Bernd Reusch (Ed.), Springer Germany, ISBN 3-540-34780-1, (2006) pp. 485-497

[17] Kovács, Sz., Kóczy, L.T.: *The use of the concept of vague environment in approximate fuzzy reasoning*, Fuzzy Set Theory and Applications, Tatra Mountains Mathematical Publications, Mathematical Institute Slovak Academy of Sciences, vol.12., Bratislava, Slovakia, (1997) pp.169-181

[18] Mamdani, E. H., Assilian, S.: *An experiment in linguistic synthesis with a fuzzy logic controller*, Int. J. Man-Mach. Stud., 7 (1975)pp. 1–13

[19] Marquardt, D.: *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, SIAM J. Appl. Math., 11, (1963) pp. 431-441

[20] Moscato, P.: *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, California, USA (1989)

[21] Nawa, N. E., Hashiyama, T., Furuhashi, T., Uchikawa, Y.: *A study on fuzzy rules discovery using pseudo-bacterial genetic algorithm with adaptive operator*, Proceedings of IEEE Int. Conf. on Evolutionary Computation, ICEC'97 (1997)

[22] Nawa, N. E., Furuhashi, T.: *Fuzzy Systems Parameters Discovery by Bacterial Evolutionary Algorithms*, IEEE Transactions on Fuzzy Systems 7 (1999) pp. 608-616

[23] Sugeno, M., Kang, K. T.: *Structure Identification of Fuzzy Model*, Fuzzy Sets and Systems, Vol. 28 (1988) pp. 15-33

Dual-Rail Asynchronous Implementation of a VLSI Processor for Digital Cellular Neural Networks

Péter Keresztes, Timót Hídvégi

Department of Automation, Széchenyi István University
H-9026 Győr, Egyetem tér 1, Hungary
Email: keresztp@sze.hu, hidvegi@sze.hu

Abstract: An emulated digital Cellular Neural Network (CNN) Universal Machine chip was designed by a research group of Analogic and Neural Computing Systems Laboratory of HAS, almost 10 years ago. During the design period of the chip, particular attention had to be paid to the clocking and control system of the chip, since each architecture element, placed on a large silicon area chip (1 cm^2), operated in a totally synchonous mode, using a single global clock. The designer's manipulations for eliminating the consequences of the too high propagation delay of long interconnections resulted in a relatively large chip, and a significant part of the chip was totally superfluous from the point of view of logical operation. The clock rate was limited at a lower level than it would have been possible without the long interconnections. So the development of 'CASTLE' CNN processor array showed the most significant problems of the submicron VLSI, which arose from the too long interconnections of big size chips.

The elimination of clocking problems and the re-designing of the CASTLE architecture using delay-insensitive, self-synchronized, asynchronous logic elements were performed. The result of the re-designing work, which is presented in this paper, is a clockless, totally asynchronous architecture. The combination of the *DUAL-RAIL logic* and the methods of the well known *4-phase asynchronous inter-register communication* were chosen. Obviously the timing and control unit of the synchronous version is unnecessary in the asynchronous one. So the tasks of re-designing several synchronous units to their dual-rail versions consisted in designing an application specific dual-rail arithmetic unit with feedback and dual-rail multidirection FIFOs. Several new dual rail logic elements were introduced, which were modeled and simulated as follows:

- Two-input dual-rail register with a common acknowledge output and priority order for inputs
- Two- or more-input dual rail register with independent acknowledge outputs and dual-rail selection inputs
- Dual-rail register with CLEAR single-rail control inputs, which enable setting the register into so called DATA-TOKEN and BUBBLE states.

The lecture presents the most interesting parts of the design process.

Keywords: *emulated digital CNN-UM, clockless processors, Dual-rail logic, self-synchronization, delay-insensitive logic systems*

1. Introduction: Experiments of designing a synchronous CNN processor array

The architecture of a VLSI chip containing a sub-array of digital CNN processors was published in 1999. [2], [11]. The chip operated with a two-phase global clock, so it demanded a very careful design of the central timing and control unit (TAC), which were based on a synchronous FSM. The cause of the difficulties is that the time delay of the relatively long metal interconnections exceeded the delay time of the logic gates in the applied submicron CMOS technology. A relatively large silicon area had to be sacrificed for the synchronity of the clock and the control signals in the different points of the chip.

The problem of clocking, which had to be solved 10 years ago, became the primary obstacle to the future development of VLSI technology. There are several concepts and methods intended to solve the clocking problem, as follows:

- The system consists of *smaller synchronous units*, which communicate *asynchronously* [3], [5],[6]
- Application of *Delay Locked Loop* (DLL) circuits to insert the required value additional delay into the paths of the clock signal [10].
- Application of *asynchronous units* which communicate *asyncronously* [1],[4], [7], [8], [9].

The re-designing of the former CNN processor architecture 'CASTLE' will be discussed in detail in this paper.

2. Dual-Rail asynchronous digital circuits and systems

The dual-rail asynchronous circuits and systems are based on two principles, as follows:

- The principle of 4-phase hand-shake communication.

  The background of the classical four-phase handshaking asynchronous communication is that the data on the output of the transmitter (sender) has to be ready before the signal '*request*' rises. The idea is that the code of the output datum itself contains the *request*. The dual-rail codes can contain it. The application of this principle leads to the asynchronous inter-register communication of the dual-rail sytems.

- The principle of logic completeness.

  It is supported by the dual-rail code of logical variables. Logic completeness means that a function unit should only give a valid output datum if it has valid data on all of its inputs, and should only switch its output to invalid, if all its inputs have turned invalid. It follows from this that a logic decision is valid and transmissible only, if all the premises necessary for it are valid. This enables hazard-free logic implementations. A network consisting of such units bears the ability of self-synchronization, that is the data validated by specific, valid input data in a given time will not mix with those validated by the input data of a different time.

If two wires are ordered to one logic variable, it is possible to distinguish valid and invalid logic values. The value of the variable is valid, if the levels of wires are (L H) or (H L). The value of the variable is invalid, if both wires are at low level (L L). The convention is that a valid datum contains a request.

Dual-rail (DR) registers can be easily built up from the C-elements, which are introduced by Müller at the end of 50'ths. The simplest Müller element, the C-2 is shown in Figure 1. The DR-LATCH consisting of C-2 elements and the symbol of a DR-register are shown in the Figure 2.

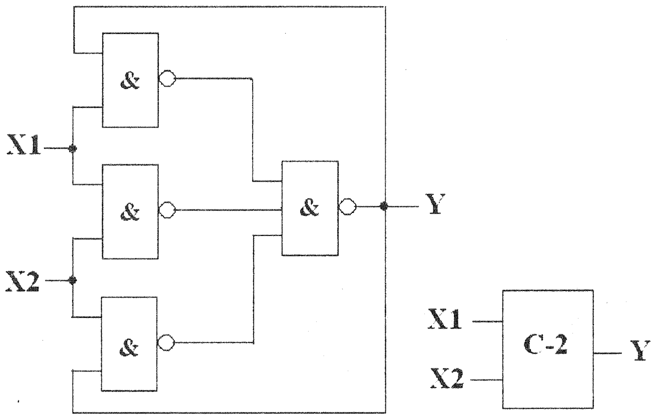


Figure 1. The scheme of Muller C-2 and its symbol

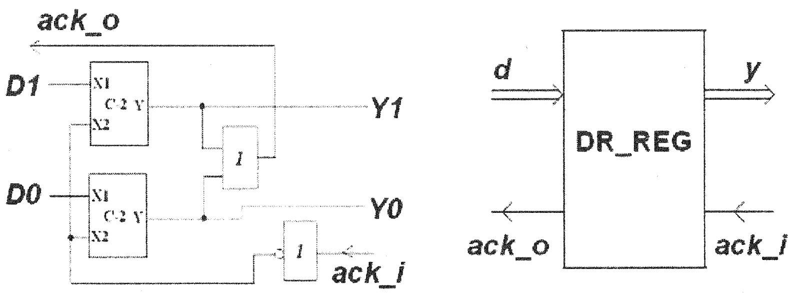


Figure 2. Scheme of a Dual-Rail latch and the symbol of Dual-Rail register

3. R-T level architecture of the synchronous CASTLE processor

The name of the architecture of the core processor of the fully synchronous array was a made-up name 'CASTLE'.

The operation of CASTLE was derived from the "full-signal-range model" of the CNN state-equations using the simplest forward Euler integration form:

$$x_{ij}(n+1) = \sum_{C(kl)\in N_r(i,j)} A'_{ij,kl} * x_{kl}(n) + g_{ij}$$

$$g_{ij} = \sum_{C(kl)\in N_r(i,j)} B'_{ij,kl} * u_{kl}(n) + h * z_{ij}$$

Here h is the time step, u_{kl} are the inputs, $x_{ij}(n+1)$ is the next state of the cell(i,j) before truncation, $x_{kl}(n)$ are the current states of the neighbouring cells, g_{ij} and z_{ij} are constants, supposing a constant input, and A' and B' are the so called modified template matrices. The R-T level architecture is shown in Figure 3.

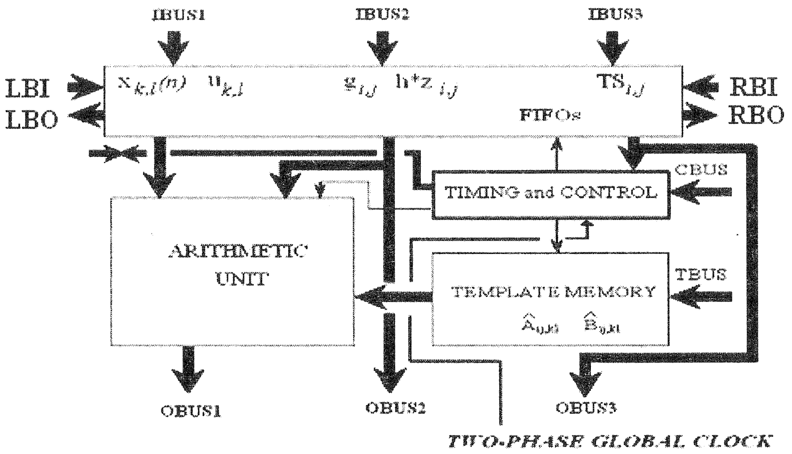


Figure 3. The original, global clock CASTLE architecture

The sub-unit TIMING and CONTROL controls not only one processor, but all the processors on the chip. This is the origin of most synchronization difficulties.

In Figure 4. it can be seen that the state variables (**IBUS1**), additive constants (**IBUS2**), and the bit-vectors of template selection (**IBUS3**) are continuously shifting in FIFO memories. The buses **LBI, LBO**, **RBI** and **RBO** are used for the communication with the left-side and right-side neighbouring processors. The FIFOs of state variables are particularly complicated circuits. These are so-called multi-direction FIFOs with horizontal and vertical shifting, combined with both horizontal and vertical rotation.

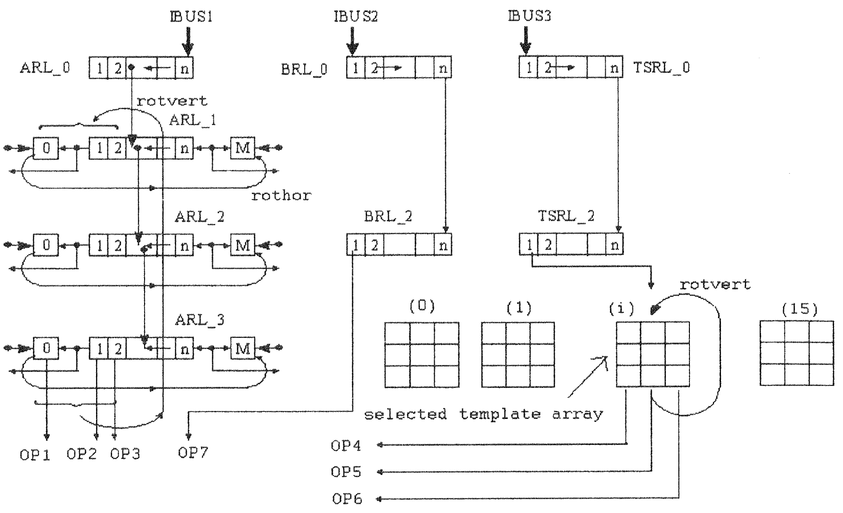


Figure 4. Traditional FSM controlled multidirection FIFOs of CASTLE

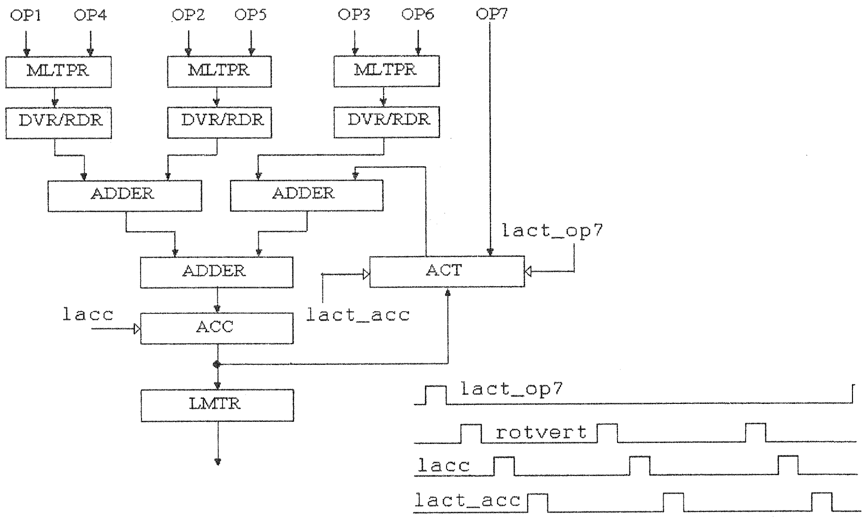


Figure 5. The FSM controlled arithmetic unit of the CASTLE processor

The left-side 9 cells with indexes 0, 1, 2 of the three state-FIFOs take part in the next-state calculation, along with the left-side cell of the constant-FIFO and the template values selected by the left-side cell of template-select FIFO. Figure 5. shows the three-multiplier arithmetic unit, for the inputs of which the state-FIFO rotate the state- and template-operands. There is a feedback via an ACCUMULATOR and a TEMPORARY

ACCUMULATOR. The latter receives the constant to be added to the sum-of- products. The control signals of the register-tranfers are marked in the figure, and a timing diagram shows the three-cycle rotation (*rotvert, lacc, lact_acc*), preceded by the loading of an additive constant (*lact_op7*).

4. Architecture elements of a Dual-Rail asynchronous CNN processor

A DR asynchronous R-T level system can be considered as a composition of DR-stages. A DR-stage consists of DR-registers, which communicate with the registers of other stages in the way detailed above, and DR function units, the operation of which fulfils the criteria of logic completeness.

Figure 6. demonstrates how the R-T level DR units are connected to each other, using DR buses and acknowledge signals.

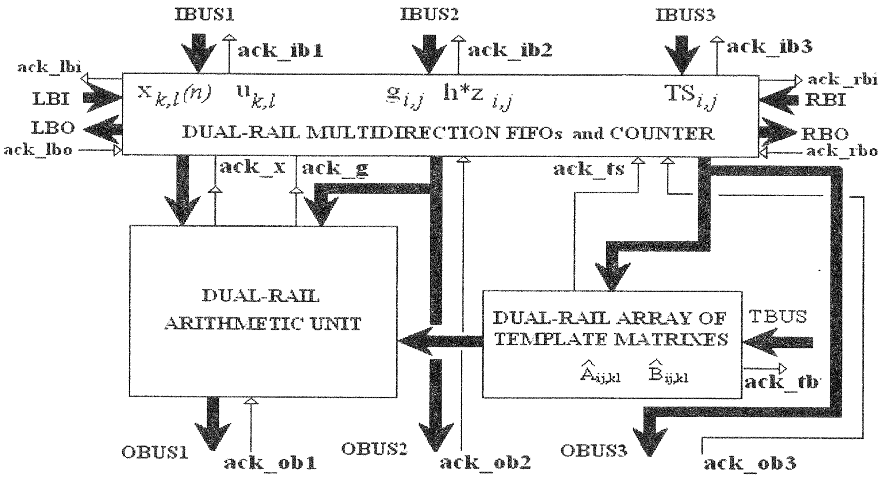

IBUS1
ack_ib1
IBUS2
ack_ib2
IBUS3
ack_ib3
ack_lbi
LBI
LBO
ack_lbo
$x_{k,l}(n)$ $u_{k,l}$ $g_{i,j}$ $h^*z_{i,j}$ $TS_{i,j}$
DUAL-RAIL MULTIDIRECTION FIFOs and COUNTER
ack_rbi
RBI
RBO
ack_rbo
ack_x
ack_g
ack_ts
DUAL-RAIL ARITHMETIC UNIT
DUAL-RAIL ARRAY OF TEMPLATE MATRIXES
$\hat{A}_{ij,kl}$ $\hat{B}_{ij,kl}$
TBUS
ack_tb
OBUS1
ack_ob1
OBUS2
ack_ob2
OBUS3
ack_ob3

Figure 6. The architecture of DR-CASTLE processor

The two most interesting DR-stages are presented in this paper. They are interesting from the point of view that several new solutions had to be invented, which had not existed until then in the literature. These are as follows:

- DR-FIFO compositions for multidirection shifting and rotations
- CNN arithmetic unit consisting of DR multipliers and adders and DR registers

4.1. Dual-Rail, multidirection FIFOs for asynchronous CNN processor

Sparso established the scheme of how to build up a shift-register from DR-latches. [9]. He called the state of a latch BUBBLE, when its output is invalid, and output *ack_out* and input *ack_in* are low. In the BUBBLE state the latch can be immediately loaded from its valid input. The opposite state is called TOKEN. There are two variants of the

TOKEN state. The fisrt is called DATA-TOKEN, in which the *Y* output is valid, *ack_out* is high, and *ack_in* low. The second one is called EMPTY-TOKEN, when the *Y* output is invalid, *ack_in* are high, and *ack_out* is low. The different state registers and a part of a DR-shift-register is shown in Figure 7.

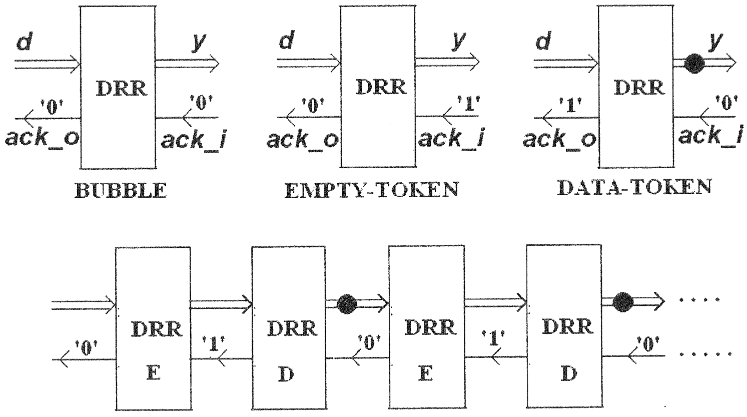


Figure 7. The states of DR-registers and the initial state of a register-chain

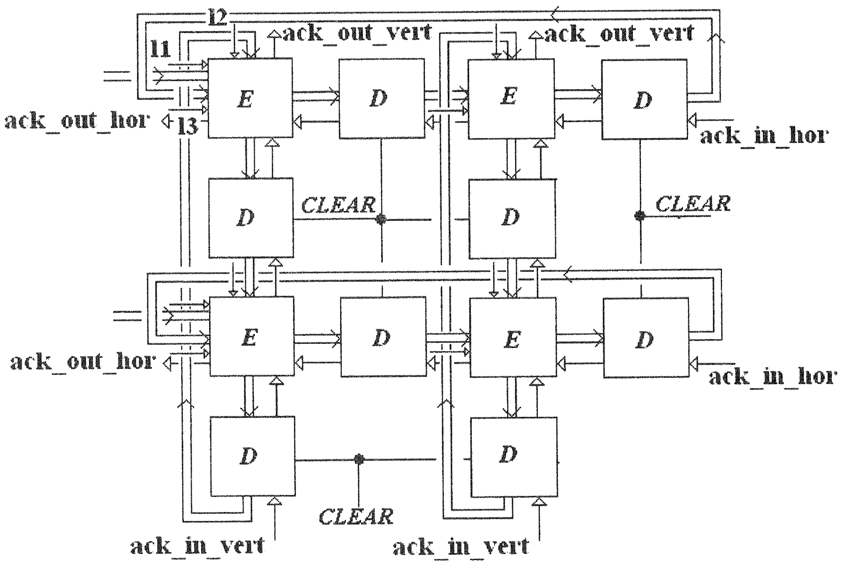


Figure 8. A simplified model of multidirection DR-FIFO

A chain of DR-LATCHES can be considered an *n*-stage shift-register, if

- there are 2n latches chained,
- the initial state of the chain has to be special, namely an EMPTY-TOKEN latch has to be followed by a DATA-TOKEN one.

Starting the DR-SR from this state with a valid input and rising the last *ack_in*, BUBBLE state runs along the chain from the last latch to the first one. So all the data will be shifted and the state of each latch will change.

4.1.1. DR-register with clear

The DR-register equipped with CLEAR input is a necessary component of DR-FIFO. The CLEAR is a single-rail control signal, H-level of which results in valid intial data on the output. This initial data in our case is the valid code of the integer 0. If signal CLEAR is raised, the initial data is stored, and the register will be in state DATA-TOKEN. It can be seen in Figure 8 that each second register is an instance of this component. The symbol of DR-register with CLEAR is shown in Figure 9., and in Figure 10. the scheme of its element is presented. The VHDL behavioural description of this unit is given in the Appendix.

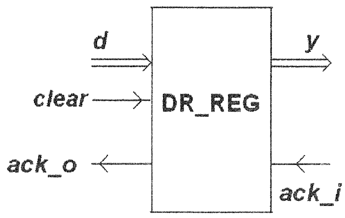


Figure 9. Symbol of DR-register with clear.

