Graphical User Interface for OpenFCM Library

[1]Filip PARAČKA, [2]Michal PUHEIM

[1,2,3] Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovak Republic

[1]filip.paracka@student.tuke.sk, [2]michal.puheim@tuke.sk

Abstract — **In this paper we introduce a new software tool with graphical user interface (GUI) for creating fuzzy cognitive maps (FCMs). This tool aims to connect new library called OpenFCM, used for calculations of FCMs, with its corresponding GUI using another library MSAGL (Microsoft Automatic Graph Layout). Fuzzy cognitive maps are understood as graphical representation of the knowledge or the perception of a given system. Application was created in Visual Studio environment using C# programming language. On created application were made several experiments to test and verify functions of graphical interface and whole application.**

Keywords — **fuzzy cognitive map, graphical user interface, simulation, Visual Studio, Windows Forms**

I. Introduction

The Fuzzy cognitive maps (FCM) [1] are convenient modeling tool, usually categorized as neuro-fuzzy method for modeling and simulation of dynamic systems. One of its main advantages is the ability to incorporate knowledge of people and adapt them to given area. FCM is built by combining existing experience and knowledge regarding system. FCM is used in a wide range of areas that must deal with the creation and usage of models of uncertainty and complex processes and systems.

The main reason for creating new application is existence of new library OpenFCM. However this library does not come with corresponding graphical user interface (GUI). Its main purpose is to provide storage for functions and data structures that provide the logic necessary to create a program that will perform modeling of FCMs. The new interface should help users of this library to easily access its functions with proposed GUI.

II. Analysis of Existing Software

There are few existing tools and libraries for creating and modeling FCMs. We will describe four of them:

- Mental modeler
- FCM Wizard
- FCMapper
- JFCM

A. Mental Modeler

Mental Modeler [2] is modeling software that helps individuals and communities to capture their knowledge in a standardized format that can be used to analyze scenarios. The program was designed to offer a flexible way to transfer disparate, loosely coupled and largely qualitative knowledge about the surrounding problems into a manageable format, which can be validated by the generation of empirical evidence. Mental modeler is comprised of three main user interfaces:

- The concept mapping interface
- The matrix interface
- The scenario interface

B. FCM Wizard

FCM Wizard [3] is a program that is used for calculation, design, learning and simulation of systems based on FCMs. The program includes 20,000 lines of source code written in Java, which are spread over 115 source files. These archives are organized in four global packages (Map, Algorithms, Interface, Resources) and several sub-packages (Causality, Topology, Stability, Dataset, Handlers, Graphics). The most important packages are Algorithms and Interface: the first contains methods for learning, while the other consists of graphical components.

C. FCMapper

FCMapper [4] is a tool designed for analysis and modeling of FCMs. It provides user-friendly, simple and intuitive environment for creating FCMs. The tool is created in Microsoft Excel and VBA (Visual Basic for Applications). FCMapper was designed to save time during the analysis and visualization of FCMs.

D. JFCM

JFCM (Java Fuzzy Cognitive Maps) [4] is a free library created in Java used for modeling of FCMs. The library was created during the programming exercise, but gradually became popular. It can also be used in commercial projects and is not dependent on other libraries. The library can be used to make a wide variety of cognitive networks. It is programmed using an object-oriented approach, so the program is seen as a group of objects that work together. Objects are created using classes. If the standard objects are not enough, we can easily create new ones.

From the analyzed tools, the FCM Wizard seemed to be the best. It contains many features and different algorithms. The main disadvantage is that it is not an open source tool. Although JFCM library contains only the basic functions for creating FCM, its availability enables its usage not only for researchers or companies that work with FCMs, but also for people who want to try creating their own FCMs or use this library in their projects. During the design and implementation of our tool we use a similar approach and work with open source libraries.

III. OpenFCM Library

OpenFCM [6] is the library which was selected for our GUI implementation. It is focused on providing flexible and appropriate use cases, from prototyping simple system to the modeling and control of complex systems. This library was used mainly because of its free availability and ease of use. It can be used for basic problems such as attractor search, which is applicable in the system control, and the regression search of system models. It can also be used in situational control to create the classifiers of situation classes.

