Available online at www.sciencedirect.com

Fuzzy Sets and Systems 153 (2005) 371 – 401

www.elsevier.com/locate/fss

Genetic learning of fuzzy cognitive maps

Wojciech Stach, Lukasz Kurgan*, Witold Pedrycz, Marek Reformat

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada T6G 2V4

Received 27 August 2004; received in revised form 14 December 2004; accepted 18 January 2005
Available online 3 March 2005

Abstract

Fuzzy cognitive maps (FCMs) are a very convenient, simple, and powerful tool for simulation and analysis of dynamic systems. They were originally developed in 1980 by Kosko, and since then successfully applied to numerous domains, such as engineering, medicine, control, and political affairs. Their popularity stems from simplicity and transparency of the underlying model. At the same time FCMs are hindered by necessity of involving domain experts to develop the model. Since human experts are subjective and can handle only relatively simple networks (maps), there is an urgent need to develop methods for automated generation of FCM models. This study proposes a novel learning method that is able to generate FCM models from input historical data, and without human intervention. The proposed method is based on genetic algorithms, and requires only a single state vector sequence as an input. The paper proposes and experimentally compares several different design alternatives of genetic optimization and thoroughly tests and discusses the best design. Extensive benchmarking tests, which involve 200 FCMs with varying size and density of connections, performed on both synthetic and real-life data quantifies the performance of the development method and emphasizes its suitability.

Keywords: Fuzzy cognitive maps; Dynamic system modelling; Genetic algorithms; Decision analysis

1. Introduction

Fuzzy cognitive maps (FCMs) are a soft computing methodology introduced by Kosko in 1986 [16] as an extension of cognitive maps. They are used for modeling of dynamic systems [45,33]. FCM

* Corresponding author. Tel.: +1 780 492 5488; fax: +1 780 492 1811.
E-mail addresses: wstach@ece.ualberta.ca (W. Stach), lkurgan@ece.ualberta.ca (L. Kurgan), pedrycz@ece.ualberta.ca (W. Pedrycz), reform@ece.ualberta.ca (M. Reformat).

0165-0114/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2005.01.009

represents a given system as a collection of concepts and mutual relations among them, and are usually classified as neuro-fuzzy systems, which are capable to incorporate and adapt human knowledge [31]. Modelling dynamic systems using FCMs exhibits several advantages. Most importantly they are very simple and intuitive to understand, both in terms of the underlying formal model, and its execution. They are also characterized by flexibility of system design and control, comprehensible structure and operation, adaptability to a given domain, and capability of abstract representation and fuzzy reasoning [22].

The FCM models were developed and used in numerous areas of applications such as electrical engineering, medicine, political science, international relations, military science, history, supervisory systems, etc. Examples of specific applications include medical diagnosis [9], analysis of electrical circuits [39], analysis of failure modes effects [32], fault management in distributed network environment [27], modeling and analysis of business performance indicators [13], modeling of supervisors [40], modeling of software development project [37,38], modeling of plant control [11], modeling of political affairs in South Africa [15] and modeling of virtual worlds [7]. The diversity and number of applications clearly show popularity of this modelling technique, justifying further research to enhance it.

However, FCM development methods are far from being complete and well-defined, mainly because of the deficiencies that are present in the underlying theoretical framework [21]. According to the literature, the development of FCM models almost always relies on human knowledge [3]. As a consequence, the developed models strongly depend on subjective beliefs of expert(s) from a given domain. A few algorithms for automated or semi-automated learning of FCMs have been proposed, but none of them provides formalized approach that is suitable for convergence [30]. Some of the proposed learning methods also suffer from being applicable only to FCMs with binary states, require multiple input datasets, which might be difficult to obtain, and finally require human intervention during the learning process. To this end, the aims of this paper are to:

- introduce a new learning method, which allows for the development of FCM model with *continuous states* directly from experimental dataset, and *without* human intervention. Such method provides a fully automated solution for a problem of learning general class of FCMs.
- *compare several designs* of the learning methods based on genetic algorithms and based on extensive set of experiments select the most effective design environment.
- carry out well-organized, thorough *tests, considering large number of diverse types of FCMs*, and come up with firm design guidelines.

The remainder of this paper is organized as follows. Section 2 presents theoretical background concerning the model and learning methods, which includes standard development methods for FCM models, and a brief history of state-of-the-art algorithms for learning FCMs (Section 2.2). Section 3 introduces and provides background of the proposed learning approach, while Section 4 presents comprehensive experimental evaluation and discussion of the achieved results. Finally, Section 5 covers conclusions and future research directions.

2. Fuzzy cognitive maps (FCMs)

This section presents a historical overview of FCMs along with detailed background information concerning both the underlying model and the ensuing learning methods. First, the theoretical model is discussed, and specific examples of FCMs reported in the scientific literature are reported. Next, the related work is presented. Standard, expert work based, and learning methods for developing FCM models are discussed and compared with the proposed learning method.