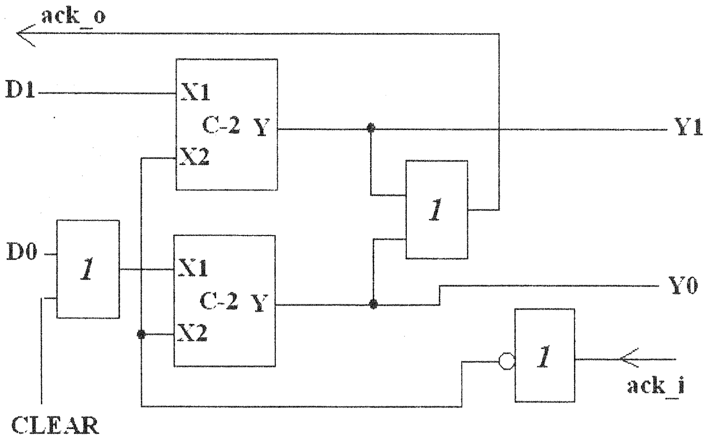


Figure 10. Scheme of DR-LATCH with clear

4.1.2. Multi-input DR-register with DR selectors (Figure 11.)

Multiple-input DR-registers with selectors are also needed for the multidirection FIFOs. A selector-wire, which is a 'single-rail' belongs to each data-input. The condition of storing the value of a given data-input is that the selector value belonging to the data-input is 'H'. It means that the 'H' is the valid-value on a single-rail. The behavioural description can be seen in the Appendix.

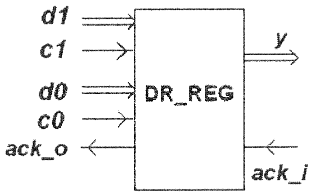


Figure 11. Scheme of two-input DR-REG with single-rail selector wires

4.1.3. Multi-input DR-register with priority order for inputs (Figure 12.)

In the case of the third type of a multiple-input register a priority order is defined for the inputs. For the two-input DR-register given in Figure 12. input **d1** dominates the input **d2**. The VHDL behavioural description of this unit is given in the Appendix.

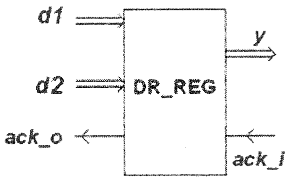


Figure 12. Scheme of two-input DR-REG with priority order for the inputs

4.2. Dual-Rail arithemetic unit with feedback

The arithmetic unit is a classical DR stage, consisiting of an arithmetic core and registers. The core is a compostion of DR multipliers and adders, and these components are dual-rail combinational networks. The input DR-registers make the asynchronous communication with the outputs of the multidirection FIFOs. The chain of three DR registers constitutes the feedback for accumulating the sum of subproducts. Sparso showed that for a DR-ring without a dead-lock the presence of at least three latches is needed. The third register of the loop receives not only the accumulated sum of the subproducts, but in the initial step of each cycle the additive constant as well. This register has two data-input with a common acknowledge signal. One of the data-input is connected to FIFO cell which stores the additive constant. Since a valid code of this input starts a new calculation cycle, it has a priority over the other data-input, which is the closing point of the loop.

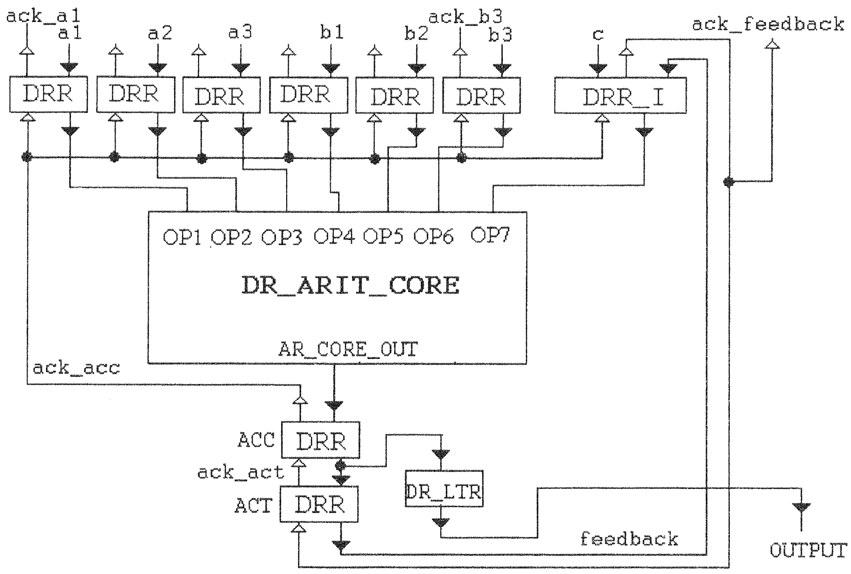


Figure 11. The scheme of the Dual-Rail arithmetic unit

5. Conclusions

The original synchronous architecture of CASTLE can easily be transformed into a Dual-Rail asynchonous version. Above the well known elements, the elaboration of several new Dual-Rail logic elements is necessary. They are as follows:

- Dual-rail register with CLEAR single-rail control input, which enables setting the register into so called DATA-TOKEN state.
- Two- or more-input dual-rail register with sigle-rail selection inputs
- Two input dual-rail register with priority order for inputs

Using these new RT elements, fully asynchronous CNN processor arrays can be realized, and the results can be applied in hardware-implementation of other types of neural networks too. Designing Dual-Rail asynchronous processors for digital VLSI implementations of neural networks helps avoid the clocking problem of submicron VLSI technology.

References

[1] Fant, K. M, Brandt, S. A.: *Null Convention Logic: A complete and consistent logic for asynchronous digital circuit Synthesis,* International Conference of Application Specific Sytems, Architectures and Processor, (1996) pp. 261-273

[2] Keresztes, P., Zarándy, Á., Roska, T., Szolgay, P., Bezák, T., Hídvégi, T., Jónás, P. and Katona, A.: *An emulated digital CNN implementation*, Journal of VLSI Signal Processing Volume 23, No 2/3, (1999) pp. 291-303

[3] Lavagno, L., Sangiovanni-Vincentelli, A.: *Algorithms for synthesis and testing of asynchronous circuits,* Kluwer Academic Publishers, (1993) pp.18-25

[4] Muller, D. E., Bartky, W. C.: *A theory of asynchronous circuits*, Annals of Computing Laboratory of Harward University, (1959) pp.204-243

[5] Nowick, S. M., Dill, D. L.: *Automatic synthesis of locally clocked asynchronous state machines*, Proc. of the International Conference on Computer Aided Design, November, (1991)

[6] Saphiro, D. M.: *Globally-Asynchronous, Locally Synchronous Systems*, PhD Thesis, Stanford University, October (1984)

[7] Smith, S. C., DeMara, R. F., Yuan, J. S., Ferguson, D. , Lamb, D.: *Optimization of Null convention self timed circuits*, INTEGRATION, the VLSI Journal 37 (2004) pp. 135-165

[8] Sutherland, I. E.: *Micropipelines*. Communications of the ACM, June, 1989. Turing Award Lecture

[9] Sparso, J., Furber, S.: *Principles of asynchronous circuit design – A system perspective,* Kluwer Academic Publisher, (2001) pp. 11-13

[10] Xilinx: *Using Delay-Locked Loops in Spartan-II/IIE FPGAs*, Application Note, http://www.xilinx.com/support/documentation/application_notes/xapp174.pdf

[11] Zarándy, Á, Keresztes, P., Roska, T., and Szolgay, P.: *An emulated digital architecture implementing the CNN Universal Machine*, Proc. of the fifth IEEE Int. Workshop on Celluler Neural Networks and Their, Applications, London, (1998) pp. 249-252

Appendix

```
package DR_PACK is

  type WIRE  is (L, H);
  subtype DR_REAL is real range -10.000000 to +9.999999;

end  DR-PACK;


-- behavioural description of DR-REGISTER with CLEAR

library work;
use work.DR_PACK.all;
entity DR_REG_cl is
 port ( clear : in bit;
        d : in DR_REAL;
        y : inout  DR_REAL;
        ack_i : in bit;
        ack_o : out bit);
end;

architecture BEH of DR_REG_cl is
constant td : time := 1 ns;
constant REALNULL : DR_REAL := -10.000;
begin

process(clear, d, ack_i, y)
 begin
  if clear = '1' then
           y <= 0.0 after td;
           ack_o <= '1' after td;

  elsif clear = '0' and Y = REALNULL and ack_i = '0' and
           d /= REALNULL then
           y <= D after td;
           ack_o <= '1' after td;

  elsif clear = '0' and Y /= REALNULL and ack_i = '1' and
            d = REALNULL then
            y <= REALNULL after td;
            ack_o <= '0' after td;
  end if;
 end process;
end BEH;
```

```
-- behavioural description of TWO-INPUT DR-REGISTER with
SELECTOR wires


library work;
use work.DR_PACK.all;
entity DR_REG_D1_D2 is
 port ( d1, d2 : in DR_REAL;
        y : inout  DR_REAL;
        ack_i : in bit;
        ack_o : out bit;
        c1, c2 : in WIRE);
end;

architecture BEH of DR_REG_D1_D2 is
constant td : time := 1 ns;
constant REALNULL : DR_REAL := -10.000;
begin
process(d1, d2, c1, c2, ack_i, y)
   begin
    if y = REALNULL and ack_i = '0' and d1 /= REALNULL and
       c1 = H then
             y <= d1 after td;
             ack_o <= '1' after td;

      elsif y = REALNULL and ack_i = '0' and d2 /= REALNULL
and
             c2 = H then
             y <= d2 after td;
             ack_o <= '1' after td;
      elsif y /= REALNULL and ack_i = '1' and d1 = REALNULL
and
             d2 = REALNULL and
             c1 = L and c2 = L then
             y <= REALNULL after td;
             ack_o <= '0' after td;
      end if;
  end process;
end BEH;


-- behavioural description of TWO-INPUT DR-REGISTER with
priority order  -- for the inputs

library work;
use work.DR_PACK.all;
entity DR_REG_DD is
 port ( d1 : in DR_REAL;
        d2 : in DR_REAL;
        y : inout  DR_REAL;
        ack_i : in bit;
        ack_o : out bit);
end;

architecture BEH of DR_REG_DD is
constant td : time := 2 ns;
constant REALNULL : DR_REAL := -10.000;
begin
process(d1, d2, ack_i, y)
   begin
    if y = REALNULL and ack_i = '0' and d1 /= REALNULL then
          y <= d1 after td;
          ack_o <= '1' after td;

      elsif y /= REALNULL and ack_i = '0' and d1 /=
REALNULL then
            y <= d1 after td;
            ack_o <= '1' after td;

      elsif y = REALNULL and ack_i = '0' and d2 /= REALNULL
then
            y <= d2 after td;
            ack_o <= '1' after td;

      elsif y /= REALNULL and ack_i = '1' and d1 = REALNULL
and
            d2 = REALNULL then
            y <= REALNULL after td;
            ack_o <= '0' after td;
      end if;
  end process;
end BEH;
```

Rollover Prevention Of A Heavy Vehicle Via TP Model Based $\mathcal{H}_\infty$ Control Design Approach

Szabolcs Nagy, Péter Baranyi and Péter Gáspár

Computer and Automation Research Institute, Hungarian Academy of Sciences

Abstract: This paper is focusing on a novel nonlinear control design approach to solve the rollover prevention problem of a heavy vehicle. To provide a heavy vehicle with the ability to resist overturning moments generated during cornering, a combined yawÂroll model including the roll dynamics of unsprung masses is studied. This model is nonlinear with respect to the velocity of the vehicle. In our model the velocity is handled as an LPV scheduling parameter. The Linear Parameter-Varying model of the heavy vehicle is transformed into a proper polytopic form by Tensor Product model transformation. The $\mathcal{H}_\infty$ gain-scheduling based control is immediately applied to this form for the stabilization. The effectiveness of the designed controller is demonstrated by numerical simulation.

Keywords: TP, LPV, $\mathcal{H}_\infty$ Control, Vehicle control

1. Introduction

Roll stability is determined by the height of the center of mass, the track width and the kinematic properties of the suspensions. The problem with heavy vehicles is a relatively high mass center and narrow track width. When the vehicle is changing lanes or trying to avoid obstacles, the vehicle body rolls out of the corner and the center of mass shifts outboard of the centerline, and a destabilizing moment is created.

In the literature there are many papers with different approaches on the active control of the heavy vehicles to decrease the rollover risk. Three main schemes concerned with the possible active intervention into the vehicle dynamics have been proposed: active anti roll bars, active steering and active brake. The control design is usually based on linear time invariant (LTI) models and linear approaches. The forward velocity is handled as a constant parameter in the yaw-roll model; however, velocity is an important parameter as far as roll stability is concerned [1–3].

Modern control theory mainly focuses on analysis and control design based on Linear Matrix Inequalities (LMI) and Linear Parameter Varying (LPV) system models. This is a widespread approach to achieve guaranteed robust and efficient nonlinear control results [4,5].

In this paper, a combined yaw-roll model including the roll dynamics of unsprung masses is studied [1]. This model is nonlinear with respect to the velocity of the vehicle. Thus, in our model velocity is handled as an LPV scheduling parameter. The controller based on this LPV model is adjusted continuously by measuring the vehicle velocity in real-time. Controller design uses modern LMI based approach to guarantee the $\mathcal{H}_\infty$ gain.

The control design is based on the following steps:

1. The Linear Parameter-Varying (LPV) dynamic model of the heavy vehicle model is given.
2. We apply the Tensor Product (TP) model transformation to transform the LPV model to a TP-type convex polytopic model form. The TP model transformation is a recently proposed automatically executable numerical method. It is developed for controller design involving LPV model representation and linear matrix inequality (LMI) based control design. It is capable of numerically generate different convex polytopic forms of LPV dynamic models, whereupon LMI-based design is immediately be executable. It is important to emphasize that in many cases, the analytical derivation of these polytopic models needs very sophisticated and time consuming derivations
3. Then we apply the LMI theorems of $\mathcal{H}_\infty$ gain-scheduling to design the controller.

The paper is organized as follows: Section 2 defines the LPV model form, its representation in TP model form and shows the link to fuzzy systems. Section 3 first introduces the LPV model of the heavy vehicle and presents its TP model representations, then presents the proposed controller design method. Section 4 shows the simulation results. Finally we give a short conclusion at the end of the paper.

2. Basic Concepts

2.1. Nomenclature

- $\{a,b,\ldots\}$: scalar values;
- $\{\mathbf{a},\mathbf{b},\ldots\}$: vectors;
- $\{\mathbf{A},\mathbf{B},\ldots\}$: matrices;
- $\{\mathcal{A},\mathcal{B},\ldots\}$: tensors;
- $\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$:vector space of real valued $(I_1\times I_2\times\cdots\times I_N)$-tensors.

- Subscript defines lower order: for example, an element of matrix **A** at row-column number i,j is symbolized as $(\mathbf{A})_{i,j} = a_{i,j}$. Systematically, the ith column vector of **A** is denoted as $\mathbf{a}_i$, i.e. $\mathbf{A} = \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots\end{bmatrix}$.
- $(\cdot)_{i,j,n}, \ldots$: are indices;
- $(\cdot)_{I,J,N}, \ldots$: index upper bound: for example: $i = 1..I$, $j = 1..J$, $n = 1..N$ or $i_n = 1..I_n$.
- $\mathbf{A}^+$: the pseudo inverse of matrix **A**.
- $\mathbf{A}_{(n)}$: n-mode matrix of tensor $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$;
- $\mathcal{A} \times_n \mathbf{U}$: n-mode matrix-tensor product;
- $rank_n(\mathcal{A})$: n-mode rank of tensor $\mathcal{A}$, that is $rank_n(\mathcal{A}) = rank(\mathbf{A}_{(n)})$;
- $\mathcal{A} \boxtimes_{n=1}^{N} \mathbf{U}_n$: multiple product as $\mathcal{A} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 .. \times_N \mathbf{U}_N$;

Detailed discussion of tensor notations and operations is given in [6].

2.2. Definitions

2.2.1. Linear Parameter-Varying state-space model

Consider the following parameter-varying state-space model:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{B}(\mathbf{p}(t))\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{C}(\mathbf{p}(t))\mathbf{x}(t) + \mathbf{D}(\mathbf{p}(t))\mathbf{u}(t),\end{aligned} \tag{1}$$

with input $\mathbf{u}(t)$, output $\mathbf{y}(t)$ and state vector $\mathbf{x}(t)$. The system matrix

$$\mathbf{S}(\mathbf{p}(t)) = \begin{pmatrix}\mathbf{A}(\mathbf{p}(t)) & \mathbf{B}(\mathbf{p}(t)) \\ \mathbf{C}(\mathbf{p}(t)) & \mathbf{D}(\mathbf{p}(t))\end{pmatrix} \in \mathbb{R}^{O\times I} \tag{2}$$

is a parameter-varying object, where $\mathbf{p}(t) \in \Omega$ is time varying N-dimensional parameter vector, and is an element of the closed hypercube

$$\Omega = [a_1,b_1] \times [a_2,b_2] \times \cdots \times [a_N,b_N] \subset \mathbb{R}^N.$$

$\mathbf{p}(t)$ can also include some elements of $\mathbf{x}(t)$ in which case the model is a quasi-Linear Parameter-Varying model.

2.2.2. Finite element TP model form of quasi LPV models

$\mathbf{S}(\mathbf{p}(t))$ is given for any parameter $\mathbf{p}(t)$ as the combination of LTI system matrices $\mathbf{S}_r$, $r = 1,\ldots,R$. Matrices $\mathbf{S}_r$ are also called *vertex systems*. Therefore, one can define weighting functions $w_r(\mathbf{p}(t)) \in [0,1] \subset \mathbb{R}$ such that matrix $\mathbf{S}(\mathbf{p}(t))$ can be expressed as parameter dependent weighted combination of system matrices $\mathbf{S}_r$. The explicit form of the TP model in terms of tensor product becomes:

$$\begin{pmatrix}\dot{\mathbf{x}}(t)\\ \mathbf{y}(t)\end{pmatrix} = \mathcal{S} \mathop{\boxtimes}_{n=1}^{N} \mathbf{w}_n(p_n(t)) \begin{pmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{pmatrix}. \tag{3}$$

Here, row vector $\mathbf{w}_n(p_n) \in \mathbb{R}^{I_n}$ $n = 1,\ldots,N$ contains the one variable weighting functions $w_{n,i_n}(p_n)$. Function $w_{n,j}(p_n(t)) \in [0,1]$ is the j-th one variable weighting function defined on the n-th dimension of Ω, and $p_n(t)$ is the n-th element of vector $\mathbf{p}(t)$. I_n $(n = 1,\ldots,N)$ is the number of the weighting functions used in the n-th dimension of the parameter vector $\mathbf{p}(t)$. The $(N+2)$-dimensional tensor

$$\mathcal{S} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N\times O\times I}$$

is constructed from LTI vertex systems $\mathbf{S}_{i_1 i_2 \ldots i_N} \in \mathbb{R}^{O\times I}$. Finite element TP model means that the LTI components of the model is bounded. For further details we refer to [7, 8].

2.2.3. Convex TP model form of qLPV model

The convex combination of the LTI vertex systems is ensured by the conditions:

Definition 1 *The TP model (3) is convex if:*

$$\forall n \in [1,N], i, p_n(t) : w_{n,i}(p_n(t)) \in [0,1]; \tag{4}$$

$$\forall n \in [1,N], p_n(t) : \sum_{i=1}^{I_n} w_{n,i}(p_n(t)) = 1. \tag{5}$$

This simply means that $\mathbf{S}(\mathbf{p}(t))$ is within the convex hull of the LTI vertex systems $\mathbf{S}_{i_1 i_2 \ldots i_N}$ for any $\mathbf{p}(t) \in \Omega$.

2.2.4. Link to the TS fuzzy model

The TP model (3) is equivalent with the transfer function of the widely used type of Takagi-Sugeno (TS) fuzzy model. When we have fuzzy rules

$$\text{IF } p_1 \text{ is } A_{1,i_1} \text{ AND } p_2 \text{ is } A_{2,i_2} \ \ldots \text{ AND } p_N \text{ is } A_{N,i_N}$$
$$\text{THEN } \mathbf{S} \text{ is } \mathbf{S}_{i_1 i_2 \ldots i_N}$$

for all combinations of $i_n = 1 \dots I_n$ $(n = 1 \dots N)$, then the transfer function (with product sum gravity defuzzyfication) is

$$\mathbf{S}(\mathbf{p}) = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} w_{1,i_1}(p_1) w_{2,i_2}(p_2) \dots w_{N,i_N}(p_n) \mathbf{S}_{i_1 i_2 \dots i_N},$$

where $w_{n,i_n}(p_n)$ function is the normalized membership-function of the fuzzy antecedent A_{n,i_n} and $\mathbf{S}_{i_1 i_2 \dots i_N}$ is the consequent of the rule. This model is the tensor product model which is given in (3) with a more compact notation. The convexity constraint of the previous subsection means that the antecedents form a Ruspini-partition.

2.2.5. Link to the polytopic form

In order to have a direct link between the TP model form and the polytop formula, we define the following index transformation:

Definition 2 (Index transformation) *Let*

$$\mathbf{S}_r = \begin{pmatrix} \mathbf{A}_r & \mathbf{B}_r \\ \mathbf{C}_r & \mathbf{D}_r \end{pmatrix} = \mathbf{S}_{i_1, i_2, \dots, i_N},$$

where $r = \text{ordering}(i_1, i_2, \dots, i_N)$ $(r = 1 \dots R = \prod_n I_n)$. *The function "ordering" results in the linear index equivalent of an N dimensional array's index* $i_1, i_2, \dots, i_N$, *when the size of the array is* $I_1 \times I_2 \times \cdots \times I_N$. *Let the weighting functions be defined according to the sequence of r:*

$$w_r(\mathbf{p}(t)) = \prod_n w_{n,i_n}(p_n(t)).$$

By the above index transformation one can write the TP model (3) in the typical polytopic form of:

$$\mathbf{S}(\mathbf{p}(t)) = \sum_{r=1}^{R} w_r(\mathbf{p}(t)) \mathbf{S}_r. \tag{6}$$

Remark: Note that the LTI systems $\mathbf{S}_r$ and $\mathbf{S}_{i_1, i_2, \dots, i_N}$ are the same, only their indices are modified, therefore the convex hull defined by the LTI systems is the same in both forms.