The base class of the library is an FCM class. Attributes of a new FCM can be set by an expert using Design module. This module contains all methods needed to create FCM: definition of the concept, membership functions, relationships and configuration. Therefore, it is possible to create FCM using only expert knowledge. The library also includes a learning module that can set the parameters automatically. The amount of parameters which can be determined automatically depends upon the used learning method. The library is still in development and eventually all modules will be implemented. Design module is already completed and we use methods of this module in our GUI.

IV. MSAGL Library

In order to implement the proposed GUI we used the MSAGL (Microsoft Automatic Graph Layout) [5], which is a .NET library and tool for creating hierarchical graphs, their viewing and saving into bitmaps. These charts can represent data with the flow of information in a graphical format. This library can display different types of graphs: flowchart, state diagram, network analysis and phylogenetic tree which is used in Bioinformatics. MSAGL consists of three integrated components:

- Layout engine (Microsoft.MSAGL.dll) - The core layout functionality. It allows you to create graphs with thousands of nodes that are easy to understand. This component can be used directly in cases when visualization is handled by a tool other than MSAGL
- The drawing module (Microsoft.MSAGL.Drawing.dll) - The Definitions of different drawing attributes like colors, line styles, etc. It also contains definitions of a node class, an edge class, and a graph class
- Viewer control (Microsoft.MSAGL.GraphViewerGDIGraph.dll) - This module uses the

previous 2 modules to create interactive graphs. With this module, you can edit graphs by dragging the nodes, change curves of edges, handle mouse events, get the properties of edges and nodes, zoom in and out, etc.

V. Design and implementation

The program is written in programming language C # using Windows Forms. The basis for any application created using Windows Forms are forms which can contain various graphical components. The aim of the proposed application is the option to create user defined FCMs. It is therefore necessary to define the basic functions of the application:

- Add concept
- Add edge
- Delete concept
- Delete edge
- Change the description of the concept
- Change the description of the edge
- Save created FCM
- Load created FCM
- Simulation
- Calculation of real and fuzzy value of concept
- Display information about the concept
- Switch between the real value and fuzzy value of concepts
- Calculation of concept values

During the implementation of particular functions we used methods from libraries, which were described in previous sections: MSAGL and OpenFCM. MSAGL provides creation of individual components in graphical form. OpenFCM aims to link these components, so that it is possible to not only create FCMs, but also to use calculations implemented in this library. This way is connected the front-end of the application represented by library MSAGL and back-end represented by library OpenFCM.

