A Proposal for Multi-Purpose Fuzzy Cognitive Maps Library for Complex System Modeling

Michal Puheim[1], Ján Vaščák[2], Ladislav Madarász[3]
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Letná 9, 042 00 Košice, Slovak Republic
[1]michal.puheim@tuke.sk, [2]jan.vascak@tuke.sk, [3]ladislav.madarasz@tuke.sk

***Abstract*—In this paper we explain the need for a new multi-purpose Fuzzy Cognitive Maps (FCM) library and offer its proposal. Within the paper we provide an overview of available tools and libraries along with their advantages and disadvantages. We use this information in order to build a list of necessary requirements for a new library. With these requirements in mind, we offer a proposal for a new FCM library. The proposed library aims provide flexibility and applicability for various use cases, from simple system prototyping to modeling and control of complex systems. In contrast to the most of the available FCM tools, it incorporates a built-in learning mechanisms for automated adaptation of the parameters of the map. An emphasis is placed on a simplicity of library deployment and ease of use.**

Keywords—fuzzy cognitive map, library, .NET framework, complex systems, modeling.

I. INTRODUCTION

The Fuzzy Cognitive Maps (FCM) [1] are considered a viable option for modeling various real-world systems. Its applications cover a wide range of diverse areas such as political and social studies, information technology, robotics, expert systems, engineering, medicine, education, etc [2]. In order to use the FCM for system modeling, we have two basic options. The first option is to implement a new dedicated program, which would do exactly the same thing as we want, e.g. modeling, prediction, control, etc. The second possibility is to rely on available tools and libraries. However, these tools may not fulfill our expectations in terms of functionality, scalability, performance etc. Only a few of them support automated adaptation of map parameters, most of them rely on expert knowledge. Therefore, we believe, that it is appropriate to start a discussion about a new FCM library, which would serve multiple purposes and also offer built-in automated learning methods.

The rest of the paper is organized as follows: In the second section, we present an overview of tools available for FCM modeling and provide a categorization of these tools according to their applicability. In the third section we discuss the advantages and disadvantages of the available tools and explain our motivation for a proposal of a new FCM library. In the fourth section we list the necessary requirements for the new library and in the fifth section we outline its system proposal. In the sixth section we present arguments for a selected implementation platform, which is the .NET framework. In the conclusion we summarize the features of the proposed library and give a list of its possible applications.

II. OVERVIEW OF AVAILABLE FCM TOOLS

There is a limited set of tools and libraries which can be used for analysis and system modeling by means of the FCMs. The most notable examples of these tools are:

- *FCMapper* [2],
- *Fuzzy Cognitive Maps Applet* [4],
- *Java Fuzzy Cognitive Maps* library [5],
- *Mental Modeler* [6],
- various libraries and toolboxes for Matlab/Simulink (e.g. *FCM Tool* [7] or *Fuzzy Logic Toolbox* [8]).

The *FCMapper* is a tool developed by a community of scientists and engineers based at the *fcmappers.net* website [2], who are interested in fuzzy mapping and its applications to network analysis and system modeling [1]. The tool is based on the *MS Office Excel* spreadsheet processor and the *Visual Basic for Applications* (VBA) framework. It does not support the visualization of the map, but created models can be exported in the net-file format and visualized in freely available external visualization programs, such as Pajek [9] or Visone [10].

The *Fuzzy Cognitive Maps Applet* [4] is a Java applet, which can be executed as an embedded application within an internet webpage. In order to run the application, the *Java Runtime Environment* (JRE) or an appropriate web-browser plugin is required. As a typical advantage of all the applets, there is no need for installation (as long as the JRE and a web-browser are installed). However, a great disadvantage (of all the applets) is the inability to save created models. Therefore it is mostly suited for educational purposes or simple FCM prototyping.

The *Mental Modeler* (Fig. 1) is FCM modeling software whose main advantage is the simplicity of use. It contains accessible graphical user interface which allows easy and intuitive building of FCMs even for larger groups of users who can collaboratively represent and test their assumptions about a modeled system. The software can be used in two ways, as a desktop application and as an online tool [6]. It is expert oriented and does not provide automated learning options.