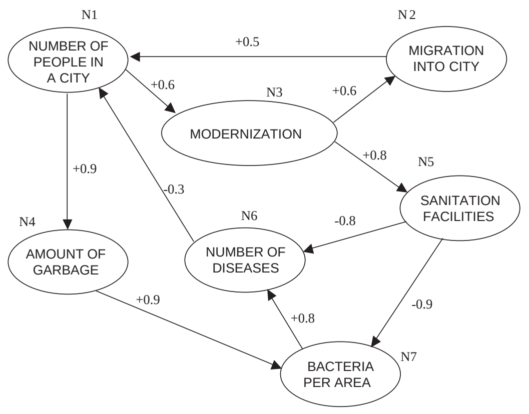
2.1. History and background

Political scientist Robert Axelrod originally proposed cognitive maps in 1976 [4]. They were used as a tool for representing social scientific knowledge. The cognitive maps model is represented by a simple graph, which consists of nodes and edges. The nodes represent concepts relevant to a given domain and the causal relationships between them are depicted by directed edges. Each edge is associated with a positive or negative sign that expresses a specific type of relationship. A positive edge from node A to node B indicates positive influence on B exerted by A. That means that increase in the value of node A will lead to increase in the value of node B, and vice-versa. A negative edge from node A to node B reflects negative type of relationship. It describes the situation, when increasing value of A leads to decreasing value of B. In many cases, however, this approach turned out to be insufficient because of limited ability to represent causality, which typically is not of a plain two-valued (Boolean) character, (i.e. captured by connections set up as -1 and $+1$), but rather continuous, i.e. expressed by a set of positive and negative numeric values.

The generic maps were significantly enhanced by Kosko, who introduced FCMs. The most significant improvement lies in the way of reflecting causal relationships. Instead of using only the sign, each edge is associated with a number (weight) that determines the degree of considered causal relation. This, in turn, allows implementing knowledge concerning the strength of relationship, which now can be described by a fuzzy term, such as weak, medium, strong, or very strong. In other words, a weight of directed edge from the node A to B quantifies how much concept A causes B [18]. The strength of relationship between two nodes (i.e. weight value) is usually normalized to the $[-1, 1]$ interval. Value of -1 represents full negative, $+1$ full positive, and 0 denotes no causal effect. As a result, a FCM model is fully described by set of nodes (concepts) and edges (cause-effect relationships), represented by weights, between them. Apart from the graph representation, for computational purposes, model can be equivalently defined by a square matrix, called *connection matrix*, which stores all weight values for edges between corresponding concepts represented by rows and columns. System with n nodes can be represented by $n \times n$ connection matrix. An example FCM model and its connection matrix are shown in Fig. 1.

FCM model was applied to many different areas to express dynamic behavior of a set of related concepts. A brief summary of several reported in literature models and their basic characteristics are shown in Table 1. The characteristics include number of nodes in the reported model, *density* that expresses the ratio of non-zero weights to the total number of weights, and *weight resolution (precision)* that concerns a resolution of the connections of the map. Precision refers to the minimal value that two weights might differ from each other. The smaller the precision, the more accurately a given system can be described by means of the FCM model. On the other hand, low values of precision make the process of finding weights more difficult and subjective.

As shown in Table 1, it becomes quite apparent that FCMs are usually relatively small, and typically consist of 5–10 nodes. Small size is a result of the manual development of such maps where we usually rely on expert knowledge. We note that mutual relationships among large number of concepts are hard to comprehend, analyze, and describe, which results in substantial difficulties in the construction of the corresponding maps. We also note that most of the FCMs are characterized by a relatively low, about 20–30% density (where by density we mean a percentage of the existing connections versus all possible



	N1	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**
N1	0	0	0.6	0.9	0	0	0
N2	0.5	0	0	0	0	0	0
N3	0	0.6	0	0	0.8	0	0
N4	0	0	0	0	0	0	0.9
N5	0	0	0	0	0	-0.8	-0.9
N6	-0.3	0	0	0	0	0	0
N7	0	0	0	0	0	0.8	0

Fig. 1. FCM model for public city health issues and the corresponding connection matrix [23].

Table 1
Example applications of FCMs

Reference	Application area	# Nodes	Density (%)	Weight precision
[23]	Public health issues	7	24	0.1
[42]	Heat exchanger model	5	50	0.025
[42]	Supervisor model for heat exchanger performance	5	75	0.001
[40]	Plant supervisor model	9	25	0.01
[20,21]	Industrial control	5	60	0.1
[25]	Crime and punishment model	7	36	1
[25]	EMU taxes and transfers model	6	27	1
[17]	Virtual squad of soldiers	10	34	0.01
[25]	EMU and the risk of war model	8	30	0.5
[1,3]	Simple model of a country	5	35	0.2
[45]	E-business company	7	40	0.1
[5]	Strategy formation model	6	33	0.01
[36]	Evidence of multiple suspicious events	6	20	0.2
[43]	Process control model	5	40	0.01
[41]	System for direct control of a process	8	23	0.01

connections for the given number of nodes). As a matter of fact, most maps reported in the literature are sparsely connected.