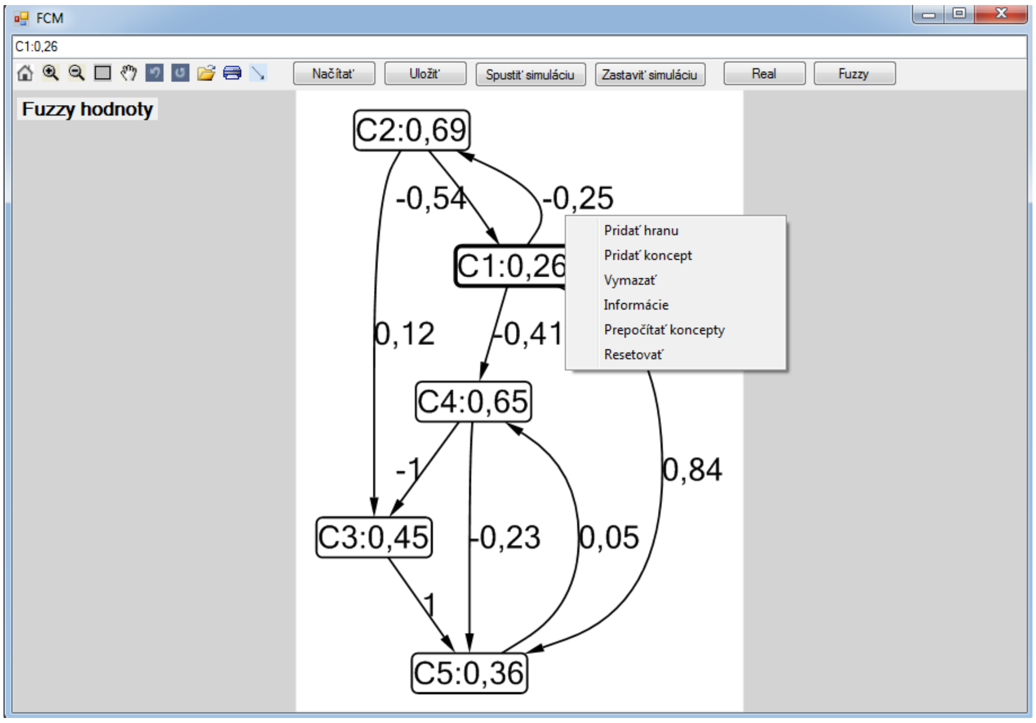


Fig. 1 Graphical user interface of application with all its functions

A. *PSO Module*

Algorithm PSO (Particle Swarm Optimization) [7] is an alternative for using expert knowledge in the creation of FCMs and is trying to automatically find FCM that best defines given system and our requirements.

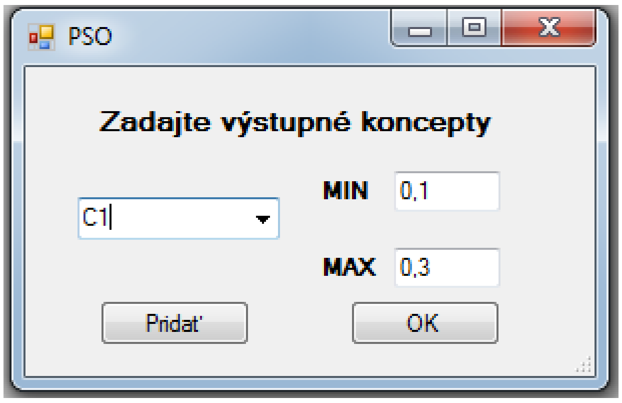
It is a stochastic optimization algorithm. It belongs to Swarm Intelligence techniques. These algorithms are inspired by social dynamics and unexpected behavior that can arise in socially organized colonies.

The PSO algorithm is based on population. It exploits a population of individuals to probe appropriate areas of the search space. In this context, the population is called swarm and the individuals are called particles. Each particle moves with an adaptable velocity within the search space and keeps in memory the best position it ever encountered. In the global variant of PSO, the best position ever achieved by all individuals of the swarm is communicated to all particles. In the local variant, each particle is assigned to a topological neighborhood that consists of a prespecified number of particles. In this case, the best position ever attained by the particles that comprise the neighborhood is communicated between them.

The proposed approach is to determine values of causal relationships between concepts, thus getting values of weights which produce the desired system behavior. The desired system behavior is characterized by the output concepts that lie within the desired intervals specified by experts. These intervals are usually dependent on each other.

There are usually many weight matrices that lead to convergence of the FCM to the desired regions of the output concepts. PSO is a stochastic algorithm and it is natural to obtain such suboptimal matrices. All these matrices are suitable for proper design of the FCM.

In our program we created a separate class which implements algorithm PSO. To the graphical interface was added a button that starts the algorithm and displays new FCM after application of PSO algorithm. Before starting this algorithm, user must define output concepts and interval he intends to limit the value of each concept.