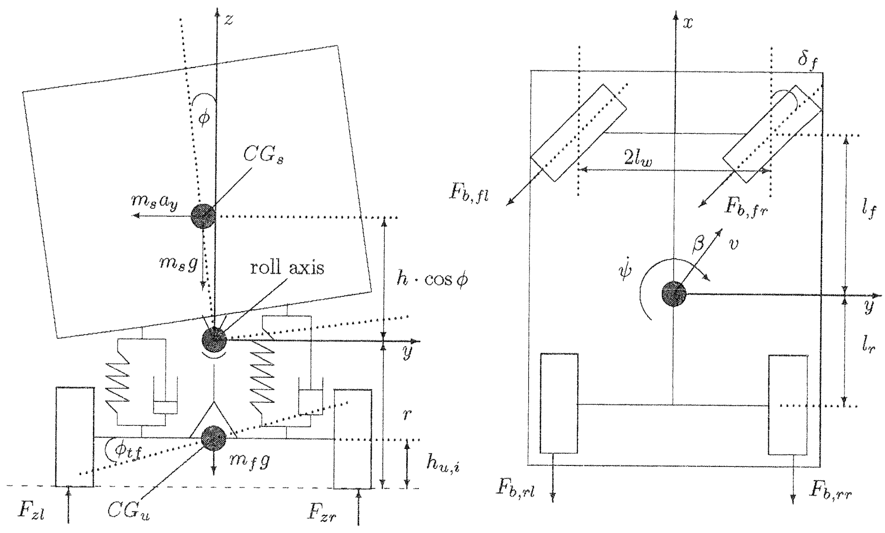


Figure 1: Rollover vehicle model

3. Control of the Heavy Vehicle

3.1. Heavy vehicle model

3.1.1. Analytic model of the heavy vehicle

Figure 1 illustrates the combined yaw-roll dynamics of the vehicle modeled by a three-body system, in which m_s is the sprung mass, $m_{u,f}$ is the unsprung mass at the front including the front wheels and axle, and $m_{u,r}$ is the unsprung mass at the rear with the rear wheels and axle.

The conditions of yaw-roll model used in control design are considered. It is assumed that the roll axis is parallel to the road plane in the longitudinal direction of the vehicle at a height r above the road. The location of the roll axis depends on the kinematic properties of the front and rear suspensions. The axles of the vehicle are considered to be a single rigid body with flexible tires that can roll around the center of the roll. The tire characteristics in the model are assumed to be linear. The effect caused by pitching dynamics in the longitudinal plane can be ignored in the handling behavior of the vehicle. The effects of

$$mv(\dot{\beta} + \dot{\psi}) - m_s h\ddot{\phi} = Y_\beta \beta + Y_{\dot{\psi}}\dot{\psi} + Y_{\delta_f}\delta_f \tag{7}$$

$$-I_{xz}\ddot{\phi} + I_{zz}\ddot{\psi} = N_\beta \beta + N_{\dot{\psi}}\dot{\psi} + N_{\delta_f}\delta_f + \frac{l_w}{2}\Delta F_b \tag{8}$$

$$\left(I_{xx} + m_s h^2\right)\ddot{\phi} - I_{xz}\ddot{\psi} = m_s g h\phi + m_s v h(\dot{\beta} + \dot{\psi}) - k_f(\phi - \phi_{t,f}) - b_f(\dot{\phi} - \dot{\phi_{t,f}}) - k_r(\phi - \phi_{t,r}) - b_r(\dot{\phi} - \dot{\phi_{t,r}}) \tag{9}$$

$$-r\left(Y_{\beta,f}\beta + Y_{\dot{\psi},f}\dot{\psi} + Y_{\delta_f}\delta_f\right) = m_{u,f}v(r - h_{u,f})(\dot{\beta} + \dot{\phi}) + m_{u,f}gh_{u,f}\phi_{t,f} - k_{t,f}\phi_{t,f} + k_f(\phi - \phi_{t,f}) + b_f(\dot{\phi} - \dot{\phi_{t,f}}) \tag{10}$$

$$-r\left(Y_{\beta,r}\beta + Y_{\dot{\psi},r}\dot{\psi}\right) = m_{u,r}v(r - h_{u,r})(\dot{\beta} + \dot{\psi}) - m_{u,r}gh_{u,r}\phi_{t,r} - k_{t,r}\phi_{t,r} + k_r(\phi - \phi_{t,r}) + b_r(\dot{\phi} - \dot{\phi}_{t,r}) \tag{11}$$

---

aerodynamic inputs (wind disturbance) and road disturbances are also ignored. The roll motion of the sprung mass is damped by suspensions and stabilizers with the effective roll damping coefficients $b_{s,i}$ and roll stiffness $k_{s,i}$.

In the vehicle modeling the the lateral dynamics, the yaw moment, the roll moment of the sprung and the unsprung masses are taken into consideration. The symbols of the yaw-roll model are found in Table 1. The motion differential equations are the following.

Here, the tire coefficients are given by:

$$Y_\beta = -(C_f + C_r)\mu, \quad N_\beta = (C_r l_r - C_f l_f)\mu, \tag{12}$$

$$Y_{\dot{\psi}} = (C_r l_r - C_f l_f)\frac{\mu}{v}, \quad N_{\dot{\psi}} = -(C_f l_f^2 + C_r l_r^2)\frac{\mu}{v}, \tag{13}$$

$$Y_{\delta_f} = C_f\mu, \quad N_{\delta_f} = C_f l_f \mu. \tag{14}$$

These equations can be expressed in a state space representation. Let the state vector be the following:

$$\mathbf{x} = \begin{bmatrix}\beta & \dot{\psi} & \phi & \dot{\phi} & \phi_{t,f} & \phi_{t,r}\end{bmatrix}^T . \tag{15}$$

The system states are the side slip angle of the sprung mass β, the yaw rate $\dot{\psi}$, the roll angle ϕ, the roll rate $\dot{\phi}$, the roll angle of the unsprung mass at the front axle $\phi_{t,f}$ and at the rear axle $\phi_{t,r}$ respectively. Then the state equation arises in the following form

$$\mathbf{E}(\mathbf{p})\dot{\mathbf{x}} = \mathbf{A}_0(\mathbf{p})\mathbf{x} + \mathbf{B}_{1,0}\delta_f + \mathbf{B}_{2,0}u, \tag{16}$$

where the matrices are defined by equations (21) and (22). The parameter of the system is the forward velocity

$$\mathbf{p} = v.$$

Equation (16) can be rewritten as

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{p})\mathbf{x} + \mathbf{B}_1(\mathbf{p})\delta_f + \mathbf{B}_2(\mathbf{p})u, \tag{17}$$

$$\mathbf{E}(v) = \begin{bmatrix} mv & 0 & 0 & -m_s h & 0 & 0 \\ 0 & I_{zz} & 0 & -I_{xz} & 0 & 0 \\ -m_s vh & -I_{xz} & 0 & I_{xx} + m_s h^2 & -b_f & -b_r \\ m_{u,f} v(r - h_{u,f}) & 0 & 0 & 0 & -b_f & 0 \\ m_{u,r} v(r - h_{u,r}) & 0 & 0 & 0 & 0 & -b_r \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B}_{1,0} = \begin{bmatrix} Y_{\delta_f} \\ N_{\delta_f} \\ 0 \\ rY_{\delta_f} \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{B}_{2,0} = \begin{bmatrix} 0 \\ l_w/2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{21}$$

$$\mathbf{A}_0(v) = \tag{22}$$

$$\begin{bmatrix} Y_\beta & Y_{\dot\psi} - mv & 0 & 0 & 0 & 0 \\ N_\beta & N_{\dot\psi} & 0 & 0 & 0 & 0 \\ 0 & m_s hv & m_s gh - k_f - k_r & -b_f - b_r & k_f & k_r \\ -rY_{\beta,f} & rY_{\dot\psi,f} - m_{u,f} v(r - h_{u,f}) & -k_f & -b_f & k_f + k_{t,f} - m_{u,f} g h_{u,f} & 0 \\ -rY_{\beta,r} & -rY_{\dot\psi,r} - m_{u,r} v(r - h_{u,r}) & -k_r & -b_r & 0 & k_r + k_{t,r} - m_{u,r} g h_{u,r} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

where

$$\mathbf{A}(\mathbf{p}) = \mathbf{E}^{-1}(\mathbf{p})\mathbf{A}_0(\mathbf{p}) \tag{18}$$

$$\mathbf{B}_1(\mathbf{p}) = \mathbf{E}^{-1}(\mathbf{p})\mathbf{B}_{1,0} \tag{19}$$

$$\mathbf{B}_2(\mathbf{p}) = \mathbf{E}^{-1}(\mathbf{p})\mathbf{B}_{2,0} \tag{20}$$

The δ_f is the front wheel steering angle. The control input is the difference of brake forces between the left and the right hand side of the vehicle.

$$u = \Delta F_b \tag{23}$$

The control input provided by the brake system generates a yaw moment, which affects the lateral tire forces directly. In our case it is assumed that the brake force difference ΔF_b provided by the controller is applied to the rear axle. This means that only one wheel is decelerated at the rear axle. This declaration is caused by an appropriate yaw moment. In our case the difference between the brake forces can be given $\Delta F_b = F_{b,rl} - F_{b,rr}$. This assumption does not restrict the implementation of the controller because it is possible that the control action be distributed on the front and the rear wheels at one of the two sides. The reason for distributing the control force to front and rear wheels is to minimize the wear of the tires. In this case a logic is required which calculates the brake forces for the wheels.

In the equation (17) the **A(p)** matrix depends on the forward velocity of the vehicle nonlinearly. In the linear yaw-roll model the velocity is considered a constant parameter. However, forward velocity is an important stability parameter so that it is considered to be a variable of the motion. Hence the throttle is constant during a lateral maneuver and the

Table 1: Symbols of the yaw-roll model

Symbols	Description
h	height of CG of sprung mass from roll axis
$h_{u,i}$	height of CG of unsprung mass from ground
r	height of roll axis from ground
a_y	lateral acceleration
β	side-slip angle at center of mass
ψ	heading angle
$\dot{\psi}$	yaw rate
ϕ	sprung mass roll angle
$\phi_{t,i}$	unsprung mass roll angle
δ_f	steering angle
u_i	control torque
C_i	tire cornering stiffness
F_{zi}	total axle load
R_i	normalized load transfer
k_i	suspension roll stiffness
b_i	suspension roll damping
$k_{t,i}$	tire roll stiffness
I_{xx}	roll moment of inertia of sprung mass
I_{xz}	yaw-roll product of inertial of sprung mass
I_{zz}	yaw moment of inertia of sprung mass
l_i	length of the axle from the CG
l_w	vehicle width
μ	road adhesion coefficient

forward velocity depends on only the brake forces. The differential equation for forward velocity is

$$m\dot{v} = -F_{b,rl} - F_{b,rr}.$$

3.1.2. TP model representation of the heavy vehicle model

In this section we derive the TP model of the LPV model (17) by TP model transformation. We execute the TP model transformation over M ($M = 137$) points grid net in the $v \in \Omega = [40\text{km/h}, 120\text{km/h}]$ domain. We have applied the MATLAB Tensor Product Model Transformation Toolbox (TPTool) (`http:\\tptool.sztaki.hu`) for the TP model transformation to determine the LTI systems ($\mathbf{S}_i$) and the weightings ($\mathbf{w}_i$). The TP model transformation shows that the LPV model of the heavy vehicle model can exactly be given by the convex combination of 3 LTI vertex systems:

$$\mathbf{S}(\mathbf{p}(t)) = \sum_{i=1}^{3} w_i(\mathbf{p}(t))\mathbf{S}_i \tag{24}$$

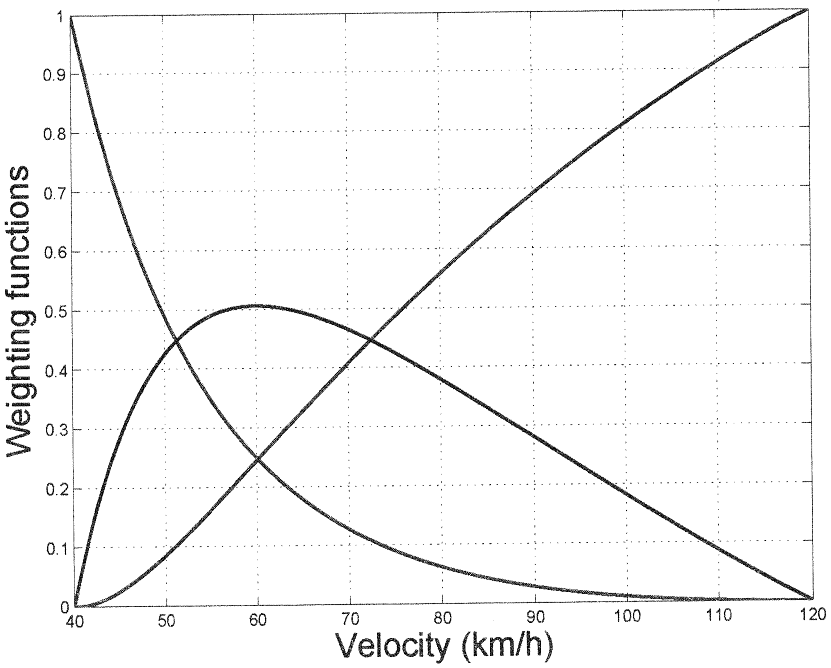


Figure 2: Close to NO weighting functions of the TP model

The type of convexity considerably influences the feasibility of LMI theorems and resulting controllers control performance. In order to relax the feasibility of the LMI conditions, we define the tight convex hull of the LPV model via generating close to NO type weighting functions by the TP model transformation, see Figure 2.

Definition 3 (NO – Normality) *Vector* $\mathbf{w}(p)$, *containing weighting functions* $w_i(p)$ *is NO if they satisfy conditions (4) and (5), and the maximum values of the weighting functions are one. We say* $w_i(p)$ *is close to NO if it satisfies conditions (4) and (5), and the maximum values of the weighting functions are close to one.*

Its geometrical meaning is that we determine a convex hull in such a way that as many of the LTI systems as possible are equal to the $\mathbf{S}(\mathbf{p})$ over some $\mathbf{p} \in \Omega$ and the rest of the LTIs are close to $\mathbf{S}(\mathbf{p})$ (in the sense of $\mathcal{L}_2$ norm).

3.2. Controller design

3.2.1. $\mathcal{H}_\infty$ controller design

The aim of the rollover prevention is to provide the vehicle with the ability to resist overturning moments generated during cornering. Roll stability is determined by the height of the center of mass, the track width and the kinematic properties of the suspensions. The problem with heavy vehicles is a relatively high mass center and narrow track width. When the vehicle is changing lanes or trying to avoid obstacles, the vehicle body rolls out of the corner and the center of mass shifts outboard of the centerline, and a destabilizing moment is created.

In this section we utilize the above obtained convex model for the stabilization control of the heavy vehicle model. We seek an LPV controller of the form

$$\begin{aligned} \dot{\mathbf{x}}_K &= \mathbf{A}_K(\mathbf{p}(t))\mathbf{x}_K + \mathbf{B}_K(\mathbf{p}(t))y \\ u &= \mathbf{C}_K(\mathbf{p}(t))\mathbf{x}_K + \mathbf{D}_K(\mathbf{p}(t))y, \end{aligned}$$

where

$$\begin{pmatrix} \mathbf{A}_K(\mathbf{p}(t)) & \mathbf{B}_K(\mathbf{p}(t)) \\ \mathbf{C}_K(\mathbf{p}(t)) & \mathbf{D}_K(\mathbf{p}(t)) \end{pmatrix} = \mathbf{K}(\mathbf{p}(t)) = \sum_{i=1}^{3} w_i(\mathbf{p}(t))\mathbf{K}_i \tag{25}$$

with the same $w_i(\mathbf{p}(t))$ weighting functions as in the model representation (24), $\mathbf{x}_K$ is the internal state of the controller, the measured output of the model is the yaw rate and lateral acceleration so

$$y = \begin{bmatrix} \dot{\psi} & a_y \end{bmatrix}, \tag{26}$$

where the lateral acceleration can be calculated as

$$a_y = v(\dot{\beta} + \dot{\phi}) - \frac{m_s}{m} h\ddot{\phi} \tag{27}$$

and the control signal is given by (23). To design a suitable $\mathbf{K}(\mathbf{p})$ for the given polytopic model the self-scheduled $\mathcal{H}_\infty$ controller design method [9, 10] was used.

The closed-loop interconnection structure, which includes the feedback structure of the model $\mathbf{P}$ and controller $\mathbf{K}$, is shown in Figure 3. In the diagram, $\mathbf{d}$, $\mathbf{u}$, $\mathbf{y}$ and $\mathbf{z}$ are the disturbance, the control input, the measured output and the performance output, respectively.

A standard feedback configuration with weights strategy is illustrated in Figure 4. In the diagram $\mathbf{u}$ is the control input, $\mathbf{y}$ is the measured output, $\mathbf{z}_p$ is the performance output, $\mathbf{z}_u$ and $\mathbf{z}_y$ are performances at the input and the output, $\mathbf{w}$ is the disturbance, $\mathbf{n}$ is the

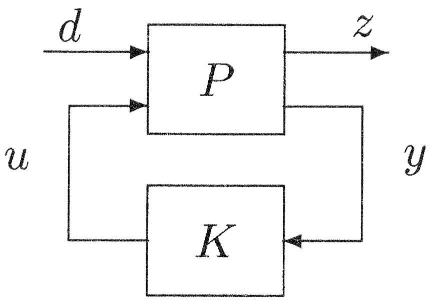


Figure 3: The general $P - K$ structure for control design

measurement noise. The aim of the weighting function $\mathbf{W}_p$ is to define the performance specifications. They can be considered as penalty functions, i.e. weights should be large in a frequency range where small signals are desired and small where large performance outputs can be tolerated. $\mathbf{W}_u$ and $\mathbf{W}_y$ may be used to reflect some restrictions on the actuator and on the output signals. The purpose of the weighting functions $\mathbf{W}_w$ and $\mathbf{W}_n$ is to reflect the disturbance and sensor noises. The disturbance and the performances in the general $\mathbf{P} - \mathbf{K}$ structure are $\mathbf{d} = \begin{bmatrix}\mathbf{w} & \mathbf{n}\end{bmatrix}^T$ and $\mathbf{z} = \begin{bmatrix}\mathbf{z}_u & \mathbf{z}_y & \mathbf{z}_p\end{bmatrix}^T$.

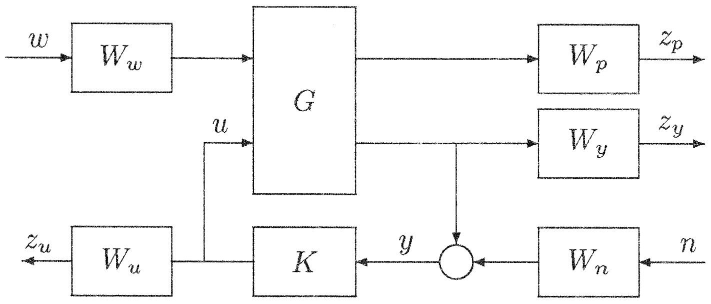


Figure 4: The standard feedback configuration with weights

The augmented plant includes the parameter dependent vehicle dynamics and the weighting functions, which are defined in the following form:

$$\begin{bmatrix}\mathbf{z}\\ \mathbf{y}\end{bmatrix} = \mathbf{P}(\mathbf{p})\begin{bmatrix}\mathbf{d}\\ \mathbf{u}\end{bmatrix}. \tag{28}$$

In a Linear Parameter Varying (LPV) model **p** denotes the scheduling variable.

The closed-loop system **M(p)** is given by a lower linear fractional transformation (LFT) structure:

$$\mathbf{M(p)} = \mathcal{F}_\ell(\mathbf{P(p)},\mathbf{K(p)}), \tag{29}$$

where **K(p)** also depends on the scheduling variable **p**. The goal of the control design is to minimize the induced $\mathcal{L}_2$ norm of an *LPV* system **M(p)**, with zero initial conditions, which is given by

$$\|\mathbf{M(p)}\|_\infty = \sup_{\mathbf{p}\in\Omega} \sup_{\|\mathbf{w}\|_2\neq 0, \mathbf{w}\in\mathcal{L}_2} \frac{\|\mathbf{z}\|_2}{\|\mathbf{w}\|_2}. \tag{30}$$

4. Simulation

4.1. Simulation setup

At the initial configuration of the simulation the system had a velocity of v = 100km/h and all the state variables were set to zero. Then a sharp maneuver was simulated as seen in Figure 5, which describes the situation when the truck performs obstacle avoidance. The goal is to stabilize the truck by braking the rear wheels.