It is instructive to recall a formal definition of a FCM along with all necessary notation. Let **R** be the set of real numbers, while **N** denotes the set of natural numbers, $K = [-1, 1]$ and $L = [0, 1]$.

A **fuzzy cognitive map** F is a 4-tuple (N, **E**, **C**, f) where

1. $N = \{N_1, N_2, \ldots, N_n\}$ is the set of n concepts forming the nodes of a graph.
2. $\mathbf{E} : (N_i, N_j) \rightarrow e_{ij}$ is a function of $N \times N$ to K associating e_{ij} to a pair of concepts (N_i, N_j), with e_{ij} denoting a weight of directed edge from N_i to N_j, if $i \neq j$ and e_{ij} equal to zero if $i = j$. Thus $\mathbf{E}(N \times N) = (e_{ij}) \in K^{n\times n}$ is a connection matrix.

3. $\mathbf{C} : N_i \rightarrow C_i$ is a function that at each concept N_i associates the sequence of its activation degrees such as for $t \in \mathbf{N}$, $C_i(t) \in L$ given its activation degree at the moment t. $\mathbf{C}(0) \in L^n$ indicates the initial vector and specifies initial values of all concept nodes and $\mathbf{C}(t) \in L^n$ is a state vector at certain iteration t.
4. $f : \mathbf{R} \rightarrow L$ is a transformation function, which includes recurring relationship on $t \geqslant 0$ between $\mathbf{C}(t+1)$ and $\mathbf{C}(t)$

$$\forall i \in \{1, \ldots, n\}, \quad C_i(t+1) = f\left(\sum_{\substack{i=1 \\ j \neq i}}^{n} e_{ji} C_j(t)\right). \tag{1}$$

Eq. (1) describes a functional model of FCM, which is used to perform simulations of the system dynamics. Simulation consists of computing state of the system, which is described by a state vector, over a number of successive iterations. The state vector specifies current values of all concepts (nodes) in a particular iteration. Value of a given node is calculated from the preceding iteration values of nodes, which exert influence on the given node through cause–effect relationship (nodes that are connected to the given node). A number of improvements to the original FCM methodology have been proposed. One of them allows concepts to have non-zero edge values to themselves [43]. This paper exploits the original FCM approach, in which the functional model is defined by (1).

The transformation function is used to confine (clip) the weighted sum to a certain range, which is usually set to [0, 1]. The normalization hinders quantitative analysis, but allows for comparisons between nodes, which can be defined as active (value of 1), inactive (value of 0), or active to a certain degree (value between 0 and 1). Three most commonly used transformation functions are shown below.

- bivalent

$$f(x) = \begin{cases} 0, & x \leqslant 0, \\ 1, & x > 0, \end{cases} \tag{2}$$

- trivalent

$$f(x) = \begin{cases} -1, & x \leqslant -0.5, \\ 0, & -0.5 < x < 0.5, \\ 1, & x \geqslant 0.5, \end{cases} \tag{3}$$

- logistic

$$f(x) = \frac{1}{1 + e^{-Cx}}, \tag{4}$$

where C is a parameter used to determine proper shape of the function.

We can envision several simulation scenarios, which are dependent on transformation function [15]. Applying *discrete-output* transformation function (e.g. bivalent or trivalent function), the simulation heads to either a fixed state vector value, which is called *hidden pattern* or *fixed-point attractor*, or keeps cycling between a number of fixed state vector values, which is known as a *limit cycle*. Using a *continuous-output* transformation function (e.g. logistic signal function), the fixed-point attractor and limit cycle, as well

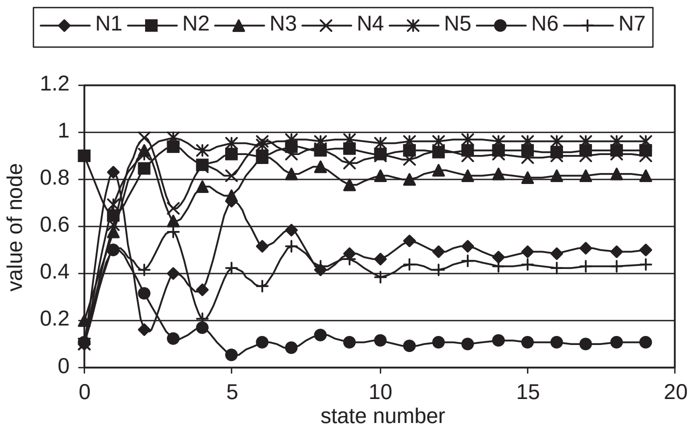


Fig. 2. Example input data.

as so called *chaotic attractor* can appear. The chaotic attractor is when the FCM continues to produce different state vector values for successive cycles. Fig. 2 shows an example of the fixed-point attractor simulation.