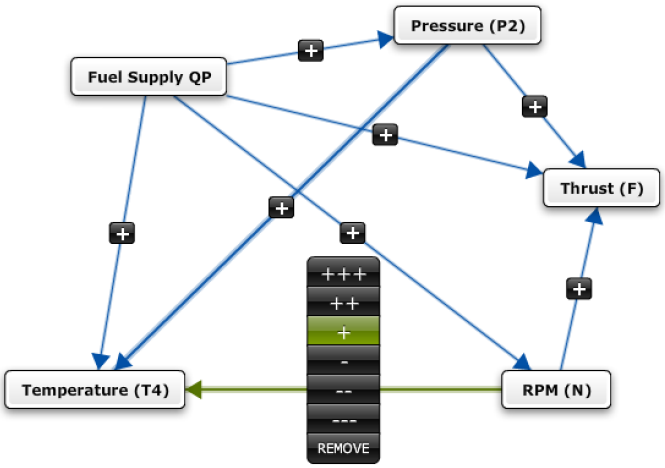

Fig. 1. Example of simple FCM model of a turbojet engine created with the Mental Modeler Concept Mapping tool.

The *Java Fuzzy Cognitive Maps* library (JFCM) [5] is another viable option for FCM modeling. It is a simple open source library written in Java, which provides classes and methods that can be used in another Java programs. It is developed with an object-oriented approach in mind, so it can be easily extended if the standard built-in components are not sufficient. It still needs the JRE in order to run, but no web-browser is necessary.

The *FCM Tool* [7] is an embedded application for *Matlab*. It is executed from the Matlab command line and runs in a separate window. It enables users to create FCMs by adding concepts and defining relations in the form of matrix. Unlike the Mental Modeler, it does not contain a graphical representation of the map. However, a change of concept values during simulation can be visualized on a 2D plot. The application does not contain any tools for automatic adaptation of FCM relations, therefore an expert knowledge is required.

The *Fuzzy Logic Toolbox* provides an additional functionality for *Matlab* by including functions, applications and a *Simulink* block for analyzing, designing, and simulating systems based on fuzzy logic [8]. The toolbox models behaviors of complex system with use of simple logic rules and implements these rules within a fuzzy inference system [8]. Functions are provided for many common methods, including fuzzy clustering and adaptive neuro-fuzzy learning [8]. Applications include the Fuzzy Logic Designer and the Neuro-Fuzzy Designer, both of which support the creation, training, and testing of fuzzy inference systems in a convenient graphical user interface [8]. The fuzzy inference blocks in Simulink are similarly the viable alternative to a creation of comprehensive models of various dynamic systems [8]. There are two basic types of fuzzy blocks. The first basic type is the classic fuzzy logic controller block. A set of various membership function (MF) blocks belongs to the second basic type. The latter can be conveniently used to model FCMs if the weights and relations between concepts are known (e.g. using expert knowledge), see Fig. 2. However, the problem is that the MF blocks do not provide any mechanism for automatic adaptation or learning, hence its application for a complex system modeling is questionable.

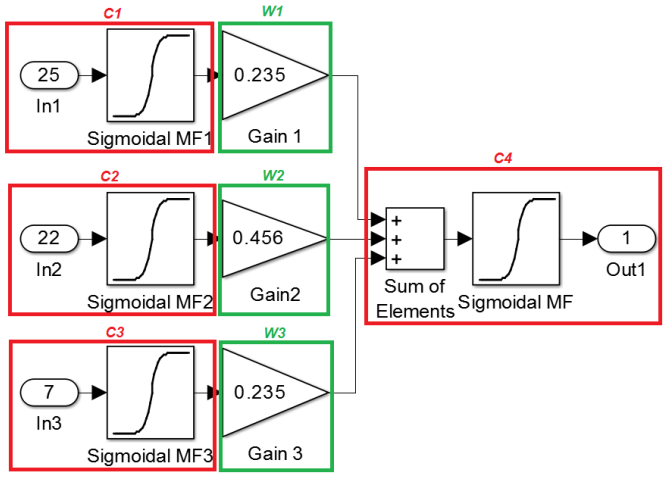

Fig. 2. An example of a Simulink model of a simple FCM with three concepts connected to the fourth concept where weights are represented as gains. Values from the real world (inputs) are fuzzified using the sigmoidal MF. The sigmoidal function is also used as a threshold function for an activation of the fourth concept.