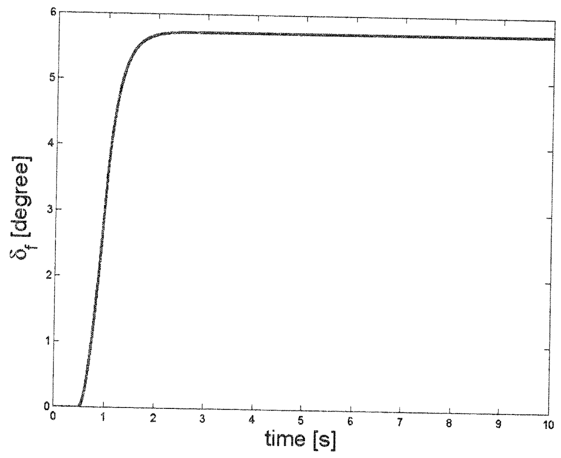
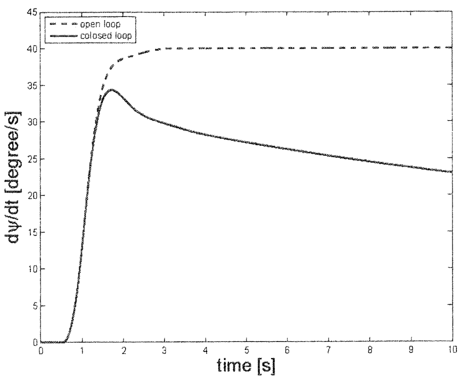
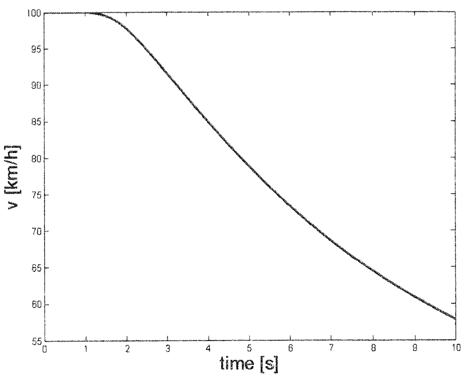


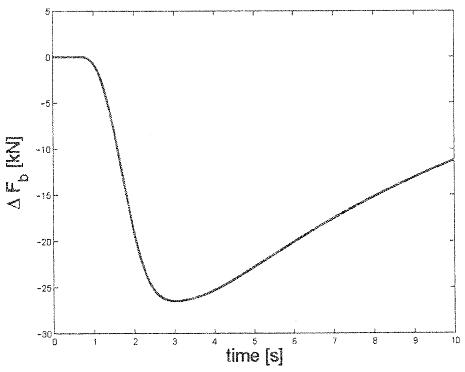
Figure 5: Disturbance signal: Steering angle



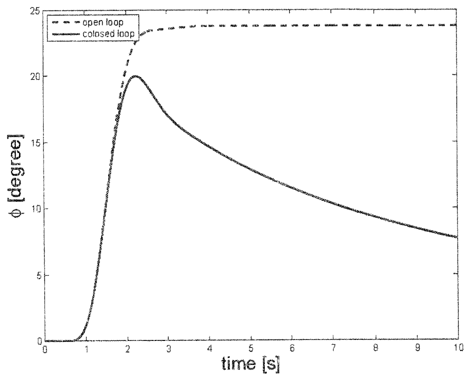
(a) Measured signal: yaw rate



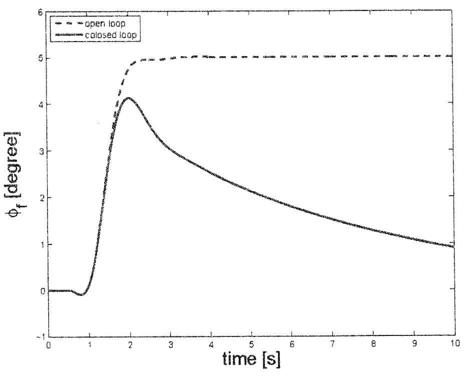
(b) Forward velocity



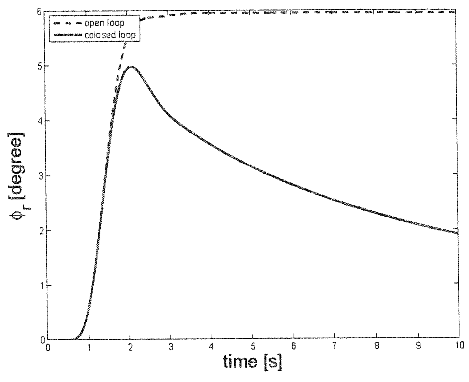
(c) Control signal: difference of left and right brake forces



(d) Sprung mass roll angle



(e) Front unsprung mass roll angle



(f) Rear unsprung mass roll angle

Figure 6: Simulation results

4.2. Simulation results

The results can be seen on Figure 6. From the results it can be seen that the rollover likelihood can be reduced efficiently with active breaking.

From the results it can be seen that the rollover likelihood can be reduced efficiently with active breaking. The controller properties such as the magnitude of the control signal or the maximum allowed roll angle can be easily tuned by the $\mathcal{H}_\infty$ weightings.

5. Conclusion

In this paper we investigated the rollover prevention problem for the heavy vehicle model. The novelty of this paper is that the convex polytopic representation of the vehicle model was generated by Tensor Product Model Transformation. Current approaches usually need sophisticated analytical methods to prepare the LPV system model for LMI design and neglect the importance of the convex representation. With the calculated model an LMI based $\mathcal{H}_\infty$ gainÂscheduling method was used to determine the controller to solve the rollover problem. Simulations present the performance of this controller design method.

Acknowledgement

This research was supported by the Hungarian National Found (OTKA) under the grant No. F 049838, Dr. Baranyi is supported by János Bolyai Postdoctoral Scholarship of the Hungarian Academy of Sciences.

References

[1] Gaspar,P., Szaszi,I., Bokor,J. *The design of a combined control structure to prevent the rollover of heavy vehicles*, European journal of control (2004) vol. 10, no. 2, pp. 148–162

[2] ——, *Design of robust controllers for active vehicle suspensions*, IFAC World Congress, Barcelona (2002) pp. 1473–1478

[3] Palkovics,L., Semsey,A., Gerum,E. *Roll-over prevention system for commercial vehicles - additional sensorless function of the electronic brake system*, Vehicle System Dynamics (1999) vol. 32, pp. 285–297

[4] Scherer, C.W. *LPV control and full block multipliers*, Automatica (2001) vol. 37, pp. 361–375

[5] Boyd,S., Ghaoui,L.E., Feron,E., Balakrishnan,V. *Linear Matrix Inequalities in Systems and Control Theory*, Philadelphia: SIAM books (1994)

[6] Lathauwer,L.D., Moor,B.D., Vandewalle,J. *A multilinear singular value decomposition*, SIAM Journal on Matrix Analysis and Applications, (2001) vol. 21, no. 4, pp. 1253–1278

[7] Baranyi,P. *Tensor-product model-based control of two-dimensional aeroelastic system*, Journal of Guidance, Control, and Dynamics (2005) vol. 29, no. 2, pp. 391–400

[8] Baranyi,P., Tikk,D., Yam,Y., Patton,R.J. *From differential equations to PDC controller design via numerical transformation*, Computers in Industry, Elsevier Science (2003) vol. 51, pp. 281–297

[9] Apkarian,P., Gahinet,P., Becker,G. *Self-scheduled $\mathcal{H}_\infty$ control of linear parameter-varying systems*, Proc. Amer. Contr. Conf. (1994) pp. 856–860

[10] Apkarian,P., Gahinet,P. *A convex characterization of gain-scheduled $\mathcal{H}_\infty$ controllers*, IEEE Trans. Aut. Contr. (1995)

Inference in Fuzzy Signature Based Models

Katalin Tamás[1], László T. Kóczy[1,2]

[1]Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics H-1117, Budapest, Magyar tudósok krt. 2., Hungary e-mail: tamas.kati@gmail.com, koczy@tmit.bme.hu

[2] Faculty of Engineering Sciences, Széchenyi István University, H-9026, Győr, Egyetem tér 1., Hungary e-mail: koczy@sze.hu

Abstract: The concept of fuzzy signatures was introduced to help model the many complex and well structured problems, where a hierarchical structure within the observed data is present. This means that one or several components of the structure can be determined at a higher level by a sub-tree of other components. In this way, the data components can be represented in tree form. By examining the problem, an arbitrary structure belonging to the data set can be determined. Due to the fact that some components might be missing from a data element, the actual tree structures of the data may slightly differ. So that these data can be evaluated and compared, aggregation operators are given for each node in the arbitrary structure for the purpose of modifying the structure. To model problems with this type of dataset, fuzzy signature based rules can be constructed. Inferring a conclusion from such a model is a key issue. In this paper fuzzy signature based rule bases will be introduced, then the generalization of the widely used Mamdani-type fuzzy inference system for fuzzy signature based rules will be presented step-by-step. Furthermore, a working software implementation of the generalized Mamdani-type inference systems will be presented. The software will be demonstrated through a possible application of the system on a realistic example. First, the problem's fuzzy signature model will be constructed, and then the process of inference for an observation with some missing data components will be shown.

Keywords: fuzzy signatures, Mamdani-type inference

1. Fuzzy Signatures

1.1. Introduction

In 1967 Goguen introduced L-fuzzy sets [2] as the generalization of the original concept of fuzzy sets, which were introduced by Zadeh [12] in 1965. L-fuzzy membership grades are elements of an arbitrary lattice L:

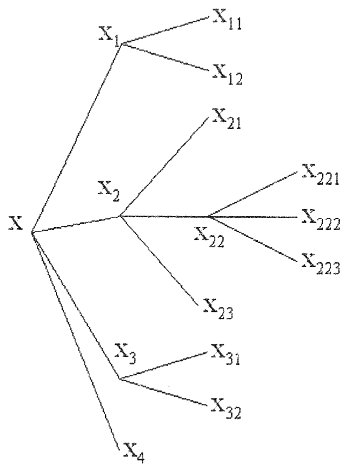
$$A_L : x \to L, \quad \forall x \in X. \tag{1}$$

In 1980 vector valued fuzzy sets were introduced [3], which are special L-fuzzy sets, where in this case L is the lattice of n-dimensional fuzzy vectors, $L = [0,1]^n$ in (1). This means that instead of assigning a single membership grade to each element of X, as is done when defining original fuzzy sets, a set of quantitative features are assigned to each element of X in vector valued fuzzy sets, this way providing additional information about that specific element of the domain.

Fuzzy signatures were introduced in 1999 [4], as a generalized form of vector valued fuzzy sets, where each component of a vector assigned to an element of X is possibly another nested vector. This generalization can be continued to any finite depth, forming a signature with depth m.

$$A_S : x \to [a_i]_{i=1}^k, a_i = \begin{cases} [0,1] \\ [a_{ij}]_{j=1}^{k_i} \end{cases}, a_{ij} = \begin{cases} [0,1] \\ [a_{ijl}]_{l=1}^{k_{ij}} \end{cases}, \quad \forall x \in X. \tag{2}$$

The structure of fuzzy signatures can be represented in vector form (as in the definition (2)) and also in a tree structure (each nested vector in the definition is represented by a sub-tree). An example of these structures can be seen in Figure 1.



$$x = \begin{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix} \\ \begin{bmatrix} x_{21} \\ \begin{bmatrix} x_{221} \\ x_{222} \\ x_{223} \end{bmatrix} \\ x_{23} \end{bmatrix} \\ \begin{bmatrix} x_{31} \\ x_{32} \end{bmatrix} \\ x_4 \end{bmatrix}$$

Figure 1. The tree structure and the vector form of an example fuzzy signature

Fuzzy signatures can be considered as special, multidimensional fuzzy data, where some of the components are interrelated in the sense that a sub-group of variables determines a feature on a higher level. This way the additional information contained within the complex and interdependent data components can be stored in the structure. The comparison (e.g. calculation of the degree of matching) of such structured data can be carried out more effectively when the data's structure is also taken into account.

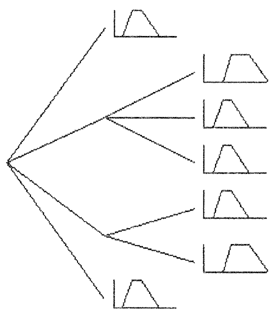
Often the information available to experts can de depicted by slightly different structures, furthermore the structure of the actual observation can also differ. However, the experts of the certain field have to make their decisions based on the available information. These dissimilar structures can be dealt with when using fuzzy signatures. The great advantage of fuzzy signatures lies exactly in this property that they can deal with differently structured signatures.

In addition, by using signatures, the problem's model can be organized into a hierarchic system [11], which is very similar to the way human experts think. Hereby fuzzy signatures could be used to model such areas like decision support systems in the medical field, where the physicians themselves take into account many data components, with possibly different signature structures to make their decisions.

It is often the case that no information is available about some elements of the model. In the conventional data mining processes, during the preparation phase, data with missing entries are eliminated, because such data cannot be handled by the model builder. However when modeling with fuzzy signatures, these data do not have to be eliminated from the dataset. The importance of fuzzy signatures is exactly this: that they can cope with cases when some elements of the original structure are not present.

1.2. Fuzzy Signature Sets

The basic structure of fuzzy signature sets is similar to that of fuzzy signatures, the only difference being that instead of having fuzzy variables on the leaves of the structure, membership functions are present (see Figure 2). The only constraint for the membership functions is that their domain must be the [0,1] interval.



$$S = \begin{bmatrix} \mu_1(x) \\ \begin{bmatrix} \mu_{21}(x) \\ \mu_{22}(x) \\ \mu_{23}(x) \end{bmatrix} \\ \begin{bmatrix} \mu_{31}(x) \\ \mu_{32}(x) \end{bmatrix} \\ \mu_4(x) \end{bmatrix}$$

Figure 2. The tree structure and the vector form of an example fuzzy signature set

1.3. Structure Modification: Aggregation Operators

The advantages of fuzzy signatures lie in organizing the available data components into a hierarchy. This hierarchy determines the arbitrary structure of our fuzzy signature observations. As some of the components of this arbitrary structure might be missing from the specific observations, some kind of structure modifying operation is essential when comparing these differently structured signatures.

Aggregation operations result in a single fuzzy value calculated from a set of other fuzzy values, while satisfying a set of axioms. The most common operators are the maximum, minimum and arithmetic mean operator.

Aggregation operators can be used to transform fuzzy signature structures by reducing a sub-tree of variables to their parent node. It is necessary to mention that only whole sub-trees of the structure can be reduced. The fuzzy value (or fuzzy set) assigned to the parent node is calculated by aggregating the fuzzy values (or fuzzy sets) of its children using the aggregation operator of the parent node. This way, the depth of this branch of the structure is reduced by one.

This procedure can only be performed when all elements of the aggregated sub-tree are leaves of the structure and have an assigned value. If one of the elements of the sub-tree branches out into a sub-tree of its own, in order to reduce the whole sub-tree to the original parent node, first the sub-sub-tree has to be reduced to its parent node (which is the child node of the original sub-tree's parent node) by aggregation. After performing the aggregation, the original sub-tree can also be reduced.

For example, to reduce the sub-tree of node x_i, first the sub-tree of node x_{i2} has to be aggregated. The value obtained can then be used when calculating the aggregate value from the sub-tree of x_i. This recursive procedure is shown in Figure 3, where $@_i$ denotes the aggregation operator of node x_i.

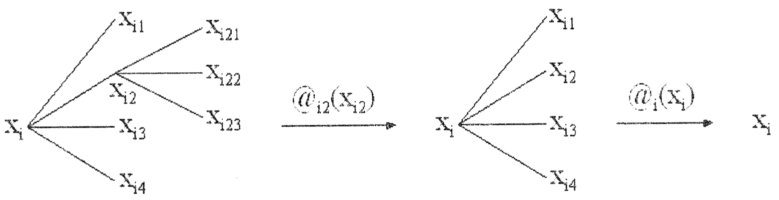


Figure 3. Recursion used to reduce a sub-tree with several layers

Because of these terms, when reducing a signature to a predefined structure it is wise to use a bottom-up method. This means to start the reduction from the leaves of the structure and work your way up one sub-tree at a time towards the intended structure.

According to the definition of fuzzy signatures, aggregation operators define the connection between a component, and its sub-components, therefore the aggregation operators are not necessarily identical for all the nodes of the structure. Finding the relevant aggregation operator for each node is a very important problem of fuzzy signatures, because when comparing two signatures, the obtained results may greatly depend on the aggregation operators used to reduce the signatures to a common structure.

It is also important to mention that when reducing a signature's sub-tree, some information is lost in all cases, because the calculated aggregated value can be the same for many different values and differently structured sub-trees.

1.3.1. Aggregation on Fuzzy Sets

In order to reduce fuzzy signature sets to a different structure aggregation has to be generalized to work not only on fuzzy values but on fuzzy sets as well.

When aggregating fuzzy sets, the membership values for each element x of [0,1] (the domain of the fuzzy sets on the leaves of the structure) are calculated for all the fuzzy sets which are subject to the aggregation. The original aggregation operator is then used on these membership values to obtain the aggregated membership value belonging to x. Let the fuzzy sets in the sub-tree be A_i. The membership function of the aggregated fuzzy set G is given in (3), where h denotes the aggregation operator.

$$\begin{aligned} &G = h(A_1, A_2, \ldots, A_k) \\ &\forall x \in [0,1] \quad \mu_G(x) = h\{\mu_{A_1}(x), \mu_{A_2}(x), \ldots, \mu_{A_k}(x)\} \end{aligned} \tag{3}$$

The aggregation of two fuzzy sets (A_1 and A_2) is shown in Figure 4. In the example, the aggregation operator is the arithmetic mean operator. The resulting fuzzy set (denoted by G) is marked with a broken line.

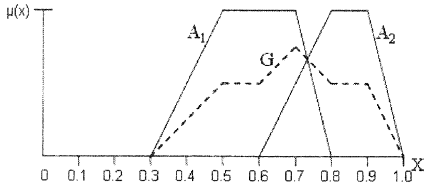


Figure 4. Aggregation of two fuzzy sets with the arithmetic mean operator

1.3.2. Weighted Relevance Aggregation Operator

With the introduction of weights [6] for each node of the fuzzy signature structure additional expert knowledge about the field can be contained within the model. The relevance weight depicts how relevant a node is in its parent's sub-tree. The weights of the nodes are taken from the [0;1] interval, and it is not necessary for the weights of the leaves in a sub-tree to add up to 1. A method for learning weights was shown in [7].

The most general form of aggregation operators is the Weighted Relevance Aggregation Operator (WRAO) introduced by Mendis *et al.* in [8]. The values and weights belonging to each child l in the sub-tree are denoted by x_l and w_l respectively. The definition of the WRAO is as follows:

$$@(x_1, x_2, \ldots, x_n; w_1, w_2, \ldots, w_n) = \left\{\frac{1}{n}\sum_{i=1}^{n}(w_l \cdot x_l)^p\right\}^{\frac{1}{p}} \tag{4}$$

where p is the aggregation factor of the above function. $(p \in \Re,\ p \neq 0)$

The well-known aggregation operators are all special cases of WRAO depending on the value of p in (4).

$$\begin{aligned}
&p \to -\infty, && \text{WRAO} \to \text{minimum}\\
&p = -1, && \text{WRAO} = \text{harmonic mean}\\
&p \to 0, && \text{WRAO} \to \text{geometric mean}\\
&p = 1, && \text{WRAO} = \text{arithmetic mean}\\
&p \to \infty, && \text{WRAO} \to \text{maximum}
\end{aligned}$$

2. Fuzzy Inference Systems

2.1. Fuzzy Rules and Rule Bases

Fuzzy rules are formulated as If... then... rules, where the If... part is called the rule antecedent and the then... part is the rule consequent. In the antecedent part of the rules there can be numerous input variables (x_i), while in the consequent part there is generally only one output variable. The antecedent part of the rules is formulated by either linguistic terms or convex and normal fuzzy sets (one for each input variable); the consequent is also defined with a linguistic term or a membership function.

The general form of a multiple input, single output fuzzy rule is the following:

$$R : \text{If } \mathbf{x} = \mathbf{A} \text{ then } y = B, \tag{5}$$

where $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ the set of input variables, whose domain is $X = X_1 \times \cdots \times X_n$, $x_j \in X_j$. $\mathbf{A} = \langle A_1, \ldots, A_n \rangle$ is the vector of the antecedent fuzzy sets for each variable, $\mathbf{A} \in X$. The output variable of the rule is y over the domain Y. The consequent of the rule is fuzzy set B, where $B \in Y$.

For example in a fuzzy system developed for use in a fuzzy air-conditioning system, a fuzzy rule could be the following: If 'air temperature' is warm and 'air humidity' is high then 'air-conditioning' is little. Of course the linguistic variables used in this example first have to be defined.

To describe a system, experienced human operators in the given field may set up many linguistic control rules such as the one shown above. A rule describes the expected behavior of the system for certain groups of inputs. The defined rules belonging to a system are grouped to form a fuzzy rule base from which conclusions can be inferred.

2.2. Mamdani-type Inference Systems

Fuzzy rule based models were first proposed by Zadeh in 1973 [13] and were later implemented practically with some technical innovations achieving the reduction of complexity by Mamdani and Assilian in 1974 [5]. This so-called Mamdani-type inference system is the most widely used inference system today. The knowledge gathered in such a system is stored in a fuzzy rule base (as described in the previous section).

Fuzzy inference systems calculate a crisp output from a set of input fuzzy variables using the model of the system. If the inputs are not fuzzy variables, the input values (x_i) first have to be fuzzified over the universe of the specific input variable (X_i).

Fuzzy inference systems are usually made up of the following four main components (see Figure 5): a fuzzy rule base, a component calculating the degree of matching, a fuzzy inference engine and a defuzzification component.

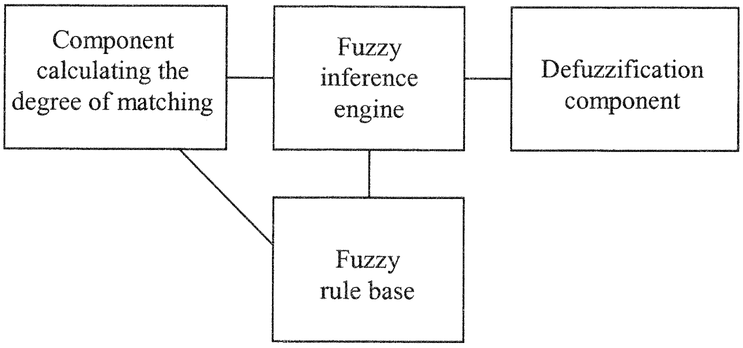

Component calculating the degree of matching
Fuzzy inference engine
Defuzzification component
Fuzzy rule base

Figure 5. Fuzzy inference system block diagram

The component calculating the degree of matching computes the degree of matching between the observation (vector of the input variables) and the antecedent of a rule. The fuzzy inference system determines a consequent fuzzy set using these values. To obtain a crisp output the consequent fuzzy set is defuzzified using the defuzzification component.