In a nutshell, simulation of a FCM results in a sequence of state vectors, which specify state of the modeled system in the successive iterations. The simulation results allow observation and analysis of each concept value, which represents the degree of its existence, over the time. Different scenarios can be considered by simulating the FCM with different *initial conditions*, which are represented by an initial state vector.

2.2. Related work

In general, two approaches to development of FCMs are used: manual and computational. Most, if not all, of the reported models were developed manually by domain expert(s) based on expert knowledge in the area of application. The experts design and implement adequate model manually based on their mental understanding of the modeled domain. Three main steps constitute this process [15]:

1. Identification of key domain issues or concepts.
2. Identification of causal relationships among these concepts.
3. Estimation of causal relationships strengths.

First two steps, which result in establishing of an initial draft of FCM model, include identification of concept nodes and relationships among them that are represented by edges. This is performed manually using pencil and paper by taking advantage of FCMs graph representation. However, the main difficulty is to accurately establish weights (strength) of the defined relationships. In order to achieve this, a following procedure might be used [15,42]:

1. The influence of a concept on another between each pair of concepts is determined as “negative”, “positive” or “none”.
2. All relationships are expressed in fuzzy terms, e.g. weak, medium, strong and very strong.

3. The established fuzzy expressions are mapped to numerical values, most frequently to the range from 0 to 1; for example weak is mapped to 0.25, medium to 0.5, strong to 0.75, and very strong to 1.0.

During establishing the fuzzy numerical value, analytical procedures, such as Analytical Hierarchy Process proposed by Saaty [34], may be applied [35].

Fuzzy cognitive maps model can be developed by a single expert or a group of experts [15]. Using a group of experts has the benefit of improving reliability of the final model. The FCM model allows for relatively simple aggregation of knowledge coming from multiple experts. Usually each expert develops his or her own FCM model, and the models are later combined together. Several procedures for combining multiple FCM models into single one exist [17].

In general, the manual procedures for developing FCM have a number of drawbacks. They require an expert, who has knowledge of the modeled domain, and at the same time knowledge about the FCMs formalism. Since even medium size models involve a large number of parameters, i.e. weights, very often it is difficult to obtain satisfactory performance. The development process may require many iterations and simulations before a suitable model is developed. In case of group development, additional parameters, such as credibility coefficients of each individual expert, need also to be estimated, which adds to the complexity of the overall process.

Manual methods for development of FCM models have also a major disadvantage of relying on human knowledge, which implies subjectivity of the developed model and problems with unbiased assessing of its accuracy. Questions, such as "*I believe, that this relationship is stronger than 0.5. Why did you choose this value to express it?*" often cannot be answered in a justifiable way. Also, in case of large and complex domains, the resulting FCM model requires large amount of concepts and connections that need to be established, which substantially adds to the difficulty of manual development process [46].

These problems led to the development of computational methods for learning FCM *connection matrix*, i.e. casual relationships (edges), and their strength (weights) based on historical data. In this way, the expert knowledge is substituted by a set of historical data and a computational procedure that is able to automatically compute the connection matrix. A number of algorithms for learning FCM model structure have been recently proposed. In general two main learning paradigms are used, i.e. Hebbian learning, and genetic algorithms, but so far none of proposed methods can be adopted as a formal methodology that is suitable for FCMs convergence [30].

In one of the first attempts, Kosko proposed simple Differential Hebbian Learning law (DHL) to be applied to learning of FCMs [8]. This law correlates changes of causal concepts:

$$\dot{e}_{ij} = -e_{ij} + \dot{C}_i \dot{C}_j, \tag{5}$$

where $\dot{e}_{ij}$ is the change of weight between concept *i*th and *j*th e_{ij} is the current value of this weight, and $\dot{C}_i \dot{C}_j$ are changes in concepts *i*th and *j*th values, respectively.

The learning process gradually updates values of weights of all edges that exist in the FCM graph until the desired connection matrix is found. In general, the weights of outgoing edges for a given concept node are modified when the corresponding concept value changes. The weights are updated according to the following formula:

$$e_{ij}(t+1) = \begin{cases} e_{ij}(t) + c_t[\Delta C_i \Delta C_j - e_{ij}(t)] & \text{if } \Delta C_i \neq 0, \\ e_{ij}(t) & \text{if } \Delta C_i = 0, \end{cases} \tag{6}$$

where e_{ij} denotes the weight of the edge between concepts C_i and C_j, ΔC_i represents the change in the C_i concept's activation value, t is the iteration number, and c_t is a learning decay coefficient.

The DHL was proposed in 1994, but there were no applications that used this approach to learning FCMs. In 2002, Vazquez presents an extension to DHL algorithm by introducing new formulas to update edge values [46]. The new algorithm, called Balanced Differential Algorithm (BDA) is based on weight update formula, for which updated value depends on values of all concepts that are acting at the same time as a cause of change for the concept. This method was applied only to FCMs with binary concept values, which significantly restricts its application areas. In 2003, Papageorgiou et al. developed another extension to Hebbian algorithm, called Nonlinear Hebbian Learning (NHL), to learn connection matrix of FCMs [29]. The main idea behind this method is to update weights associated only with edges that are initially suggested by expert(s). The NHL algorithm requires human intervention before the learning process starts, which is a substantial disadvantage. Another method of design of FCMs based on Hebbian algorithm was introduced in [30]. Active Hebbian Learning (AHL) introduces the determination of the sequence of activation concepts and improves the accuracy of FCMs. Nevertheless it still requires some initial human intervention.