The introduced tools can be listed to the following categories according to their utilization potential:

- *Simple single-purpose applications*, such as the Fuzzy Cognitive Maps Applet, whose usage is often limited to very basic tasks, such as simple prototyping or demonstration and explanation of basic principles of modeling with use of FCMs (education). Advanced functions, such as import or export of created models are not possible.
- *Limited multi-purpose applications*, which are similar to the FCMapper and enable wider utilization range, possibility to export created models and its use within other programs. This class of application is often embedded in superior program (e.g. spreadsheet processor). An advantage of this condition is the simplicity of use. However, the applicability of these applications is limited and its functionality is hardly extended.
- *Extendable multi-purpose applications*, such as the Fuzzy Logic Toolbox or the FCM Tool for Matlab. This group of applications provides the widest range of options and possibilities. Such applications can be extended and interconnected with other applications and frameworks, which allows for example direct use of created FCM models for diagnostics and control of real world dynamic systems.
- *Multi-purpose libraries*, such as the JFCM, whose design is deliberately oriented to cover as large set of possible applications as possible. While these libraries usually do not include the user interfaces, they provide a large set of simply deployable functions which cover the fundamental background logic. These functions can be used to rapidly prototype any kind of needed application.

III. Motivation for New Library

While the previous statements may give an impression that there are far enough tools for FCM modeling, from simple applets to complete applications and libraries, and that there is no need for other solutions, the opposite is the case. Despite JFCM is being considered a competent and efficient library, it does not provide *support for advanced cognitive map designs* such as nonlinear, dynamic relations, conditional relations or relations represented by neural networks [11]. However, such advanced designs are important for modeling relations between elements of complex systems, since it is difficult to model nonlinear systems with linear models (such as FCM) only [2].

Additionally, if we summarize the problems and shortcomings of the already mentioned tools, which are *implementation* related, we get the following list of general issue categories (along with respective affected tools and dependencies):

- *Restrictions of embedded applications* (MS Office spreadsheets, Java applets, etc.), which obstruct capabilities of embedded scripts or applications due to inherent deficiencies of a parent program (spreadsheet processor, internet browser). This often affects basic necessary user activities, such as input/output operations (e.g. saving of FCM models) or problems with FCM visualization. Even though the applications which belong to this category are comprehensive and accessible even to inexperienced users, their usability for more advanced tasks is limited due to the lack of flexibility within the parent program. This often leads to a point where such applications slowly become unmaintained and consequently obsolete and inapplicable.
- *Dependency on an external proprietary software* (MS Office, Matlab, etc.), which limits the accessibility and deployment options of related tools (and subsequently created models and/or derived control algorithms). These limitations do not directly restrict the flexibility of application usage or its implementation. The restrictions are merely economical, which may not be of an issue in most cases. However with growing number of required licences (e.g. in larger education facilities, or in distribution for general public), the financial element becomes relevant, especially with expensive programs (such as Matlab).
- *Dependency on external frameworks* (Java Runtime Environment, Matlab Runtime Compiler, Python, etc.), which partially hinder the deployment of affected tools. The installation and execution of such applications is delayed by requirements for necessary third party components, which (though being often well accessible and freely available) are not well integrated within the given operating system (platform or device).

However, the most relevant motivation for a new library is definitely the fact that almost all of the available tools rely heavily on expert knowledge and only a few of them do support *automated learning methods*. The missing option for learning automation is a major obstacle, especially for modeling complex systems.

IV. Library Requirements

If we consider all of the pros and cons of the FCM tools mentioned in the section II and summarize their most relevant attributes, we can generate a list of basic requirements for a new FCM library, which should include:

- *Map design*, which includes definition of concepts and eventually also connections between the concepts.
- *Membership function setup*, which defines a fuzzification (or normalization) of real world values for each concept. In other words the library needs to be able to specify transformation from real world values to concept activations and vice versa.
- *Definition of casual relations between concepts*, including support for advanced approaches, such as the nonlinear, conditional and dynamic relations [12], possibly utilizing neural networks [11].
- *Concept initialization* and manual *setting of concept activation values* (in case of use of a control or sensor nodes [13]).
- *Computation of FCM updates* during a runtime. The updated activation values of map concepts need to be computed in a reasonable time. Therefore it is beneficial to support parallelized and scalable algorithms.
- *Automated adjustment of map parameters* including definition of concepts, membership functions, relations etc. The adjustments may be performed using either supervised or unsupervised learning methods. Support for basic learning methods should be included within the library. Other specialized methods should be easily implemented via library application interface (API).
- *Model import (export)* from (to) a file, preferably in the XML format compatible with other FCM libraries and applications.