2.2.1. Calculating the Degree of Matching

As the first step of the inference, the degree of matching of the input and each rule has to be computed. For this each component of the input vector has to be matched with the corresponding component of the antecedent vector of each rule. Let the n-dimensional input vector be **A'**. The degree of matching in the j^{th} dimension between the i^{th} rule and the input is marked by $w_{j;i}$ in (5). In the equation A'_j marks the j^{th} dimensional input fuzzy set and $A_{j,i}$ marks the antecedent fuzzy set of the i^{th} rule in the j^{th} dimension.

$$w_{j,i} = \max_{x_j}\left\{ \min\left\{ A'_j(x_j), A_{j,i}(x_j) \right\} \right\} \tag{5}$$

Figure 6 also illustrates the calculation of the degree of matching between two fuzzy sets (shown in (5)).

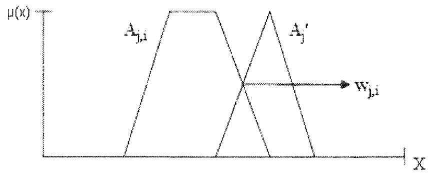


Figure 6. Determining the degree of matching of two fuzzy sets

When the system's input is a vector of crisp values (**x'**), then Equation 5 alters in the following way:

$$w_{j,i} = A_{j,i}(x'_j), \tag{5'}$$

where x'_j is the j^{th} dimension of the crisp input vector.

After determining the degree of matching ($w_{j,i}$) in each dimension, the degree of matching also has to be calculated for the whole antecedent, because the degrees of matching in each dimension of the rule affect the overall match of rule R_i along with the output. The overall degree of matching between rule R_i and the input is obtained by taking the minimum of the values calculated for each dimension, as shown in (6).

$$w_i = \min_{j=1}^{n} w_{j,i} \tag{6}$$

The degree of matching w_i gives the extent of the influence that rule R_i has on the conclusion drawn for the given input.

The above process is repeated for all rules R_i ($i=1,...,r$) in the rule base, so that all the degrees of matching ($w_1,...,w_r$) are produced.

2.2.2. Mamdani-type Inference Engine

The inference engine uses the previously calculated degrees of matching to infer an overall conclusion for the given input.

First the conclusion fuzzy set (denoted as B'_i) belonging to each rule R_i has to be determined, by taking the t-norm of the original output fuzzy set of the rule (B_i) and the degree of matching between the rule antecedent and the input (w_i). The t-norm used in Mamdani-type inference systems is the minimum operator, as shown in (7).

$$B'_i(y) = \min(w_i, B_i(y)) \tag{7}$$

Figure 7 shows the whole inference process (calculating w_i and B'_i as well) on rule R_i with a crisp input.

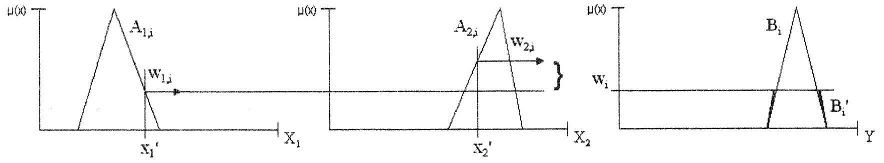


Figure 7. The conclusion of a rule

Next the final conclusion fuzzy set belonging to the whole rule base is generated by taking the s-norm of all the conclusion fuzzy sets calculated for each rule in (7). The s-norm used in the Mamdani-type inference systems is the maximum operator, as shown in (8).

$$B'(y) = \max_{i=1}^{r} B'_i(y) \tag{8}$$

A complete inference process with a crisp two dimensional input (**x'** = ‹*x'*$_1$, *x'*$_2$›) and two rules in the rule base (r = 2) is shown in Figure 8.

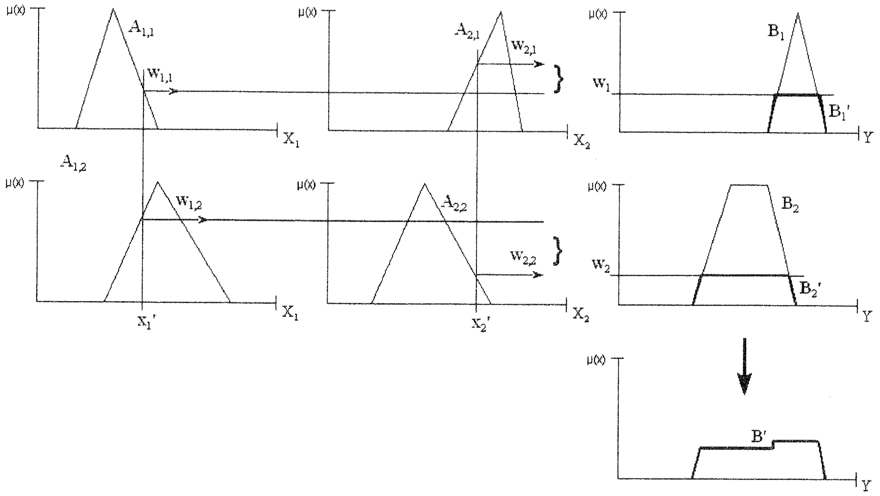


Figure 8. A complete inference process

2.2.3. Defuzzification methods

At the end of the inference process a conclusion fuzzy set (*B'*(y)) is obtained, although in most cases a crisp value is expected as the output of the fuzzy system. This means that the crisp value which best characterizes the conclusion has to be determined. This process is called defuzzification.

There are many different methods of which the Center of Area (COA) is the most widely used.

The Center of Area of a fuzzy set can be calculated using the formula in (9).

$$y_{COA} = \frac{\int_{y \in B'} B'(y) y \, dy}{\int_{y \in B'} B'(y) \, dy} \tag{9}$$

The disadvantage of this method is that the integrals are hard to compute when dealing with complicated fuzzy sets.

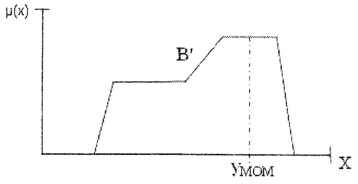


Figure 9. Defuzzification with the Middle of Maximum method

The Middle of Maximum (MOM) method defines the defuzzified value of the fuzzy set as the average of the base values whose membership values are maximal. This method is best illustrated by Figure 9.

3. Fuzzy Signatures in Rule Bases

The general form of a fuzzy rule is the following:

If x is A_i then y is B_i,

where A_i is the rule antecedent, B_i is the rule consequent, x is the observation and y is the conclusion. This rule can be extended to function on fuzzy signatures too, as it was shown in [9].

The rule antecedent, A_i can either be a fuzzy signature set or simply a fuzzy signature singleton, keeping in mind that for all the rules in the fuzzy signature based rule base all the signatures have the same arbitrary structure and the corresponding aggregation operators are uniform for every rule.

The consequent parts of the rules remain fuzzy sets.

The observation A' is either a fuzzy signature singleton or a fuzzy signature set. The structure of the observation can be obtained from the arbitrary structure used in the rule antecedents by removing some leaves or even whole sub-trees. This indicates that some information might not be available in our observation.

Signature A' is obtained from the original fuzzy singleton or fuzzy set observations by normalizing the domains on each leaf of the signature.

4. Mamdani Inference in Fuzzy Signature Based Models

To infer a conclusion from a fuzzy signature based rule base for an observation given in fuzzy signature form, a modified version of the original inference algorithm introduced by Mamdani [5] was given in [9].

The modified algorithm consists of three main steps. First the degree of matching between the observation and each rule in the rule base is calculated. In the second step a fuzzy set is inferred from the consequents of the rules based on the previously calculated degrees of matching, like in the original algorithm. In the third step the conclusion is obtained by applying a defuzzification method known from literature on the previously inferred fuzzy set.

Compared to the original Mamdani method, only the first step, where the degree of matching between a fuzzy signature observation (A') and the fuzzy signature rule antecedents (A_i) contains a novel approach, so this step will be discussed in detail.

4.1. Calculating the Degree of Matching

This step can be further divided into three sub steps, which are the following: finding the common structure, constructing the signature representing the degree of matching and then calculating the degree of matching.

4.1.1. Finding the Common Structure

In order to compare two signatures which have the same arbitrary structure, but whose structures differ slightly, first the common structure of these signatures has to be found, then both signatures have to be reduced to this new structure using aggregation (see Figure 10).

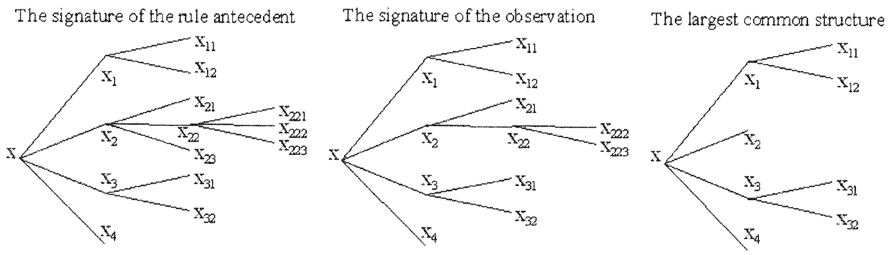

The signature of the rule antecedent
The signature of the observation
The largest common structure
x
x1
x11
x12
x2
x21
x22
x221
x222
x223
x23
x3
x31
x32
x4

Figure 10. The largest common structure of two signatures

The common structure of two fuzzy signatures is defined as the maximal common sub-tree of the structures to which both structures can be reduced using aggregation [10]. This structure can be defined by performing width-first search (WFS) simultaneously on both tree structures. In the algorithm the common structure of two signatures (denoted by S1 and S2 respectively) are obtained.

At the start of the algorithm the root node is added to the search queue, which is denoted by Q. The first step consists of taking the first node in the search queue and naming it *act* (short for actual node). In the second step, the lists of indexes of the children of the actual node in both signatures (S1 andS2) are queried from the respective fuzzy signatures. All nodes of a signature are denoted by an index, which is obtained by concatenating the ordinal number of a child node to its parent's index. The index of the root node is ''.

The lists of children's indexes are denoted by c1 and c2 respectively. At this point two cases are possible: either the number of children in the two lists is equal, or it is not. In the first case, if the indexes contained in the lists are also the same, the actual node is added to the common structure and the children in the lists are added to the search queue of the WFS, because they also have to be included in the search. However, if the indexes in the lists are not the same or the number of children of the actual node differs in the two signatures, then the actual node is added to the common structure, but no new nodes are added to the search queue. This means that when reducing the signatures to their maximal common structure using the aggregation operators, the value of the actual node will have to be aggregated from the values contained in its sub-tree in at least one of the signatures.

While there are still nodes in the search queue, the WFS continues from the first step described above, else the algorithm exits. The largest common structure of the two fuzzy signatures is now defined. The flowchart of the algorithm is shown in Figure 11.

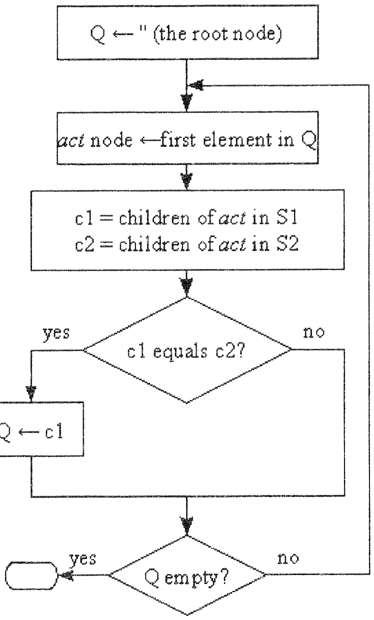

Q ← '' (the root node)
act node ←first element in Q
c1 = children of act in S1
c2 = children of act in S2
c1 equals c2?
yes
no
Q ← c1
Q empty?
yes
no

Figure 11. Flowchart of finding the largest common structure of two signatures

When the largest common structure of the observation and the rule antecedent has been defined, both the signature of the rule antecedent and the signature of the observation have to be reduced to this largest common structure using the aggregation operators assigned to each node in the arbitrary structure. The signatures of the rule antecedent and the observation after the reduction are denoted by $A_i^{(r)}$ and $A^{'(r)}$ respectively.

4.1.2. Constructing the Signature Representing the Degree of Matching

At this point, the signature structures of the rule antecedent and of the observation are identical, which means that they have the leaves of their structures at the same levels. This means that the indexes of the leaves of both structures are the same. Let us refer to the leaves at equivalent levels as corresponding leaves.

The signature representing the degree of matching (denoted by M_i) between rule R_i and the observation has the same structure as the largest common structure. The values on its leaves are obtained by calculating the degree of matching between the corresponding leaves of the two structures. This is done by applying the formula in (10), where l_j denotes a leaf with index j and $A(l_j)$ denotes the value on leaf l_j of fuzzy signature A. The function $W(x_j,y_j)$ gives the degree of matching between two corresponding leaves.

$$\forall\ leaf\ l_j,\ M_i(l_j) = W\left(A^{'(r)}(l_j), A_i^{(r)}(l_j)\right) \tag{10}$$

Figure 12 illustrates the construction of the signature representing the degree of matching.

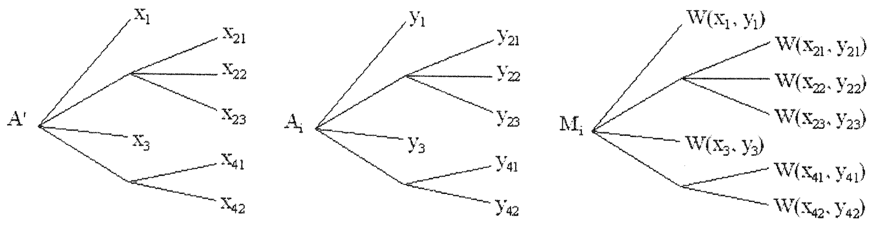


Figure 12. Constructing the signature representing the degree of matching

The degree of matching between two fuzzy sets is the maximal value of the t-norm (in this case the minimum operator is used) of the two membership functions (as seen in Figure 6). For defining the degree of matching between a fuzzy set and a crisp value, the membership grade of the given fuzzy set has to be taken at the specified crisp value. The degree of matching of two crisp values can be obtained by calculating the equivalence between the two values. In our work, the formula given in (11) was used, where *c* is a fuzzy complement, *t* is a t-norm and *s* is an s-norm. Zadeh's operator triplet (1-x, minimum, maximum) was used.

$$W(x_j, y_j) = s(t(x_j, y_j), t(c(x_j), c(y_j))) = \max(\min(x_j, y_j), \min(1 - x_j, 1 - y_j)) \quad (11)$$

4.1.3. Calculating the Degree of Matching for Rule Ri

The degree of matching between the observation and rule R_i is obtained by reducing the signature representing the degree of matching (M_i) that was constructed in the previous section to its root node. The aggregation operators defined in the arbitrary signature structure assigned to the original fuzzy signature based rule base are used in the aggregation. Also the relevance weights specified for the nodes in the arbitrary structure may be used.
The degree of matching between the observation and rule R_i is denoted by w_i.

4.2. Inferring the Consequent Fuzzy Set

After the separate degrees of matching between each rule of the rule base (R_i, $i=1,\dots,r$) and the fuzzy signature observation (A') has been obtained (these degrees are denoted by $w_1,\dots,w_r$), the final conclusion is calculated in the same way as in the original Mamdani method (see Section 2.2.2).

4.3. Defuzzification

As the result of the fuzzy signature based inference, a conclusion fuzzy set is obtained. If a crisp output of the system is expected, then this fuzzy set has to be defuzzified using one of the methods given in Section 2.2.3

5. Software Implementation

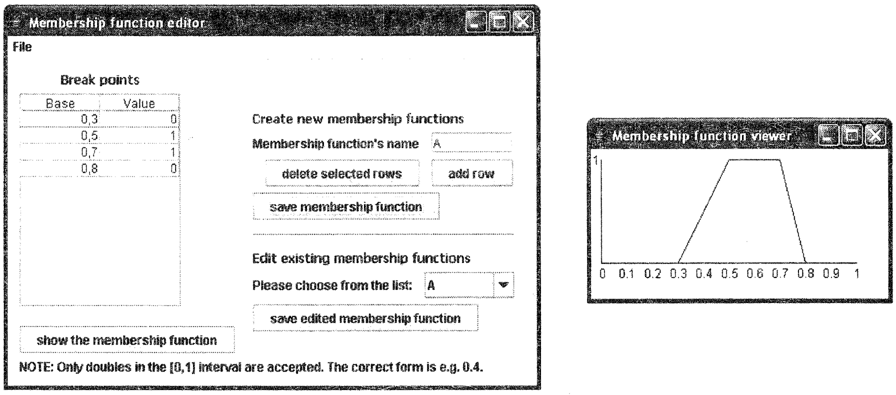
A software capable of performing inference in fuzzy signature based rule bases was implemented using the Java programming language. This software can store some basic types of fuzzy sets, it can handle fuzzy signature arbitrary structures (with aggregation operators and relevance weights for each node) and fuzzy signature based rules. Generalized Mamdani-type inference can be executed automatically for a given observation based on the previously allocated rule base. The result of the inference (including the conclusion fuzzy set and its defuzzification) is visualized.

A graphical user interface facilitating the software's usage is also available.

5.1. Description of the Main Components

5.1.1. Fuzzy Sets

In the software implementation piecewise linear fuzzy sets can be specified by defining the breakpoints of the membership function. The break points are stored in a table, to which new break points can be added one by one. The actual membership function can be visualized at any point.


Membership function editor
File
Create new membership functions
delete selected rows
add row
save membership function
Edit existing membership functions
save edited membership function
show the membership function
Membership function viewer
1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Break points

Base	Value
0,3	0
0,5	1
0,7	1
0,8	0

Membership function's name: A

Please choose from the list: A

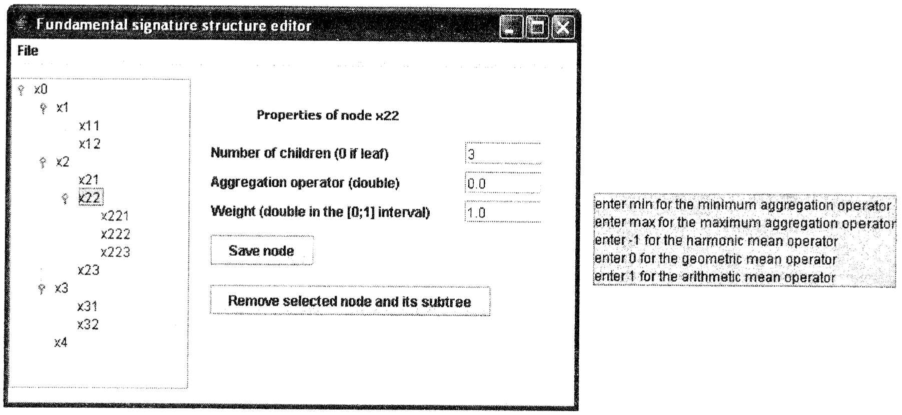
NOTE: Only doubles in the [0,1] interval are accepted. The correct form is e.g. 0.4.

Figure 13. The membership function editor and viewer

5.1.2. Arbitrary Structure of Fuzzy Signatures

To define the arbitrary structure of the fuzzy signatures involved in the system, the structure itself has to be built (from the root downwards), by specifying the number of children of each node. When specifying this number, the aggregation operator and weight of the node can also be defined. The arbitrary structure cannot be saved until an aggregation operator and a relevance weight have not been specified for all nodes.

In the software only Weighted Relevance Aggregation Operators can be specified for each node, by defining the aggregation factor p in Equation 4.


Fundamental signature structure editor
File
x0
x1
x11
x12
x2
x21
x22
x221
x222
x223
x23
x3
x31
x32
x4
Save node
Remove selected node and its subtree

Properties of node x22

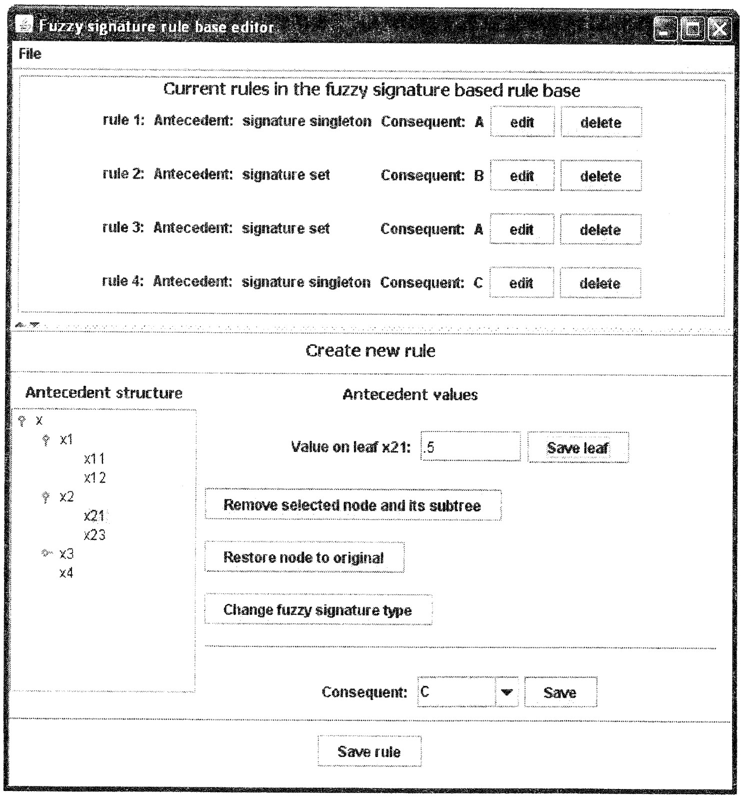
Number of children (0 if leaf) 3
Aggregation operator (double) 0.0
Weight (double in the [0;1] interval) 1.0

enter min for the minimum aggregation operator
enter max for the maximum aggregation operator
enter -1 for the harmonic mean operator
enter 0 for the geometric mean operator
enter 1 for the arithmetic mean operator

Figure 14. Arbitrary signature structure editor

5.1.3. Fuzzy Signature Based Rules

Fuzzy signature based rules can be defined in the software implementation by specifying the rule antecedent (a fuzzy signature) and the consequent (a fuzzy set).