Another main stream in computational methods for learning connection matrix of FCM involves application of genetic algorithms. In 2001, Koulouriotis et al. applied the Genetic Strategy (GS) to compute FCMs cause–effect relationships, i.e. weight values of the FCM model [21]. In this method, the learning process is based on a collection of input/output pairs, which are called examples. The inputs are defined as the initial state vector values, whereas the outputs are final state vector values, i.e. values of state vector after the FCM simulation terminates. Its main drawback is the need for multiple state vector sequences (input/output pairs), which might be difficult to obtain for many of real-life problems. Recently, in 2003, Parsopoulos et al. applied Particle Swarm Optimization (PSO) method, which belongs to the class of Swarm Intelligence algorithms, to learn FCM connection matrix based on a historical data consisting of a sequence of state vectors that leads to a desired fixed-point attractor state [31,28]. The algorithm was applied to find the connection matrix in a search space that is restricted to certain FCM concepts values and imposes constraints on the connection matrix, all of which are specified by domain expert(s). This method was tested only with one small FCM model that involves five concepts. Another recent work involving genetic algorithms was proposed by Khan and Chong, who performed a goal-oriented analysis of FCM in 2003 [14]. Their learning method did not aim to compute the connection matrix, but rather to find initial state vector, which leads a predefined FCM (a map with a fixed connection matrix) to converge to a given fixed-point attractor or limit cycle solution. The method was also tested with only one FCM model.

2.3. Objectives, scope and motivation

This study aims to provide a learning method, which avoids disadvantages of the existing methods. It uses a real-coded genetic (RCGA) algorithm to develop FCM connection matrix based on historical data consisting of one sequence of state vectors. In contrast, the approach introduced in [21], requires a set of such sequences. The proposed method is fully automatic, i.e. in contrast to NHL and AHL methods it does not require input from a domain expert. RCGA algorithm learns the connection matrix for a FCM that uses continuous transformation function, which is a more general problem that the one considered in [46]. Finally, the evaluation of the proposed method is performed in a very comprehensive manner. The tests involve ten fold cross-validation experiments for FCMs of varying sizes and densities. The above

Table 2
Overview of learning approaches applied to FCMs

Algorithm	Ref.	Learning goal	Human intervention	Type of data used[a]	FCM type		Learning type
					Transformation function	# nodes	
DHL	[8]	Connection matrix	No	Single	**N/A**	**N/A**	Hebbian
BDA	[46]	Connection matrix	No	Single	**Binary**	5,7,9	Modified Hebbian
NHL	[29]	Connection matrix	**Yes&No**[b]	Single	Continuous	5	Modified Hebbian
AHL	[28]	Connection matrix	**Yes&No**[b]	Single	Continuous	8	Modified Hebbian
GS	[22]	Connection matrix	No	**Multiple**	Continuous	7	Genetic
PSO	[30]	Connection matrix	No	**Multiple**	Continuous	5	Swarm
GA	[14]	**Initial vector**	N/A	N/A	Continuous	11	Genetic
RCGA	This paper	Connection matrix	No	Single	Continuous	4,6,8,10	Genetic

[a] *Single*—historical data consisting of one sequence of state vectors, *Multiple*—historical data consisting of several sequences of state vectors, for different initial conditions.
[b] Initial human intervention is necessary but later when applying the algorithm there is no human intervention needed.

methods were tested on several, or sometimes even only one FCMs, and therefore lack comprehensive set of experiments, which would allow appropriate assessment of accuracy and correctness. In contrast, the proposed method was tested on almost 200 different FCMs.

To easy comparison between the above methods, and the proposed learning algorithm, a brief summary is shown in Table 2. The table compares the methods based on several factors, such as the learning goal, involvement of a domain expert, input historical data, type of transformation function, and learning strategy type. It also shows, in the "number of nodes" column, for how many and of what size FCM model a given method was tested. All learning methods, except the RCGA, were tested on a single map of the indicated size. For the RCGA method, almost 50 maps for a given size were simulated. Values in bold indicate the main disadvantage of a given learning method.

We note that the proposed method is a natural continuation of the research performed in the domain of learning FCM connection matrix. It draws conclusions from the methods proposed in the past, and provides substantial advancement. Next section provides detailed description of the proposed method.

3. Proposed learning method

The proposed learning method aims to learn the FCM connection matrix using a genetic algorithm. First, a detailed problem statement is formulated, which is followed by a detailed description of a specific genetic algorithm that was applied.