Additionally to these basic requirements, the library should also aim to achieve the following goals:

- *Native support* for most used desktop and server oriented operating systems intended for productive work.
- *Independence* from external runtime frameworks (JRE, Python, MRC).
- *Easy integration* with Matlab/Simulink (and/or other front-end software suites) as an external shared library.
- *Open source* distribution to enable simple updates of the library depending on the additional requirements of the application or the user.

The library does not necessarily have to include any graphical user interface support or additional tools for visualization. Its main purpose is to be a container for functions and data structures providing backend logic necessary to create programs which could perform FCM modeling. However, it may be beneficial if the appropriate frontend module is provided along with the library.

V. LIBRARY PROPOSAL

Our proposal for the new multi-purpose FCM library is based on the requirement presumptions from the previous section. A system diagram of the proposed library is shown in Fig. 3. The base class of the library is the FCM class. The attributes of a new FCM object can be set by an expert using the methods from the design module. This module includes all the methods which are necessary for a FCM creation including a concept definition, a membership function definition and a relation definition and configuration. Therefore it is possible to create a map entirely with use of an expert knowledge. However, the library also contains a learning module, which can set these parameters automatically. The amount of parameters which can be determined automatically depends upon the used learning method. Unsupervised learning methods can only adjust relations between concepts. With supervised methods it is also possible to define concepts and suitable corresponding membership functions using the information from the training data. The expert oriented and automated approaches for the adaptation of parameters of the map are further discussed in the following subsections.

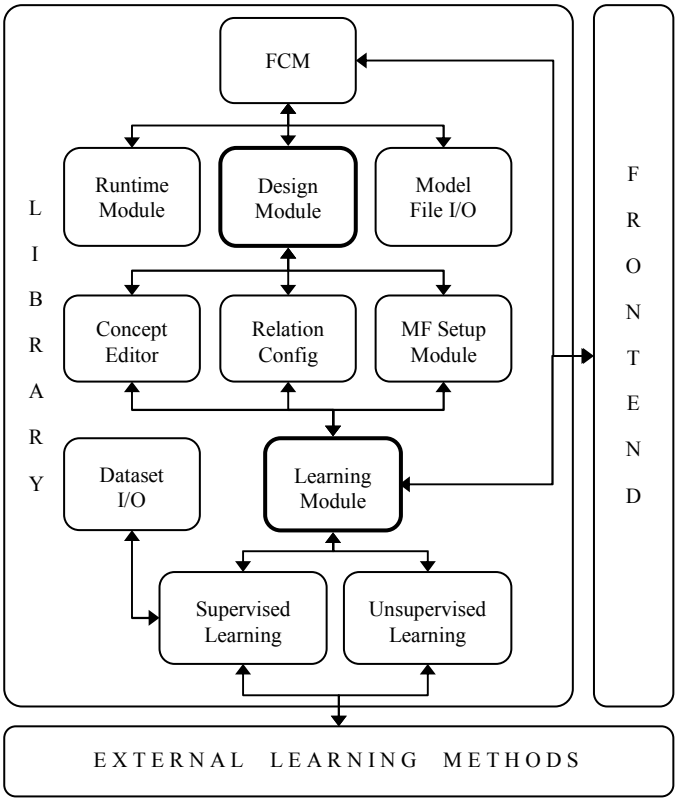

Fig. 3. A system diagram of the proposed FCM library.

A. Design Module Description

The first step in the FCM design is the definition of the concepts. Therefore, the library needs to support the method for addition and removal of the concepts. Furthermore, the methods which provide information of already defined concepts (such as names, units of measurement, etc.) are necessary.

The next logical step is the definition of membership functions (MF) for each concept. The definition of MF should support a variety of linear and nonlinear normalization functions, such as rescaling (min-max linear normalization), standardization or logistic normalization.

Final stage in the FCM design is the definition of concept relations. The relation definition consists of two steps. The first one is the declaration of the relation. Thus a method which creates a connection between two concepts is needed. The second step is the identification of the relation (setting its value/s). If it is set by an expert, it can be defined as:

- *simple linear weight* with singleton value,
- *nonlinear analytic function* (useful for modeling physical phenomena where relation are well known),
- *Three-Term Relation* (TTR) [11] with three linear weights, one for proportional, derivative and average component,
- *time-window relation* [11] with number of weights equivalent to the size of the time window.

Additionally, the relations can be represented as *small multilayer feed-forward neural networks* [11]. However, in this case it is necessary to use supervised learning methods, which are further described in the next section.