Current rules in the fuzzy signature based rule base

rule 1: Antecedent: signature singleton Consequent: A
rule 2: Antecedent: signature set Consequent: B
rule 3: Antecedent: signature set Consequent: A
rule 4: Antecedent: signature singleton Consequent: C

Value on leaf x21: .5

Consequent: C

Figure 15. Fuzzy signature based rule base editor

When creating a new rule, either a fuzzy signature set or a fuzzy signature singleton can be chosen as the rule antecedent. The arbitrary structure of the antecedent fuzzy signature is a given, but if not all leaves or sub-trees are needed, then they can be easily deleted from the structure. After the desired structure has been obtained, a fuzzy set or a fuzzy value has to be specified for all the leaves of the modified structure, depending on the type of the fuzzy signature.

5.1.4. Fuzzy Signature Observation

A fuzzy signature observation can either be specified as a fuzzy signature set, or a fuzzy signature singleton. The arbitrary structure of the observation is also a given, but some leaves or sub-trees may be deleted if they are not needed, which means that they could not be observed in the specific observation.

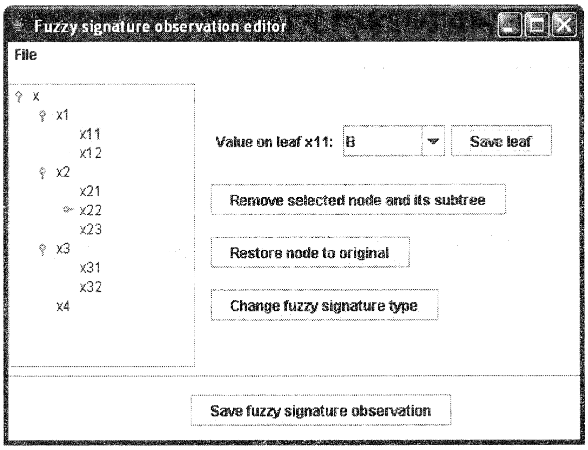
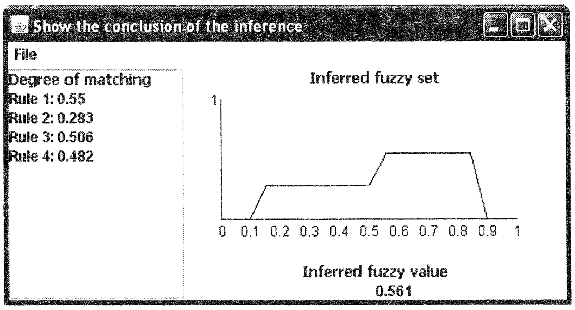

Fuzzy signature observation editor
File
x
x1
x11
x12
x2
x21
x22
x23
x3
x31
x32
x4
Value on leaf x11: B
Save leaf
Remove selected node and its subtree
Restore node to original
Change fuzzy signature type
Save fuzzy signature observation

Figure 16. Fuzzy signature observation editor

5.1.5. Generalized Mamdani-type Inference

After all the rules have been added to the rule base, and the observation is specified, the inference process can be launched. The software calculates all the degrees of matching which it then uses to compute the conclusion fuzzy set. A crisp conclusion is also computed.



Degree of matching
Rule 1: 0.55
Rule 2: 0.283
Rule 3: 0.506
Rule 4: 0.482

Inferred fuzzy value
0.561

Figure 17. Generalized Mamdani-type inference conclusion viewer

On the graphical user interface the degrees of matching, the conclusion fuzzy set and the crisp conclusion can all be seen (as shown in Figure 17).

6. Application on a Realistic Example

The use of the extended Mamdani-method in fuzzy signature based models will be shown in the following problem taken from the construction industry (see also [1]).
In Hungary there are many road bridges, whose state differs greatly when taking into account their age, width, structural properties and so on. The scheduling of maintenance on these bridges is a very important task. To be able to determine the importance of intervention, many constant parameters have to be stored, while the variable parameters have to be measured on a regular basis. For each bridge a set of parameters are available, where the data components can be organized into a hierarchy, in this way fuzzy signatures can be used for modeling this problem.

6.1. Fuzzy Signature Model

For modeling the given example with fuzzy signatures, first the arbitrary signature structure characterizing the problem has to be determined from the available parameters, while taking into account the expert knowledge about the problem. Secondly the domains of the parameters and the respective fuzzy sets which will be used in the model have to be assigned. Then fuzzy signature based rules have to be built using the previously computed input-output pairs, which were determined using expert knowledge. When the rules in the rule base have been set, an observation representing the actual state of a bridge can be given, and a conclusion for this bridge can be calculated, which gives the importance of maintenance on the given bridge.

6.1.1. Arbitrary Structure

The information available about a bridge can be divided into three main groups. The first group of parameters defines the overall state of the bridge, the second group defines the service level, while the third group contains parameters defining the sensitivity of the structure. These features define the first level of the fuzzy signature arbitrary structure (nodes x_1, x_2, x_3 in Figure 18).

The overall state of the bridge can be typified with five components, which are the following:

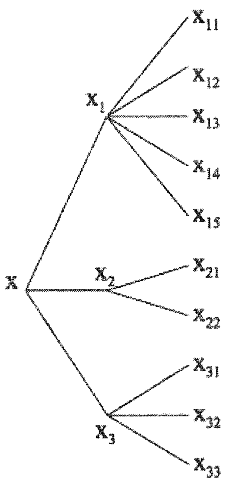
- state of the upper-structure: quality of the bridge and the bearer, etc.
- state of the substructure: quality of the foundation, the stanchion, etc.
- state of the deck: quality of the tarmac and other elements used by traffic
- state of the fittings: quality of the railings, steps, etc.
- state of the surroundings: quality of the road signs and public utilities taken across bridge

These features define the sub-tree of the node defining the overall state of the bridge (x_1), and are marked by nodes x_{11}, x_{12}, x_{13}, x_{14}, x_{15} in Figure 18. These features could have sub-trees of their own, in this way giving more specific information of the state of the bridge.

The service level of the bridge can be represented by two components, the load bearing and the width of the bridge. These features define the sub-tree of the node defining the service level of the bridge (x_2), and are marked by nodes x_{21} and x_{22} respectively in Figure 18.

The sensitivity of the structure of a bridge can be given using three components, the maintenance technology, the age and the sensitivity of the material of the structure. The maintenance technology in this case defines the rate of the use of salt in winter. These three features define the sub-tree of the node defining the sensitivity of the structure (x_3), and are marked by nodes x_{31}, x_{32} and x_{33} in Figure 18.

The arbitrary fuzzy signature structure of the model can be seen in Figure 18. The aggregation operators and the relevance weights of the nodes were adjusted by an expert in the given field (*cf.* [1]).



$$x = \begin{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{bmatrix} \\ \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} \\ \begin{bmatrix} x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} \end{bmatrix}$$

Figure 18. Arbitrary fuzzy signature structure

6.1.2. Domains and Fuzzy Sets

To describe the components of the fuzzy signature structure, linguistic fuzzy variables (and their respective membership functions) need to be defined over the specific domains of each component. The domains and fuzzy sets specified for each component are not necessarily different.

As mentioned when introducing fuzzy signature sets, the fuzzy sets on the leaves of the structure have to be over the [0,1] domain, therefore the domains of each component of the arbitrary fuzzy signature have to be normalized to the [0,1] interval. For example, the width of a bridge could be in the 300 cm to 7000 cm range. A bridge with a width of 3000 cm could be described with a fuzzy value of 0.4 after normalization.

To describe, for example the state of a bridge, five linguistic fuzzy variables may be used. These could be the following: “perfect”, “good”, “decent”, “acceptable”, and “bad”.

The output of the inference system, *i.e.* the importance of maintenance could also be described by five linguistic fuzzy variables. These could be: “not necessary”, “can be delayed”, “within 10 years”, “within 3 years” and “immediately”. The greater the fuzzy value, the more necessary the intervention is.

6.1.3. Fuzzy Signature Based Rule Base

In the example the fuzzy signature based rules used in the rule base were extracted from the input-output pairs received from the domain experts. These input-output pairs were determined by taking a bridge with all its constant and variable parameters as an input, and determining the importance of maintenance using an algorithm defined by experts in the bridge maintenance field.

The antecedent parts of the fuzzy signature based rules found in the rule base are fuzzy signatures, as mentioned previously. The antecedent fuzzy signature can be constructed from the input set of variables by first determining the structure of the signature by taking into account which components of the arbitrary structure are present in the actual input. When the structure has been set, the components have to be fuzzified according to their domains used in the previous section, and the actual fuzzy signature has to be constructed.

The consequent part of the rule is one of the five fuzzy sets mentioned above. For every input, this conclusion fuzzy set is the one that fits best the output calculated using the expert knowledge.

When all input-output pairs were converted to fuzzy signature based rule form, the rule base is generated.

6.2. Inference for an Example Fuzzy Signature Observation

The example fuzzy signature observation for the fuzzy inference system will be specified in fuzzy signature singleton form. Let us assume that the fuzzy signature model of the example is already loaded in the software.

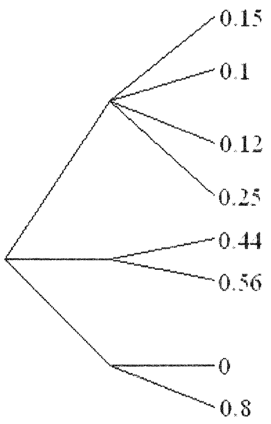
The bridge used for the observation can be described with the following set of parameters from which the necessity of maintenance has to be determined.

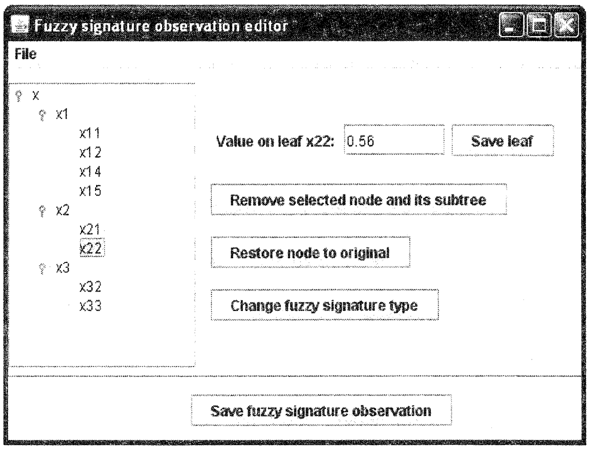
The overall state of the bridge is only described by four parameters, because in the example no measurements were carried out regarding the state of the deck. The state of the upper-structure, the substructure and of the fittings is almost perfect, thus the values 0.15, 0.1 and 0.12 are used to describe it. The state of the surroundings is good, so the value 0.25 was used.

The values describing the service level of the bridge can be calculated by taking the measured parameters and normalizing them over the [0,1] domain. The load bearing of the bridge in the example is 40 tons, which is normalized to 0.44. The width of the bridge is measured as 750 cm, which is normalized to 0.56.

The parameters describing the sensitivity of the structure also need to be normalized. For the bridge used in the example, no information is available for the maintenance technology, so leaf x_{31} is missing from the observation. The bridge in question was built in 1900, this date is normalized to 0. The bridge was built from stone so the sensitivity of the material of the structure is defined by the value 0.8.

Once these values have been determined, the fuzzy signature singleton representing the observation can be built. The signature with the relevant values on the leaves of the structure, along with the fuzzy signature observation defined in the software can be seen in Figure 19.

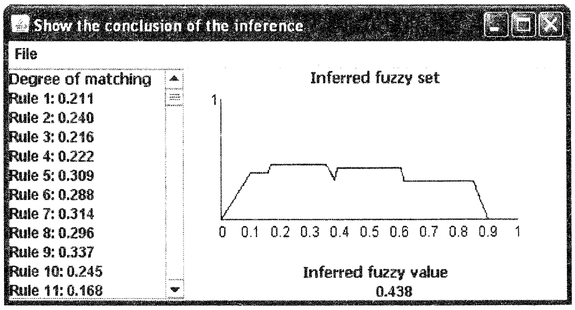



Fuzzy signature observation editor
File
x
x1
x11
x12
x14
x15
x2
x21
x22
x3
x32
x33
Save leaf
Remove selected node and its subtree
Restore node to original
Change fuzzy signature type
Save fuzzy signature observation

Value on leaf x22: 0.56

Figure 19. Example fuzzy signature observation in tree form and in the observation editor

After saving the fuzzy signature observation, the conclusion is calculated by clicking a button in the implemented software. The window seen in Figure 20 appears. In the column on the left hand side, the degrees of matching between each rule antecedent and the observation can be seen, while on the right the inferred fuzzy set and the defuzzified conclusion for the given observation are shown.



Degree of matching
Rule 1: 0.211
Rule 2: 0.240
Rule 3: 0.216
Rule 4: 0.222
Rule 5: 0.309
Rule 6: 0.288
Rule 7: 0.314
Rule 8: 0.296
Rule 9: 0.337
Rule 10: 0.245
Rule 11: 0.168

Inferred fuzzy value
0.438

Figure 20. The conclusion of the inference for the given observation

The defuzzified value can be interpreted in a way that the maintenance of the given bridge is only necessary "within 10 years". The conclusion fuzzy set itself does not give such an unambiguous result, as its support is very wide, and the values are about the same over the whole domain.

The result of the inference is sensitive to any change made in the relevance weights or the aggregation operators specified for the nodes of the fuzzy signature structure.

The generalized Mamdani-type inference method for fuzzy signature based rule bases could be improved greatly by developing a learning method for adjusting the relevance weights and the aggregation operators according to the available information stored in the input-output data pairs used to build the rule base.

7. Conclusions

In this paper the notion of fuzzy signatures was introduced and their main advantages were stated. The most important operator, the aggregation operator used for structure modification was presented. A brief overview of fuzzy inference systems was given and Mamdani-type inference systems were shown step-by-step. After this, a way of defining fuzzy signature based models was given, and the generalized Mamdani-type inference on fuzzy signature based models was introduced. A software capable of calculating a conclusion for a fuzzy signature observation was implemented. The use of the software was explained through a realistic example taken from the bridge maintenance field. Our results show that the method gives acceptable results on the test cases, where rules with slightly differently structured rule antecedents were used in the rule base, and the signature of the observation was different too. This shows that the principles for structure modification and inference are fitting, although the results could be further improved by modifying certain parameters of the model (the relevance weights and the aggregation operators in the arbitrary fuzzy signature structure). In the future a method for learning these key parameters from the input-output pairs used for modelling could be developed and later implemented.

Acknowledgement

This paper was supported by the Széchenyi University Main Research Direction Grant 2008, and a National Scientific Research Fund Grant OTKA T048832.

References

[1] Agárdy, G.: *Possibilities of use of fuzzy logic and fuzzy techniques in bridge-management systems,* Acta Technica Jaurinensis, Series Intelligentia Computatorica (2008)

[2] Goguen, J. A.: *L-fuzzy Sets*, Journal of Mathematical Analysis and Applications, vol. 18, (1967) pp. 145-174

[3] Kóczy, L. T.: *Vector Valued Fuzzy Sets,* Busefal, eté, (1980) pp. 41-57

[4] Kóczy, L. T., Vámos, T., Biró, G.: *Fuzzy Signatures,* Proceedings of EUROFUSE-SIC'99, Budapest, Hungary, (1999) pp. 210-217

[5] Mamdani, E. H., Assilian, S.: *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller,* International Journal of Man-Machine Studies, vol. 7, (1975) pp. 1-13

[6] Mendis, B. S. U., Gedeon, T. D., Kóczy, L. T.: *Investigation of aggregation in fuzzy signatures,* Proceedings of the 3rd International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore, (2005)

[7] Mendis, B. S. U., Gedeon, T. D., Kóczy, L. T.: *On the issue of learning weights from observations for fuzzy signatures,* Proceedings of theWorld Automation Congress 2006, Budapest, Hungary, (2006) pp. 1–6

[8] Mendis, B. S. U., Gedeon, T. D., Botzheim, J., Kóczy, L. T.: *Generalised Weighted Relevance Aggregation Operators for Hierarchical Fuzzy Signatures,* Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation, Sydney, Australia (2006)

[9] Tamás, K., Kóczy, L. T.: *Mamdani-type inference in fuzzy signature based rule bases,* Proceedings of the 8th International Symposium of Hungarian Researchers, Hungary, (2007) pp. 513–525

[10] Tamás, K., Kóczy, L. T.: *Selection from a fuzzy signature database by Mamdani-algorithm,* Proceedings of the 6th International Symposium on Applied Machine Intelligence and Informatics, Herlány, Slovakia, (2008) pp. 63–68

[11] Wong, K. W., Chong, A., Gedeon, T. D., Kóczy, L. T., Vámos, T.: *Hierarchical fuzzy signature structure for complex structured data,* Proceedings of the International Symposium on Computational Intelligence and Intelligent Informatics, Nabeul, Tunisia, (2003) pp. 105–109

[12] Zadeh, L. A.: *Fuzzy sets,* Information and Control, vol. 8, (1965) pp. 338–353

[13] Zadeh, L. A.: *Outline of a new approach to the analysis of complex systems and decision processes,* IEEE Transactions on Systems, Man and Cybernetics 3, (1973) pp. 28-44

Fuzzy Rule Interpolation from a Practical Point of View

Szilveszter Kovács

Department of Information Technology, University of Miskolc
Miskolc-Egyetemvaros, Miskolc, Hungary, H-3515
e-mail: szkovacs@iit.uni-miskolc

Abstract: The main goal of this paper to give a short survey on various Fuzzy Rule Interpolation (FRI) methods together with a simple demonstration of their application benefits. There are relatively few FRI techniques used in practical fuzzy rule based applications. On one hand the FRI methods are not widely known, and many of them have limitations from practical application point of view, e.g. can be applied only in one dimensional case, or defined based on the two closest surrounding rules of the actual observation. On the other hand enabling the application of sparse rule bases the FRI methods can dramatically simplify methods of fuzzy rule base creation, since FRI methods can provide reasonable (interpolated) conclusions even if none of the existing rules fires under the current observation. These methods can save the expert from dealing with derivable rules and help to concentrate on cardinal actions only and hence simplify the rule base creation itself. Thus, compared to the classical fuzzy CRI, the number of the fuzzy rules needed to be handled during the design process, could be dramatically reduced. In this paper, among the brief structure of several FRI methods, a simple and quick FRI method "FIVE" will be introduced in more detail, and for demonstrating the benefits of the interpolation-based fuzzy reasoning as systematic approach, the construction of a fuzzy rule base through a simple example will be also discussed.

Keywords: Fuzzy Rule Interpolation (FRI), Interpolation-based fuzzy reasoning, FRI applications

1. Inroduction

Since the classical fuzzy reasoning methods (e.g. compositional rule of inference (CRI)) are demanding complete rule bases, the classical rule base construction claims a special care of filling all the possible rules. In case if there are some rules missing (the rule base is "sparse"), observations may exist which hit no rule in the rule base and therefore no conclusion is obtained. Having no conclusion in a fuzzy control structure is hard to explain. E.g. one solution could be to keep the last real conclusion instead of the missing one, but applying historical data automatically to fill missing rules could cause unpredictable side effects. Another solution for the same problem is the application of the fuzzy rule interpolation (FRI) methods, where the derivable rules are deliberately missing, since FRI methods can provide reasonable (interpolated) conclusions even if none of the existing rules fires under the current observation. The rule base of an FRI controller is not necessarily complete; hence it could contain the most significant fuzzy rules only without risking the chance of having no conclusion for some of the observations. On the other hand most of the FRI methods share the burden of high computational demand, e.g. the task of searching for the two closest surrounding rules to the observation, and calculating the conclusion at least in some characteristic α-cuts. Moreover in some methods the interpretability of the fuzzy conclusion gained is also not straightforward [14]. There have been a lot of efforts to rectify the interpretability of the interpolated fuzzy conclusion [24]. In [1] Baranyi *et al.* give a comprehensive overview of the recent existing FRI methods. In [4] Johanyák and Kovács surveyed and evaluated a wide range of FRI methods from an application oriented point of view. Beyond these problems, some of the FRI methods are originally defined for one dimensional input space, and need special extension for the multidimensional case (e.g. [2]-[3]). In [29] Wong *et al.* gave a comparative overview of the recent multidimensional input space capable FRI methods. In [2] Jenei suggested the comprehensive axiomatic investigation of the FRI methods. The high computational demand, mainly the search for the two closest surrounding rules to an arbitrary observation in the multidimensional antecedent space makes many of these methods hardly suitable for real-time applications. Some FRI methods, e.g. LESFRI [5] or the method introduced by Jenei *et al.* in [3], eliminate the search for the two closest surrounding rules by taking all the rules into consideration, and therefore speed up the reasoning process. On the other hand, keeping the goal of constructing fuzzy conclusion, and not simply speeding up the reasoning process, they still require some additional (or repeated) computational steps for the elements of the level set (or at least some relevant α levels). A rather different application oriented aspect of the FRI emerges in the concept of "FIVE" (Fuzzy Interpolation based on Vague Environment). In the followings, among the brief structure of several FRI methods, the "FIVE" will be introduced in more details.

2. A brief overview of several FRI techniques

One of the first FRI techniques was published by Kóczy and Hirota [12]. It is usually referred as the *KH method*. It is applicable to convex and normal fuzzy (CNF) sets. It determines the conclusion by its α-cuts in such a way that the ratio of distances between the conclusion and the consequents should be identical with the ones between the observation and the antecedents for all important α-cuts. The applied formula

$$d(A^*,A_1):d(A^*,A_2)=d(B^*,B_1):d(B^*,B_2),$$

can be solved for B^* for relevant α-cuts after decomposition.

It is shown in, e.g. in [14], [15] that the conclusion of the KH method is not always directly interpretable as fuzzy set. This drawback motivated many alternative solutions. A modification was proposed by Vass, Kalmár and Kóczy [27] (*VKK method*), where the conclusion is computed based on the distance of the centre points and the widths of the α-cuts, instead of lower and upper distances. VKK method decreases the applicability limit of KH method, but does not eliminate it completely. The technique cannot be applied if any of the antecedent sets is singleton (the width of the antecedent's support must be nonzero). In spite of the disadvantages, KH is popular because its simplicity that infers its advantageous complexity properties. It was generalized in several ways. Among them the *stabilized KH interpolator* is emerged, as it is proved to hold the universal approximation property [26], [23]. This method takes into account all flaking rules of an observation in the calculation of the conclusion in extent to the inverse of the distance of antecedents and observation. The universal approximation property holds if the distance function is raised to the power of the input's dimension.