3.1. Problem statement

Based on the formal definition of fuzzy cognitive map presented in Section 2.1, the objective of the FCM learning is to determine the matrix connection $\hat{\mathbf{E}}$ given the set of concepts N, sequence of their activation

Table 3
Connection matrix for a FCM model that consists of N concept nodes

	N_1	N_2	...	N_{N-1}	N_N
N_1	0	e_{12}	...	e_{1N-1}	e_{1N}
N_2	e_{21}	0	...	e_{2N-1}	e_{2N}
...	...	...	...	...	...
N_{N-1}	e_{N-11}	e_{N-12}	...	0	e_{N-1N}
N_N	e_{N1}	e_{N2}	...	e_{NN-1}	0

degrees, called input data, $\mathbf{C}(t)$ at certain iteration interval $t \in \{0, \ldots, t_K\}$, and some transformation function *f*, such that the FCM minimizes the error between given sequence $\mathbf{C}(t)$ and sequence $\hat{\mathbf{C}}(t)$ obtained from running the FCM model with initial condition specified as $\hat{\mathbf{C}}(0) = \mathbf{C}(0)$. The error can be measured in different ways, as will be explained later on.

The aim of the proposed learning method is to eliminate human intervention during development of a FCM model. This process is performed by exploiting information from historical data to compute FCM model connection matrix that is able to mimic the data. The input (historical) data comprise of one sequence of state vectors over time. The *data length* is defined as the number of successive iterations (time points) of the given historical data. The input data is used to compute a FCM model, called *candidate FCM*, by applying a learning procedure that uses RCGA algorithms.

Assuming that edges are allowed only between different concepts (nodes), i.e. concepts do not exhibit cause–effect relationships on themselves, connection matrix of a FCM model can be completely described by $N(N-1)$ variables, where N is the number of concepts. Thus, the learning of FCM connection matrix boils down to computing $N(N-1)$ parameters, which are shown in Table 3.

The proposed learning algorithm uses input data to find the parameters. Input data is a sequence of states described by state vectors at a particular time (iteration). They illustrate the system's behavior over time, and are represented by a set of state vectors $\mathbf{C}(t)$ at time point *t*. The input data can be plotted to easy understanding and analysis of dependencies among concepts, see Fig. 2. The example plot corresponds to the fixed-point attractor simulation.

The proposed learning method constructs a connection matrix based on input data. The learned model objective is to generate the same state vector sequence for the same initial state vector, as it is defined by the input data. At the same time, the learned model generalizes the inter-relationship between concept nodes, which are inferred from the input data. Therefore, the FCM model is suitable to perform simulation for different initial state vectors, and quantify the degree and type of the cause–effect relationships between the concepts. The learning method uses real-coded genetic algorithm, described next, which allows eliminating expert involvement during development of the model. A high-level overview of the learning process is shown in Fig. 3.

3.2. *Proposed genetic algorithms based learning method*

Genetic algorithms (GAs) are used to perform optimization and search tasks. They have been used in numerous and diverse problem domains [6]. Their origin and principles were inspired by natural genetics [10]. Their advantages are broad applicability, relative ease of use, and global perspective. In this paper we

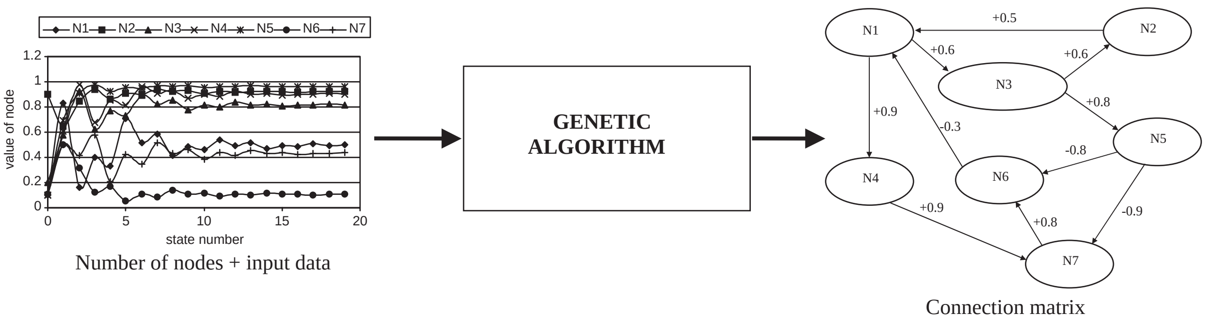


Fig. 3. High-level diagram of the proposed learning method.

assume reader's familiarity with GAs. A useful summary about relevant GAs can be found in [6,10,12]. The proposed learning method uses an extended GA called real-coded genetic (RCGA) algorithm, where a chromosome consists of floating point numbers. RCGA algorithm performs linear transformation for each variable of the solution to decode it to the desired interval. Its main advantages are ability to use with highly dimensional and continuous domains, and richer spectrum of evolution operators that can be applied during the search process [12]. The RCGA algorithm uses the input data to develop and optimize, with respect to the input data, connection matrix of a candidate FCM model. The following sections provide details as to all essential elements of the RCGA environment, including a structure of chromosomes, fitness function, stopping condition, genetic operators, and selection strategy.