B. Automated Learning Module Desctiption

A complementary approach to the FCM design, which lessens the demand on expert knowledge, is to use automated learning methods. These methods can be classified into two basic groups – supervised and unsupervised. The chosen method depends on the intended use of the FCM. In general, the FCM can be used to solve the following elementary problems [14][15]:

- *Regression problem* [14], in which we use the FCM as a model of dynamic system. We are looking for such relation parameters, for which the FCM model most closely matches the real system.
- *Attractor search problem* [15], in which we are looking for such relation parameters, which force the FCM model to converge into a desired end point or a limit cycle. The initial activation values of concepts may vary, only the final values are specified. In this way the FCM can be used as a controller.

Unsupervised learning methods can be applied mostly to the *attractor search problem*. Among these we can count the family of Hebbian based learning methods, such as [15]:

- Differential Hebbian Learning,
- Active Hebbian Learning,
- Nonlinear Hebbian Learning.

If we can define an objective (fitness) function of the adjusted parameters, evolutionary or population-based optimization algorithms can be used:

- Evolutionary algorithms,
- Particle Swarm Optimization,
- Simulated annealing,
- Tabu search.

These optimization algorithms can be used also for the *regression problem*. Moreover, in order to solve this problem, the supervised learning algorithms from the area of neural network learning can be applied [14][15]. These are mostly the Least Mean Square (LMS) based methods, such as:

- Basic Delta Rule,
- Backpropagation of Error,
- Backpropagation through Time.

Additionally, the Data Driven Nonlinear Hebbian Learning method (DD-NHL) [16] can be used for supervised learning of FCMs [14].

As for the implementation of these methods within the library, it is not necessary to include all of them. It should be sufficient to implement single unsupervised Hebbian method and single supervised LMS method, coupled with a selected evolutionary or population based algorithm or DD-NHL respectively. However, it would be beneficial if the library includes an interface which would enable simple addition of external learning methods. Such an interface would need to expose adjustable parameters of the FCM to the external method. Additionally, it would need to supply the training data in case of the supervised learning.

If we consider the format of the training data within the dataset file, it should be in form of measurements of variables of a real world system in operation. It should contain values of given number of variables from time 0 to *n* during a single or multiple measurements.

VI. Implementation Remarks

We evaluated several aspects during the selection of the suitable platform for the implementation of the library. Along with the objectives outlined in the section IV, we considered the following:

- *avoidance of platforms* which are already *well covered* with available FCM tools,
- *possible spread* of the library to as large user-base as possible,
- *simplicity of development* and implementation,
- *simplicity of deployment*, installation and use.

For the first point, our goal was to avoid platforms, which are already covered with available tools. Particularly we referred to Java with a very competent JFCM library [5]. Furthermore, we wanted to aim mostly for desktop operating systems which are used for productive work. In most cases this applies to the MS Windows operating system. We also considered the development speed and application deployment obstructions. As a result, we decided to implement the library using the .NET framework, which is well integrated in modern versions of Windows. Additionally, parts of the framework were recently released as open source, which should ensure its accessibility even on other platforms (e.g. with the Mono implementation). And finally, the newly available .NET Native compilation option enables generation of programs and libraries which can be run natively, without the need for a presence of the framework within the system.

We aim to build the proposed library as a shared dynamic library (DLL), which can be easily incorporated into programs in various languages and environments. e.g. Python, Matlab, Simulink, etc. There is also an obvious possibility to create a frontend application for the library with support of Windows Forms or Windows Presentation Foundation.

VII. Conclusion

In this paper we proposed a new library intended for modeling using the FCMs. We proposed the library with a goal of enabling fast and simple prototyping of system models. The library is not limited only to the basic FCMs. It can also apply derived methods, such as recently proposed Three-Term Relation Neuro-Fuzzy Cognitive Maps (TTR-NFCM) [11]. The proposed library also supports various methods for relation expression (nonlinear relations, dynamic, relations, function relations, neural relations, etc.). It employs built in automatic unsupervised or supervised data-driven learning algorithms to adjust the parameters of the map.

We proposed such an implementation of the library, which is native to the MS Windows platform and which is not dependent on external runtime frameworks (JRE, Python etc.). However, it can be easily used and integrated within various frontend applications and external programs. It can also provide backend computations in the Matlab/Simulink environment.

The possible application of the library is in the areas of *fast model prototyping, complex system modeling, control, diagnostics, prediction and prognostics*. It can be used to solve both elementary FCM problems, such as the *attractor search problem* applicable in system control, and the *regression search* of system models. It can be also used within *situational control* to create the classifiers of situation classes.