Another modification of KH is the modified alpha-cut based interpolation (*MACI*) method [24], which alleviates completely the abnormality problem. MACI's main idea is the following: it transforms fuzzy sets of the input and output universes to such a space where abnormality is excluded, then computes the conclusion there, which is finally transformed back to the original space. MACI uses vector representation of fuzzy sets and originally applicable to CNF sets [30]. These latter conditions (CNF sets) can be relaxed, but it increases the computational need of the method considerably [25]. MACI is one of the most applied FRI methods [29], since it preserves advantageous computational and approximate nature of KH, while it excludes its abnormality.

Another fuzzy interpolation technique was proposed by Kóczy et al. [13]. It is called conservation of "relative fuzziness" (*CRF*) method, which notion means that the left (right) fuzziness of the approximated conclusion in proportion to the flanking fuzziness of the neighboring consequent should be the same as the (left) right fuzziness of the observation in proportion to the flanking fuzziness of the neighboring antecedent. The technique is applicable to CNF sets.

An improved fuzzy interpolation technique for multidimensional input spaces (*IMUL*) was proposed in [28], and described in details in [29]. IMUL applies a combination of CRF and MACI methods, and mixes advantages of both. The core of the conclusion is determined by MACI method, while its flanks by CRF. The main advantages of this method are its applicability for multi-dimensional problems and its relative simplicity.

Conceptually different approaches were proposed by Baranyi *et al* [1] based on the relation and on the semantic and inter-relational features of the fuzzy sets. The family of these methods applies "General Methodology" (GM); this notation also reflects to the feature that these methods are able to process arbitrary shaped fuzzy sets. The basic concept is to calculate the reference point of the conclusion based on the ratio of the distances between the reference points of the observation and the antecedents. Then, a single rule reasoning method (revision function) is applied to determine the final fuzzy conclusion based on the similarity of the fuzzy observation and an "interpolated" observation. The methods proposed by Johanyák and Kovács (LESFRI [5], FRIPOC [6] and VEIN [7]) also follow the two-step concept of GM. LESFRI applies the method of least squares for the determination of the shape of the conclusion. FRIPOC introduces the concept of polar cut applying a polar coordinate system in course of the calculations. VEIN solves the task of rule interpolation in the vague environment similar to FIVE.

3. A simple and quick FRI method: "FIVE"

A rather different application oriented aspect of the fuzzy rule interpolation emerges in the concept of *FIVE*. The fuzzy reasoning method "FIVE" (Fuzzy Interpolation based on Vague Environment, originally introduced in [16], [17] and [18]) was developed to fit the speed requirements of direct fuzzy control, where the conclusions of the fuzzy controller are applied directly as control actions in a real-time system.

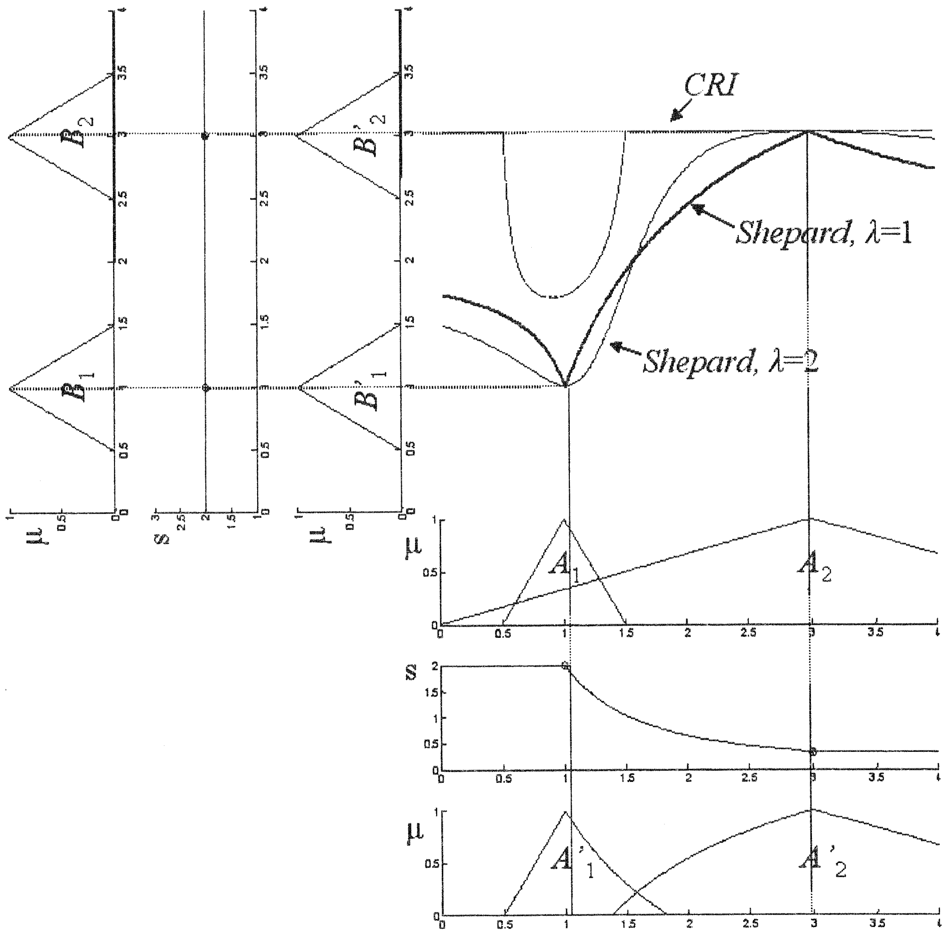


Fig. 1: Interpolation of two fuzzy rules (R_i: $A_i \rightarrow B_i$), by the Shepard operator based FIVE, and for comparison the min-max CRI with COG defuzzification.

The main idea of the FIVE is based on the fact that most of the control applications serve crisp observations and require crisp conclusions from the controller. Adopting the idea of the vague environment (VE) [18], FIVE can handle the antecedent and consequent fuzzy partitions of the fuzzy rule base by scaling functions [18] and therefore turn the fuzzy interpolation to crisp interpolation.

The idea of a VE is based on the similarity (in other words: indistinguishability) of the considered elements. In VE the fuzzy membership function $\mu_A(x)$ indicates the level of similarity of x to a specific element a that is a representative or prototypical element of the fuzzy set $\mu_A(x)$, or, equivalently, as the degree to which x is indistinguishable from a [18]. Therefore the α-cuts of the fuzzy set $\mu_A(x)$ are the sets which contain the elements that are $(1-\alpha)$-indistinguishable from a. Two values in a VE are ε-distinguishable if their distance is greater than ε. The distances in a VE are weighted distances. The weighting factor or function is called *scaling function (factor)* [18]. If VE of a fuzzy partition (the scaling function or at least the approximate scaling function [16], [18]) exists, the member sets of the fuzzy partition can be characterized by points in that VE (see e.g. scaling function s on fig. 1). Therefore any crisp interpolation, extrapolation, or regression method can be adapted very simply for FRI [16], [18]. Because of its simple multidimensional applicability, in FIVE the *Shepard operator* based interpolation (first introduced in [22]) is adapted (see e.g. fig. 1).

In this case if the fuzzy rules R_k has the following form:

If $x_1 = A_{k,1}$ **And** $x_2 = A_{k,2}$ **And** … **And** $x_m = A_{k,m}$ **Then** $y = B_k$

The FIVE interpolation can be expressed by the following formula:

$$\delta_s(b_0, y(\mathbf{x})) = \begin{cases} \delta_s(b_0, b_k) & \text{if } \mathbf{x} = \mathbf{a}_k \text{ for some } k, \\ \left(\sum_{k=1}^{r} \delta_s(b_0, b_k)/\delta_{s,k}^{\lambda}\right) \Big/ \left(\sum_{k=1}^{r} 1/\delta_{s,k}^{\lambda}\right) & \text{otherwise.} \end{cases}$$

where $y(\mathbf{x})$ is the requested one dimensional conclusion, r is the number of the fuzzy rules in the rule base R, $\lambda > 0$ is a parameter of the Shepard operator, b_0 is the first element of the one dimensional consequence universe (Y: $b_0 \leq y$, $\forall y \in Y$), and $\delta_{s,k}$, δ_s can be calculated by the following formula:

$$\delta_{s,k} = \delta_s(\mathbf{a}_k, \mathbf{x}) = \left[\sum_{i=1}^{m} \left(\int_{a_{k,i}}^{x_i} s_{X_i}(x_i)dx_i\right)^2\right]^{1/2},$$

$$\delta_s(b_0, b_k) = \int_{b_0}^{b_k} s_Y(y)dy,$$

where s_{X_i} is the i^{th} scaling function of the m dimensional antecedent universe, $\mathbf{x}$ is the m dimensional crisp observation, $\mathbf{a}_k$ are the cores of the m dimensional fuzzy rule antecedents A_k and s_Y is the i^{th} scaling function of the one dimensional consequent universe, b_k are the cores of the one dimensional fuzzy rule consequents B_k.

An implementation of FIVE as a component of the FRI Matlab Toolbox [8] can be downloaded from [31].

4. FRI rule base design example

The application example introduced in this paper for demonstrating the benefits of the interpolation-based fuzzy reasoning as systematic approach, is the fuzzy rule base construction of an automated guided vehicle (AGV) steering control [19], [20]. FRI methods have been also successfully applied in several other areas like fuzzy modeling of an anaerobic tapered fluidized bed reactor (Johanyák *et al.* [9]) or tool life modeling

(Johanyák and Szabó [10]). In the current AGV example of this paper, the steering control has two goals, the path tracking (to follow a guide path) and the collision avoidance. The guide path is usually a painted marking or an active guide-wire on the floor. The guiding system senses the position of the guide path by special sensors (*guide zone*) tuned for the guide path. The goal of the steering control is to follow the guide path by the guide zone with minimal path tracking error on the whole path (fig. 2).

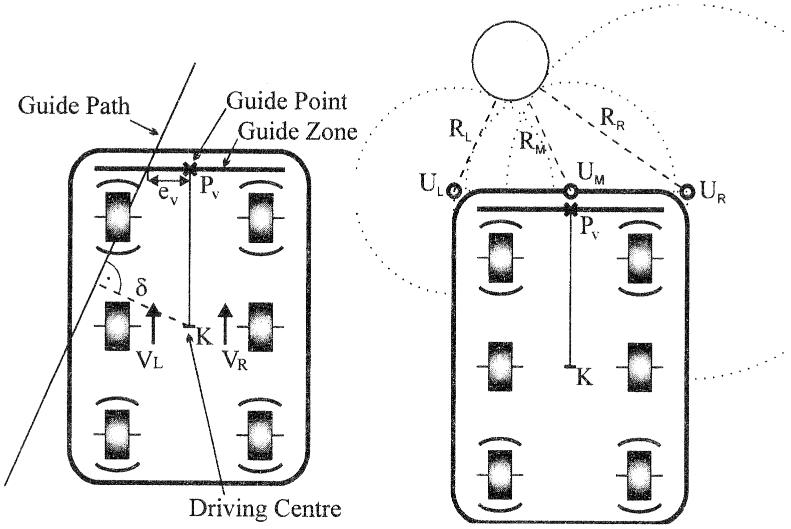


Fig.2. Differential steered AGV with guide zone, δ is the path tracking error, ev is the distance of the guide path and the guide point, Pv is the guide point, K is the driving centre, RL, RR, RM are the distances measured by the left, right and middle ultrasonic sensors (UL, UR, UM).

4.1. Path tracking, complete rule base

The design of the steering control rule base can be divided into two main steps. First the path tracking rule base needed to be elaborated and then it can be extended by the rules of collision avoidance. The simplest way of defining the fuzzy rules is based on studying the operator's control actions in relevant situations. These control actions could form the later rule base. The basic idea of the path tracking strategy is very simple: keep the driving centre K of the AGV as close as it is possible to the guide path, and than simply turn the AGV into the new direction. This strategy needs two observations: the measured distance between the guide path and the guide point (e_V), and the estimated distance between the guide path and the driving centre (δ) (see fig. 2 for the notation). Based on these observations, as a conclusion, the level of steering ($V_d = V_L - V_R$) needed to be calculated. Collecting the operator's control actions, the path tracking strategy can be characterized by five relevant situations, collected on fig. 3.

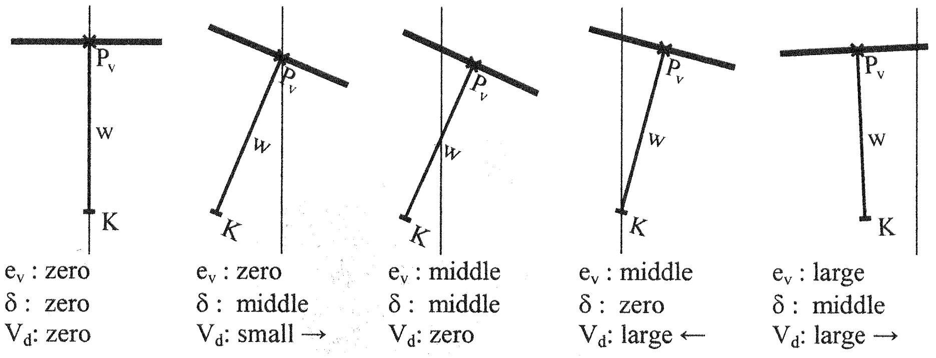


Fig.3. Relevant control actions (Vd: steering) characterizing the path tracking strategy (see Fig. 2. for the notation).

Having the relevant control actions and the linguistic term fuzzy sets (fuzzy partitions) of the two antecedent and one consequent universes, the fuzzy rule base can be simply constructed. The ith rule of the rule base has the form:

$R_{Vd(i)}$: If $e_v=A_{1,i}$ **And** $\delta=A_{2,i}$ **Then** $V_d=B_i$.

Let us have the linguistic term fuzzy partitions built up five fuzzy sets, namely: negative large (*NL*), negative middle (*NM*), zero (*Z*), positive middle (*PM*), positive large (*PL*) for the two antecedents universes (e_v, δ), and negative large (*NL*), negative small (*NS*), zero (*Z*), positive small (*PS*), positive large (*PL*) for the consequent universe (V_d). Building a complete fuzzy rule base first, according to the antecedent terms, we have to set up an antecedent grid of all possible fuzzy rules, and then fill it with the corresponding rule consequents (see Table 1.). First we can fill the rule consequents already known as relevant situations from the knowledge acquisition phase (noted by underline on Table 1.), than to make the rule base complete, the "filling" rules too.

In most cases, the "filling" rules have the only task to get "smooth transient" between the relevant rules. Selecting a fuzzy reasoning method, e.g. the max-min Compositional Rule of Inference (CRI), and center of gravity (COG) defuzzification, the control surface of the steering can be directly calculated (see fig. 4).

Table 1: Path tracking, complete rule base

R_{Vd}:

		e_v =				
		NL :	*NM* :	*Z* :	*PM* :	*PL* :
δ =	*NL* :	$\underline{PL}$	*PS*	*Z*	*NS*	$\underline{NL}$
	NM :	$\underline{PL}$	*PS*	$\underline{PS}$	$\underline{Z}$	$\underline{NL}$
	Z :	$\underline{PL}$	$\underline{PL}$	$\underline{Z}$	$\underline{NL}$	$\underline{NL}$
	PM :	$\underline{PL}$	$\underline{Z}$	$\underline{NS}$	*NS*	$\underline{NL}$
	PL :	$\underline{PL}$	*PS*	*Z*	*NS*	$\underline{NL}$

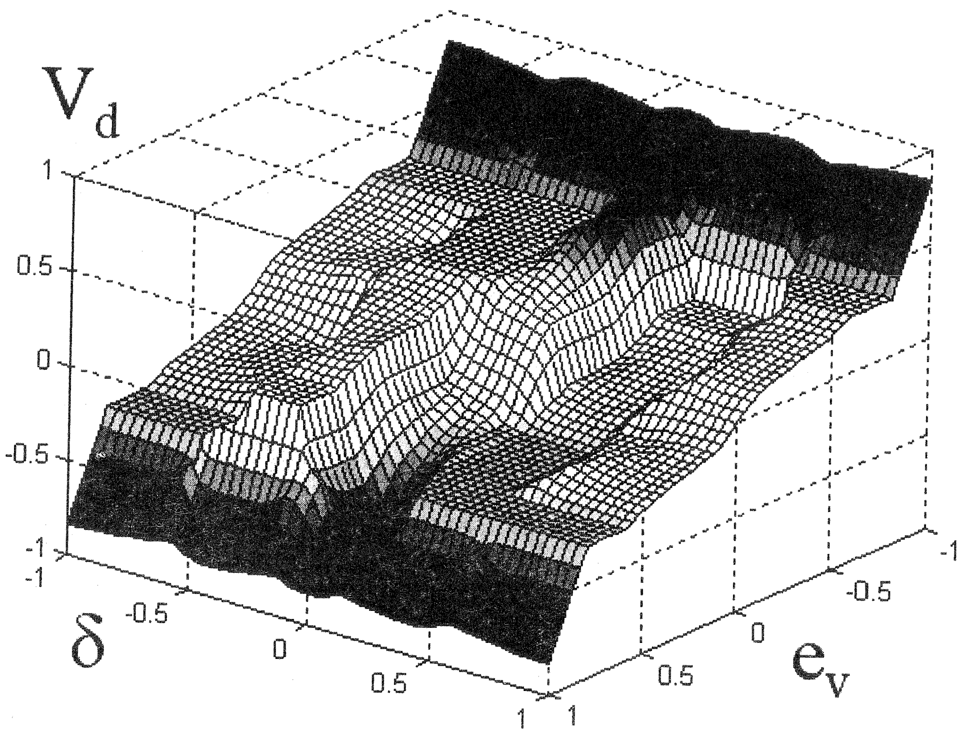


Fig.4. Control surface of the path tracking steering strategy, max-min CRI, centre of gravity defuzzification, 25 rules, complete rule base (Table 1.)

4.2. Path tracking, sparse rule base

The design of the path tracking steering control rule base for FRI is very similar to the complete rule base situation. The main difference is the lack of the "filling" rules. The rule base contains the rules of the relevant situations, known from the knowledge acquisition phase, only (see Table 2).

Introducing single antecedent rules, rules which have the same conclusion independently from some of the antecedents can be merged to single rules i.e. according to our example the rule base of Table 2 can be simplified to Table 3.

Selecting a fuzzy reasoning method suitable for sparse rule bases, i.e. in our case the "FIVE" FRI, introduced in Section 3, after constructing the scaling functions of the antecedent and consequent universes based on their fuzzy partitions (see fig. 6, fig. 7, fig. 8), the control surface of the steering can be directly calculated (see fig. 5). Comparing the control surfaces on fig.4 and fig.5 (the classical max-min CRI and COG vs. "FIVE"), one may see the function generated by the FRI is smoother. This feature is inherited from the applied Shepard operator and has little effect on the performance of the application itself.

Table 2. Path tracking, sparse rule base

R_{Vd}:

		e_v = *NL* :	*NM* :	*Z* :	*PM* :	*PL* :
δ =	*NL* :	*PL*				*NL*
	NM :	*PL*		*PS*	*Z*	*NL*
	Z :	*PL*	*PL*		*NL*	*NL*
	PM :	*PL*	*Z*	*NS*		*NL*
	PL :	*PL*				*NL*

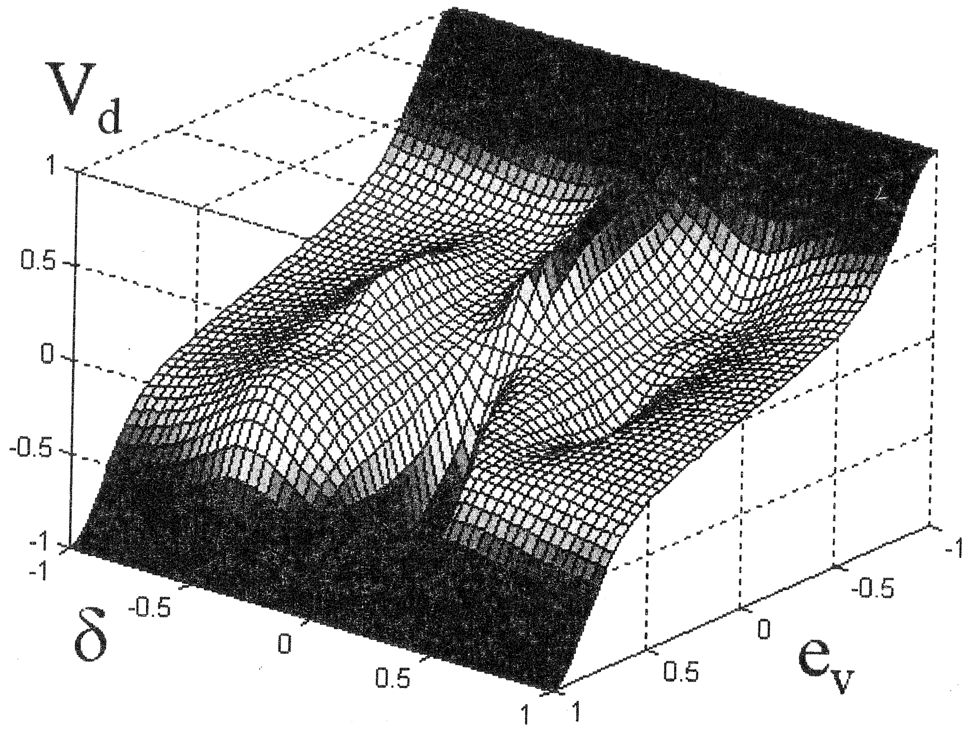


Fig.5. Control surface of the path tracking steering strategy, "FIVE" FRI, 8 rules, sparse rule base, according to Table 3.