3.2.1. *Chromosome structure*

RCGA defines each chromosome as a floating-point vector. Its length corresponds to the number of variables in a given problem. Each element of the vector is called *gene*. In case of the learning FCMs, each chromosome consists of $N(N-1)$ genes, which are floating point numbers from the range $[-1, 1]$, defined as follows:

$$\hat{\mathbf{E}} = [e_{12}, e_{13}, \ldots, e_{1N}, e_{21}, e_{23}, \ldots, e_{2N}, \ldots, e_{NN-1}]^{\mathrm{T}},$$

where e_{ij} specifies the value of a weight for an edge from *i*th to *j*th concept node.

Each chromosome has to be *decoded* back into a candidate FCM. This involves copying weight values from the chromosome to the corresponding cell in the connection matrix, which defines connection matrix of the FCM model; for details see Section 3.1. The number of chromosomes in a population is constant for each generation, and it is specified by the *population_size* parameter.

3.2.2. *Fitness function*

One of the most important considerations for a successful application of GA is design of a fitness function, which is appropriate for a given problem. In case of application to learning connection matrix of FCMs, the design takes advantage of a specific feature of the FCM theory. At each iteration of FCM model simulation state vector $\mathbf{C}(t+1)$ depends only on the state vector at the preceding iteration. This implies that whenever system reaches the state that has been already reached in one of the preceding iteration, its behavior will be exactly the same regardless the simulation history. This results in so called limit cycle and means that even if a given input data length is *K*, but the limit cycle or fixed point attractor occurs at the Lth iteration, $L < K$, the input data that can be used for learning has to be truncated to the first L iterations. The remaining $K - L$ iterations should not be used since they describe system's behavior, which is already described by the first L iterations, and therefore would only add unnecessary computational complexity, for details see Section 2.

Let us assume that the input data length is K, where all iterations that happen after a limit cycle or fixed pattern attractor occur, are already ignored. By grouping each two adjacent state vectors, $K - 1$ different pairs can be formed:

$$\mathbf{C}(t) \to \mathbf{C}(t+1) \quad \forall t = 0, \ldots, K-1. \tag{7}$$

If we define $\mathbf{C}(t)$ as an *initial vector*, and $\mathbf{C}(t+1)$ as *system response*, $K - 1$ pairs in the form of {initial vector, system response} can be generated from the input data. The larger is K the more information about the system behavior we have. The fitness function is calculated for each chromosome by computing the difference between system response generated using a candidate FCM and a corresponding system response, which is known directly from the input data. The system response of the candidate FCM is computed by decoding the chromosome into a FCM model and performing one iteration simulation for initial state vector equal to the initial vector. The difference is computed across all $K - 1$ initial vector/system response pairs, and for the same initial state vector. Three measures of error based on L_1*-norm*, L_2*-norm*, and L_∞*-norm* are shown below:

$$\text{Error\_}L_p = \alpha \sum_{t=1}^{K-1} \sum_{n=1}^{N} |C_n(t) - \hat{C}_n(t)|^p, \tag{8}$$

where $\mathbf{C}(t) = [C_1(t), C_2(t), \ldots, C_n(t)]$ the known system response for $\mathbf{C}(t-1)$ initial vector, $\hat{\mathbf{C}}(t) = [\hat{C}_1(t), \hat{C}_2(t), \ldots, \hat{C}_n(t)]$ the system response of the candidate FCM for $\mathbf{C}(t-1)$ initial vector, $p = 1, 2, \infty$ the norm type, α the parameter used to normalize error rate, which is equal to $1/(K-1) \cdot N$ for $p \in \{1, 2\}$, and $1/K - 1$ for $p = \infty$, respectively.

Each of these error measures can be used as the core of fitness function

$$\text{Fitness function} = h(\text{Error\_}L_p) \tag{9}$$

where h is an auxiliary function.

The auxiliary function h was introduced for two main reasons:

1. To ensure that better individuals correspond to greater fitness function values. Argument of this function is the summed error rate, and thus needs to be inversed.
2. To embed non-linearity that aims to reward chromosomes, which are closer to the desired solution.

The following function h was proposed:

$$h(x) = \frac{1}{ax+1}, \tag{10}$$

where parameter a is established experimentally.

The fitness function is normalized to the $(0, 1]$ where:

- the worse is the individual the closer to zero its fitness value is;
- fitness value for an ideal chromosome, which results is exactly the same state vector sequence as the input data, is equal to one.

```
if (Fitness function(best individual) > max_ fitness
or
t>max_ generation)
then stopping_condition = true;
where:best individual–the chromosome in the current generation with
highest fitness function value, t – current generation number
```

Fig. 4. Definition of the stopping condition.