Acknowledgement

This paper is supported by VEGA, Grant Agency of Ministry of Education and Academy Science of Slovak Republic under Grant No. 1/0298/12 – "Digital control of complex systems with two degrees of freedom." It is also supported by KEGA under Grant No. 018TUKE-4/2012 – "Progressive methods of education in the area of control and modeling of complex systems object oriented on aircraft turbo-compressor engines."

References

[1] B. Kosko, "Fuzzy Cognitive Maps," International Journal on Man-Machine Studies, vol. 24, no. 1, pp 65-75, 1986.

[2] E. I. Papageorgiou, J. L. Salmeron, "A Review of Fuzzy Cognitive Maps research during the last decade," in I*EEE Transactions on Fuzzy Systems*, vol.21, no.1, pp.66,79, Feb. 2013. ISSN 1063-6706.

[3] FCMappers. (2011, February 8). *FCMappers.net website*. [Online]. Available: http://www.fcmappers.net/ (Cited: 2014, September 20)

[4] G. O. de Aspuru. (2006). *Fuzzy Cognitive Maps*. [Online]. Available: http://www.ochoadeaspuru.com/fuzcogmap/index.php (Cited: 2013, September 20)

[5] D. de Franciscis, "JFCM : A Java Library for Fuzzy Cognitive Maps," in *Fuzzy Cognitive Maps for Applied Sciences and Engineering* (Intelligent Systems Reference Library, Vol. 54), Springer Berlin Heidelberg, 2014, pp. 199-220, ISBN 978-3-642-39738-7.

[6] S. A. Gray, S. Gray, L. Cox, S. Henly-Shepard, “Mental modeler: A fuzzy-logic cognitive mapping modeling tool for adaptive environmental management,” in *Proceedings of the 46th International Conference on Complex Systems*, Hawaii, 2013, pp. 965-973, DOI 10.1109/ HICSS.2013.399

[7] M. León, C. Rodriguez, M. M. García, R. Bello, K. Vanhoof, “Fuzzy Cognitive Maps for Modeling Complex Systems,” in *9th Mexican International Conference on Artificial Intelligence*, MICAI. Pachuca, Mexico, 2010, pp 166-174 ISBN 978-3-642-16760-7

[8] MathWorks. (2014). *Fuzzy Logic Toolbox*. [Online]. Available: http://www.mathworks.com/products/fuzzy-logic/ (Cited: 2014, September 20)

[9] V. Batagelj, A. Mrvar. (2014). *Pajek Wiki*. [Online]. Available: http://pajek.imfm.si/doku.php (Cited: 2014, September 20)

[10] Visone community. (2014). *Visone website*. [Online]. Available: http://visone.info/ (Cited: 2014, September 20)

[11] M. Puheim, J. Vaščák, L. Madarász, “Three-Term Relation Neuro-Fuzzy Cognitive Maps,” in *15th IEEE International Symposium on Computational Intelligence and Informatics: Proceedings*, Budapest, Hungary, 2014, pp. 477-482. ISBN 978-1-4799-5338-7.

[12] M. Hagiwara, "Extended fuzzy cognitive maps," in *IEEE International Conference on Fuzzy Sysems*, 1992, San Diego, CA, pp.795-801, ISBN 0-7803-0236-2.

[13] J. Vaščák, L. Madarász, "Adaptation of Fuzzy Cognitive Maps – a Comparison Study," in *Acta Polytechnica Hungarica*, Vol. 7, No. 3, pp.109-122, 2010. ISSN 1785-8860.

[14] M. Gregor, P. P. Groumpos, "Training Fuzzy Cognitive Maps Using Gradient-Based Supervised Learning", in *IFIP Advances in Information and Communication Technology*, Volume 412, pp 547-556, Jan. 2013. ISSN 1868-4238.

[15] M. Gregor, P. P. Groumpos, "Tuning the Position of a Fuzzy Cognitive Map Attractor Using Backpropagation through Time", in *The 7th International Conference on Integrated Modeling and Analysis in Applied Control and Automation*, Athens, Greece, 25-27 Sep. 2013.

[16] W. Stach, L. Kurgan, W. Pedrycz, "Data-driven Nonlinear Hebbian Learning method for Fuzzy Cognitive Maps," in *IEEE International Conference on Fuzzy Systems (FUZZ), IEEE World Congress on Computational Intelligence*. pp.1975-1981, 1-6 June 2008. ISBN 978-1-4244-1818-3