Table 3. Path tracking, sparse rule base

R_{Vd}:	e_v	δ	V_d
Rule 1:	*NL*		*PL*
Rule 2:	*PL*		*NL*
Rule 3:	*NM*	*Z*	*PL*
Rule 4:	*PM*	*Z*	*NL*
Rule 5:	*NM*	*PM*	*Z*
Rule 6:	*PM*	*NM*	*Z*
Rule 7:	*Z*	*PM*	*NS*
Rule 8:	*Z*	*NM*	*PS*

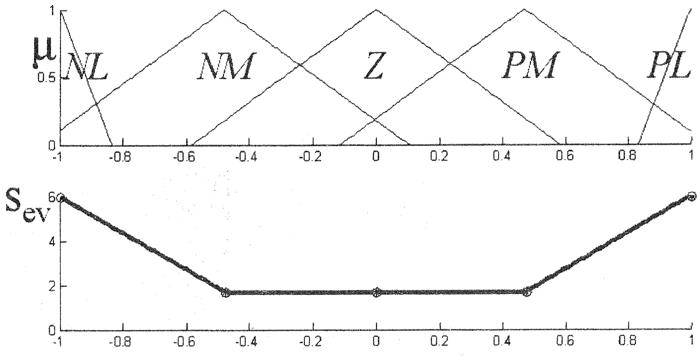


Fig. 6. The ev antecedent fuzzy partition and its scaling function sev.

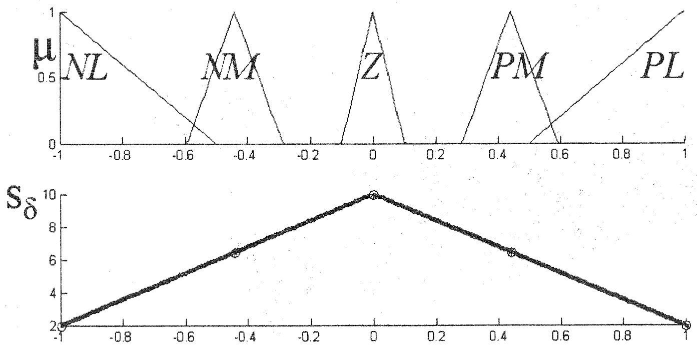


Fig. 7. The δ antecedent fuzzy partition and its scaling function sδ.

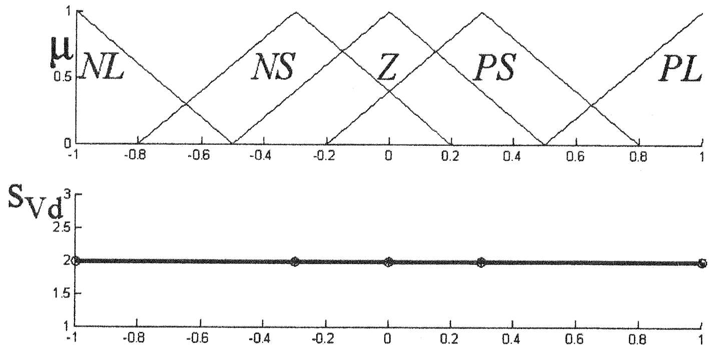


Fig. 8. The Vd consequent fuzzy partition and its scaling function sVd.

4.3. Path tracking, and collision avoidance, sparse rule base

For extending the path tracking rule base by the rules of collision avoidance, first the new observations required for detecting collision situations are needed to be defined.

In our example collision avoidance strategy, two different collision situations, the frontal and the side collision are distinguished. For the obstacle sensor configuration, having the precondition of motionless obstacles, it is sufficient to have three ultrasonic distance sensors at the front of the AGV, one in the middle (U_M) and one-one on both sides (U_L, U_R) (see fig. 2)) to approximate both the collision conditions [20]. Since in case of motionless obstacles, the obstacle distance measurements of the near past can be used for scanning the boundaries of the obstacles. Collecting the previous measurements of the left and right obstacle sensors and the corresponding positions of the AGV (measured by the motion sensors on the wheels), the boundaries of the obstacles can be approximated [20].

Hence for avoiding frontal collision situations the distances measured by the left middle and right ultrasonic sensors (R_L, R_M, R_R) can be applied as additional observations for the steering control, and for the side collision situations based on the approximated boundaries of the obstacles, the approximated maximal left and right turning angle without side collision (α_{ML}, α_{MR}) can be applied as additional observations. Together with the two original observations of the steering control it takes seven observations, therefore the ith rule of the rule base will have the following form:

$R_{Vd(i)}$: **If** $e_v=A_{1,i}$ **And** $\delta=A_{2,i}$ **And** $R_L=A_{3,i}$ **And** $R_R=A_{4,i}$ **And** $R_M=A_{5,i}$ **And** $\alpha_{ML}=A_{6,i}$ **And** $\alpha_{MR}=A_{7,i}$
Then $V_d=B_i$.

In this example the main goal of the *Path tracking and, collision avoidance strategy* is the path tracking (to follow a guide path) and as a sub goal, a kind of restricted (limited) collision avoidance. Here the restricted collision avoidance means: "avoiding obstacles without risking the chance of loosing the guide path".

The extension of the path tracking rule base (Table 3.) with collision avoidance can be done in the similar way, as the original rule base was designed. Having FRI, it is enough to concentrate on the relevant situations only. First we have to define the linguistic term fuzzy partitions of the new observations (namely for: R_L, R_M, R_R, α_{ML}, α_{MR}), and then by collecting the control actions of the relevant situations, setting up the new rules. For the new antecedents universes it is enough to have two fuzzy sets in the linguistic term fuzzy partitions, namely large (*L*), and small (*S*), as we are mainly interested in the existence of the collision situation (small value (*S*): collision situation, large value (*L*): no collision situation). The rest of the antecedent universes and the consequent universe may remain unchanged.

At the first step of introducing collision avoidance in the path tracking rule base, we have simply extend the original rule base (Table 3.) by simply permitting the control actions if there is no collision situation (*L*) concerned (see Table 4.). The rules we get such a way (Table 4.), has the same effect as the original path tracking rules (Table 3.), if there is no obstacle, but they are loosing their influence if a corresponding collision situation appears.

Table 4. Path tracking and collision avoidance, first step

$\mathbf{R}_{Vd}$:	e_v	δ	R_L	R_R	R_M	α_{ML}	α_{MR}	V_d
Rule 1:	*NL*						*L*	*PL*
Rule 2:	*PL*					*L*		*NL*
Rule 3:	*NM*	*Z*					*L*	*PL*
Rule 4:	*PM*	*Z*				*L*		*NL*
Rule 5:	*NM*	*PM*	*L*		*L*	*L*		*Z*
Rule 6:	*PM*	*NM*		*L*	*L*		*L*	*Z*
Rule 7:	*Z*	*PM*	*L*		*L*	*L*		*NS*
Rule 8:	*Z*	*NM*		*L*	*L*		*L*	*PS*

The next step is handling the collision situations. Keeping the structure of the original relevant path tracking rules, we have to add rules which gaining influence, when an obstacle is approaching (*S*). In our simple example, we need only four additional rules. Two rules (9, 10 of Table 5.) are required for modifying the consequence of the original path tracking rules (7, 8 of Table 5.) in collision situation, and two other rules (11, 12 of Table 5.) for handling the frontal collision situation.

Table5. Path tracking and collision avoidance, sparse rule base

R_{Vd}:	e_v	δ	R_L	R_R	R_M	α_{ML}	α_{MR}	V_d
Rule 1:	*NL*						*L*	*PL*
Rule 2:	*PL*					*L*		*NL*
Rule 3:	*NM*	*Z*					*L*	*PL*
Rule 4:	*PM*	*Z*				*L*		*NL*
Rule 5:	*NM*	*PM*	*L*		*L*	*L*		*Z*
Rule 6:	*PM*	*NM*		*L*	*L*		*L*	*Z*
Rule 7:	*Z*	*PM*	*L*		*L*	*L*		*NS*
Rule 8:	*Z*	*NM*		*L*	*L*		*L*	*PS*
Rule 9:	*Z*	*PM*	*S*		*S*			*PL*
Rule 10:	*Z*	*NM*		*S*	*S*			*NL*
Rule 11:	*Z*	*Z*	*L*	*S*	*S*			*NL*
Rule 12:	*Z*	*Z*	*S*	*L*	*S*			*PL*

Table 6. Path tracking and restricted collision avoidance

R_{Vd}:	e_v	δ	R_L	R_R	R_M	α_{ML}	α_{MR}	V_d
Rule 1:	*NL*							*PL*
Rule 2:	*PL*							*NL*
Rule 3:	*NM*	*Z*					*L*	*PL*
Rule 4:	*PM*	*Z*				*L*		*NL*
Rule 5:	*NM*	*PM*	*L*		*L*	*L*		*Z*
Rule 6:	*PM*	*NM*		*L*	*L*		*L*	*Z*
Rule 7:	*Z*	*PM*	*L*		*L*	*L*		*NS*
Rule 8:	*Z*	*NM*		*L*	*L*		*L*	*PS*
Rule 9:	*Z*	*PM*	*S*		*S*			*PL*
Rule 10:	*Z*	*NM*		*S*	*S*			*NL*
Rule 11:	*Z*	*Z*	*L*	*S*	*S*			*NL*
Rule 12:	*Z*	*Z*	*S*	*L*	*S*			*PL*

The rule base (Table 5.) we got, is slightly differs from our original goal. Its main task is the collision avoidance and the path tracking remains the sub goal only. To keep the original goal the *Path tracking and, restricted collision avoidance*, “avoiding obstacles without risking the chance of loosing the guide path”, the rule base of Table V needs a slight modification. The main rules of path tracking (1, 2 of Table 5.) needed to be kept even if an obstacle is approaching, by removing the antecedents related to obstacles (*S*). Therefore for the final sparse rule base of the *Path tracking and, restricted collision avoidance strategy* we get Table 6.

The next step is constructing the scaling functions of the new antecedent universes based on their fuzzy partitions. The last step of the steering control design is the fine tuning of the whole system by parameter optimizing the antecedent and consequent scaling functions by a gradient free optimization method. In our case a simple hill climbing method was applied to optimize the docking distance of the AGV in case of a given obstacle configuration. Fig. 9 introduces two simulated run of the optimized steering control, one for the obstacle free, and one for the obstacle disturbed situation.

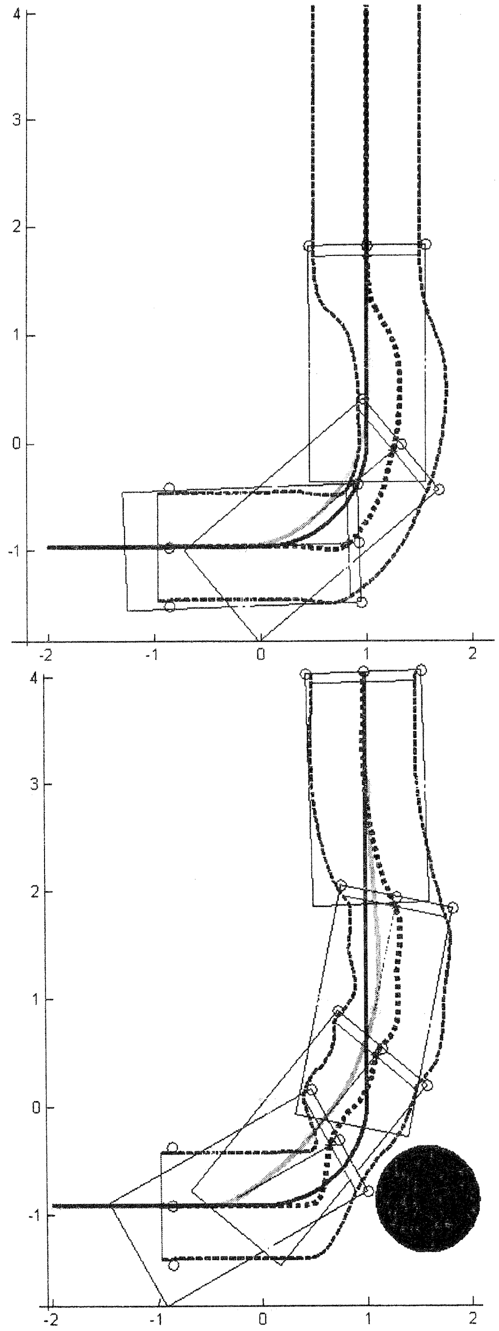


Fig.9. Track of obstacle free and obstacle disturbed run of the optimized FIVE FRI AGV steering control.

5. Conclusion

The main goal of this paper was to give a short survey on various FRI methods together with a simple demonstration of their benefits in fuzzy rule base construction. In more detail, FRI method "FIVE" was introduced in the paper, as a simple and quick FRI method. Applying FRI methods, and hence sparse rule bases can dramatically simplify fuzzy rule base creation. FRI methods can save the expert from dealing with derivable rules and therefore help to reduce the number of the fuzzy rules needed to be handled considerably. In the example "*Path tracking and, restricted collision avoidance strategy*" introduced in this paper, the steering control sparse fuzzy rule base was build upon 12 rules only. In case of classical FLC e.g. max-min CRI, and complete rule base, having the same antecedent fuzzy partitions, the steering control should contain $5^2 + 2^5 = 57$ rules.

More details of the automated guided vehicle (AGV) navigation control example together with some docking distance performance comparison of the classical CRI and FRI solutions in the obstacle free case can be found in [19] and its extension with obstacle avoidance capability (FRI solution only) in [20].

The main benefit of the applied FRI method ("FIVE") compared to the other FRI methods in case of embedded control applications is the relatively low computational demand. Adopting the idea of the vague environment (VE) [18], "FIVE" can handle the antecedent and consequent fuzzy partitions of the fuzzy rule base by scaling functions [18] and therefore turn the fuzzy interpolation to crisp interpolation. Hence there is no need for fuzzification of the observation and defuzzification of the conclusion. Moreover the required scaled distances can be pre-calculated and stored in advance. More details of the FRI method "FIVE" can be found in [16], [17], [18] and its extension with the ability of handling fuzzy observation in [21].

The code of the example application of this paper together with other FRI applications, and a freely available FRI Toolbox (a collection of Matlab functions implementing FRI techniques, under GNU General Public License) can be downloaded from [31] and [32].

Acknowledgement

This research was partly supported by the Intelligent Integrated Systems Japanese Hungarian Laboratory and the National Scientific Research Fund Grant OTKA T049499.

References

[1] Baranyi, P., Kóczy, L. T., Gedeon, T. D.: *A Generalized Concept for Fuzzy Rule Interpolation*, IEEE Trans. on Fuzzy Systems, vol. 12, No. 6, (2004) pp 820-837

[2] Jenei, S.: *Interpolating and extrapolating fuzzy quantities revisited – an axiomatic approach*, Soft Comput., vol. 5., (2001) pp. 179-193

[3] Jenei, S., Klement, E. P., Konzel, R.: *Interpolation and extrapolation of fuzzy quantities – The multiple-dimensional case*, Soft Comput., vol. 6 (2002) pp. 258-270

[4] Johanyák, Z. C., Kovács, S.: *Survey on various interpolation based fuzzy reasoning methods*, Production Systems and Information Engineering Volume 3 (2006) pp. 39-56

[5] Johanyák, Z. C., Kovács, S.: *Fuzzy Rule Interpolation by the Least Squares Method*, 7th International Symposium of Hungarian Researchers on Computational Intelligence (HUCI 2006), November 24-25, (2006) Budapest, pp. 495-506

[6] Johanyák, Z. C., Kovács, S.: *Fuzzy Rule Interpolation Based on Polar Cuts*, Computational Intelligence, Theory and Applications, Springer Berlin Heidelberg, (2006) pp. 499-511

[7] Johanyák, Z. C., Kovács, S.: *Vague Environment-based Two-step Fuzzy Rule Interpolation Method*, 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics (SAMI 2007), January 25-26, Poprad, Slovakia, (2007) pp. 189-200

[8] Johanyák, Z. C., Tikk, D., Kovács, S., Wong, K.K.: *Fuzzy Rule Interpolation Matlab Toolbox - FRI Toolbox*, Proc. of the IEEE World Congress on Computational Intelligence (WCCI'06), 15th Int. Conf. on Fuzzy Systems (FUZZ-IEEE'06), Vancouver, Canada (2006) pp. 1427-1433

[9] Johanyák, Z. C., Parthiban, R., Sekaran, G.: *Fuzzy Modeling for an Anaerobic Tapered Fluidized Bed Reactor*, SCIENTIFIC BULLETIN of "Politehnica" University of Timisoara, ROMANIA, Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE, ISSN 1224-600X, Vol:52(66) No:2, (2007) pp. 67-72

[10] Johanyák, Z. C., Szabó, A.: *Tool life modelling using RBE-DSS method and LESFRI inference mechanism*, A GAMF Közleményei, Kecskemét, XXII. (2008) pp. 5-16

[11] Klawonn, F.: *Fuzzy Sets and Vague Environments*, Fuzzy Sets and Systems, 66 (1994) pp. 207-221

[12] Kóczy, L. T., Hirota, K.: *Rule interpolation by α-level sets in fuzzy approximate reasoning*, In J. BUSEFAL, Automne, URA-CNRS. Vol. 46. Toulouse, France, (1991) pp. 115-123

[13] Gedeon, T. D., Kóczy, L. T.: *Conservation of fuzziness in rule interpolation*, Intelligent Technologies, vol. 1, International Symposium on New Trends in Control of Large Scale Systems, Herlany (1996) pp. 13-19

[14] Kóczy, L. T., Kovács, S.: *On the preservation of the convexity and piecewise linearity in linear fuzzy rule interpolation*, Tokyo Inst. Technol., Yokohama, Japan, Tech. Rep. TR 93-94/402, LIFE Chair Fuzzy Theory, (1993)

[15] Kóczy, L. T., Kovács, S.: *Shape of the Fuzzy Conclusion Generated by Linear Interpolation in Trapezoidal Fuzzy Rule Bases,* in Proceedings of the 2nd European Congress on Intelligent Techniques and Soft Computing, Aachen (1994) pp. 1666–1670

[16] Kovács, S.: *New Aspects of Interpolative Reasoning*, Proceedings of the 6th. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain (1996) pp. 477-482

[17] Kovács, S., Kóczy, L. T.: *Approximate Fuzzy Reasoning Based on Interpolation in the Vague Environment of the Fuzzy Rule base as a Practical Alternative of the Classical CRI*, Proceedings of the 7th International Fuzzy Systems Association World Congress, Prague, Czech Republic, (1997) 144-149

[18] Kovács, S., Kóczy, L. T.: *The use of the concept of vague environment in approximate fuzzy reasoning*, Fuzzy Set Theory and Applications, Tatra Mountains Mathematical Publications, Mathematical Institute Slovak Academy of Sciences, Bratislava, Slovak Republic, vol.12, (1997) pp. 169-181

[19] Kovács, S., Kóczy, L. T.: *Application of the Approximate Fuzzy Reasoning Based on Interpolation in the Vague Environment of the Fuzzy Rulebase in the Fuzzy Logic Controlled Path Tracking Strategy of Differential Steered AGVs*, Lecture Notes in Computer Science, 1226, Springer, Germany, (1997) pp.456-467

[20] Kovács, S., Kóczy, L. T.: *Application of an approximate fuzzy logic controller in an AGV steering system, path tracking and collision avoidance strategy*, Fuzzy Set Theory and Applications, In Tatra Mountains Mathematical Publications, Mathematical Institute Slovak Academy of Sciences, vol.16, Bratislava, Slovakia, (1999) pp. 456-467

[21] Kovács, S.: *Extending the Fuzzy Rule Interpolation "FIVE" by Fuzzy Observation*, Advances in Soft Computing, Computational Intelligence, Theory and Applications, Bernd Reusch (Ed.), Springer Germany, ISBN 3-540-34780-1, (2006) pp. 485-497

[22] Shepard, D.: *A two dimensional interpolation function for irregularly spaced data*, Proc. 23rd ACM Internat. Conf., (1968) pp. 517-524

[23] Tikk, D.: *Notes on the approximation rate of fuzzy KH interpolator*, Fuzzy Sets and Systems, 138(2), Sept., (2003) pp. 441-453

[24] Tikk, D., Baranyi, P.: *Comprehensive analysis of a new fuzzy rule interpolation method*, In IEEE Trans. Fuzzy Syst., vol. 8, No. 3, June, (2000) pp. 281-296

[25] Tikk, D., Baranyi, P., Gedeon, T. D., Muresan, L.: *Generalization of a rule interpolation method resulting always in acceptable conclusion*, Tatra Mountains Math. Publ., 21, (2001) pp. 73–91

[26] Tikk, D., Joó, I., Kóczy, L. T., Várlaki, P., Moser, B., Gedeon T. D.: *Stability of interpolative fuzzy KH-controllers*, Fuzzy Sets and Systems, 125 (1) (2002) pp. 105–119

[27] Vass, F., Kalmár, L., Kóczy, L. T.: *Extension of the fuzzy rule interpolation method*, in Proc. Int. Conf. Fuzzy Sets Theory Applications (FSTA'92), Liptovsky M., Czechoslovakia, (1992) pp. 1-6

[28] Wong, K. W., Gedeon, T. D., Tikk, D.: *An improved multidimensional α-cut based fuzzy interpolation technique*, In Proc. Int. Conf Artificial Intelligence in Science and Technology (AISAT'2000), Hobart, Australia, (2000) pp. 29–32

[29] Wong, K. W., Tikk, D., Gedeon, T. D., Kóczy, L. T.: *Fuzzy Rule Interpolation for Multidimensional Input Spaces With Applications*, IEEE Transactions on Fuzzy Systems, ISSN 1063-6706, Vol. 13, No. 6, December, (2005) pp. 809-819

[30] Yam, Y., Kóczy. L. T.: *Representing membership functions as points in high dimensional spaces for fuzzy interpolation and extrapolation*, Dept. Mech. Automat. Eng., Chinese Univ. Hong Kong, Tech. Rep.CUHK-MAE-97-03, (1997)

[31] The FRI Toolbox is available at: http://fri.gamf.hu

[32] Some FRI applications are available at: www.iit.uni-miskolc.hu/~szkovacs