PSO

Zadajte výstupné koncepty

C1

MIN 0,1

MAX 0,3

Fig. 2 PSO module which specifies output concepts

VI. Proposed Model

In order to test our application, we have created a simple model representing natural ecosystem. In this ecosystem living organisms and non-living components affect each other. The main components of this system are:

- Wolves
- Sheep
- Sun
- Grass
- Rain
- Humidity

We had to determine interdependencies between these concepts:

- Wolves are trying to attack the sheep and earn much-needed food for survival
- The presence of sheep and their higher numbers increase the quantity of attacking wolves
- Sheep grazing in the meadow reduce the amount of grass
- The sun promotes the growth of grass
- Solar radiation reduces humidity
- Rain increases the humidity
- The amount of grass reduces humidity

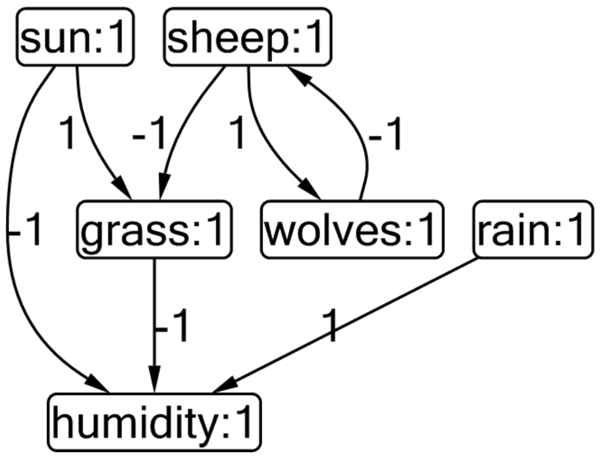


Fig. 3 Proposed model of ecosystem used for experiments

We performed a total of five experiments in which we observed the ability of the application to perform simulations and change of concept values by applying the sigmoid function for calculating the values of concepts, and in one experiment we used the method of averaging. In most experiments our model has stabilized and its concept values were not changing anymore after 4 or 5 iterations. A small number of iterations for stabilizing the system may be due the size of the proposed model and also because some concepts like Sun or Rain are not affected by other concepts and therefore their values do not change.

VII. Conclusion

Application was created with all the necessary features and functions supporting the creation of FCMs. These functions were tested on created model. The application also allows saving of created FCMs and their loading into application for further work. We also implemented algorithm PSO, which is used to automatically find corresponding optimal values of weight matrix.

Acknowledgment

This paper is supported by KEGA, Grant Agency of Ministry of Education of Slovak Republic, under Grant No. 014TUKE-4/2015 – “Digitalization, virtualization and testing of a small turbojet engine and its elements using stands for modern applied lecturing.“

References

[1] E.I. Papageorgiou, „A Review of Fuzzy Cognitive Maps Research During the Last Decade“, IEE Transactions on fuzzy systems, vol.21, No.1, pp. 66-79, February 2013

[2] S.A. Gray, S. Gray, L.J. Cox, S. Henly-Shepard, „Mental Modeler: A fuzzy-logic cognitive mapping modeling tool for adaptive environmental management“, 2013 46th Hawaii international conference on system sciences, pp. 965-973, January 2013

[3] G. Nápoles, I. Grau, R. Bello, M. León, K. Vahoof, E. Papageorgiou, FCM Tool- A Java Software Tool for Fuzzy Cognitive Maps, Available online: <http://epapageorgiou.com/images/A%20Java%20Software%20Tool%20for%20Fuzzy%20Cognitive%20Maps%2001-07-2015.pdf>

[4] FCM Research and Development Group: Software tools, Available online: <http://joomla.epapageorgiou.com/index.php/fcm-research-group>

[5] Microsoft Automatic Graph Layout: Github, Available online: <https://github.com/Microsoft/automatic-graph-layout/>

[6] M. Puheim, L. Madarász, J. Vaščák, "A Proposal for Multi-Purpose Fuzzy Cognitive Maps Library for Complex System Modeling." In: SAMI 2015: IEEE 13th International Symposium on Applied Machine Intelligence and Informatics: Proceedings. January 22-24, 2015, Herľany, Slovakia. - Danvers: IEEE, 2015, pp. 175-180. ISBN: 978-1-4799-8220-2

[7] K.E. Parsopoulos, E.I. Papageorgiou, P.P. Groumpos, M.N. Vrahatis, „A First Study of Fuzzy Cognitive Maps Learning Using Particle Swarm Optimization“, The 2003 Congress on Evolutionary Computation, vol.2, pp. 1440-1447, December 2003