3.2.3. Stopping condition

The proposed stopping condition takes into consideration two possible scenarios of the learning process:

- the learning is successful, i.e. the state vector sequence obtained by simulating the candidate FCM is identical or satisfactory close to the input data. The similarity is described by the fitness function value for the best chromosome in each generation. Therefore, the learning should be terminated when the fitness function value reaches a threshold value called *max_fitness*;
- the learning in not successful, but a maximum number of generations, defined by the *max_generation* parameter, has been reached.

The *stopping_condition* is defined in Fig. 4.

Both, the *max_fitness* and *max_generation* parameters are established experimentally and given in Section 4.1.1.

3.2.4. Evolutionary operators and selection strategy

Recombination is performed using crossover operations.

There are many different crossover operators used in RCGA, cf. [12,47]. In our experiments, we consider a simple one-point crossover. It carries a low computational cost yet as demonstrated through experiments, it effectively handles the optimization problem. Also, three different mutation operators were applied: random mutation [24] non-uniform mutation [24], and Mühlenbein's mutation [26]. Two popular selection strategies, such as roulette wheel selection and tournament selection, were applied [12,10].

4. Experiments and results

4.1. Experimental setup

The goal of the experiments is to assess quality of the proposed method for learning FCMs. First, the experimental results that were focused on selecting best fitness function from those proposed in Section 3.2.2 are performed and reported. The aim of these experiments is to analyze the influence of fitness function on the learning process and select the one that gives the best quality in terms of convergence and accuracy. The tests carried out with the selected best function, are divided into two groups: tests performed with *synthetic*, and with *real-life* data. The first group uses synthetic input data, which are generated from randomly generated input FCM models. The latter group uses real-life, i.e. previously published in a scientific literature, input FCMs to generate the input data. Evaluation of the solution

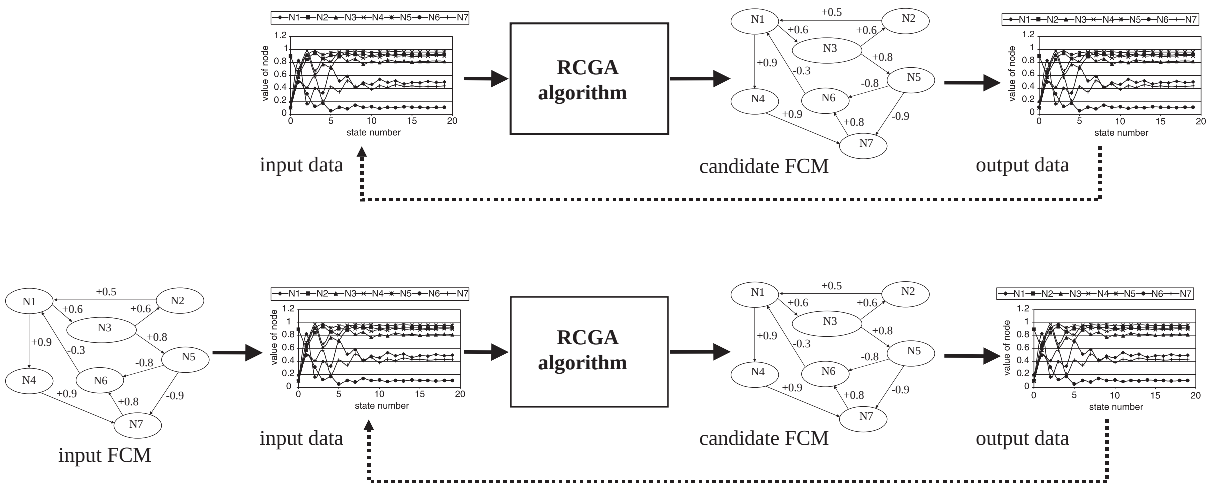


Fig. 5. High-level diagram of the experimental setup.

quality is performed by testing similarity between the data and by studying generalization capabilities of the candidate FCM. The latter criterion takes advantage of availability of the input FCM model and is performed by running simulation from new, randomly generated, initial state vectors and observing if the two maps exhibit the same behavior. A diagram presenting an overview of the test procedure is shown in the bottom part of Fig. 5. The upper part of Fig. 5 shows a typical application of the proposed learning method, where the RCGA algorithm is used to infer FCM connection matrix from the input data.

In general, the goals of learning FCM can be divided into two categories. The first objective is to find FCM connection matrix that generates the same state vector sequence as the input data for a given initial state vector. Since there is a risk that overfitted models would behave correctly only for this particular initial state vector, the second goal is to find candidate FCM that behaves similarly to the input FCM for different initial state vectors. The fulfillment of this goal is guaranteed only in case, in which the model connection matrices are exactly the same. However, since we cannot assume that only FCM with identical connection matrices behave the same, a test that simulates the input and the candidate FCMs for a set of different initial state vectors, and compares their outcomes, need to be performed. To evaluate quality of tests that need to be performed to verify the above goals two different criteria were developed. They aim to evaluate quality of the candidate FCM based on similarity of input data and generalization. Below, the two criteria are defined and explained:

1. *Input data error criterion*, which measures similarity between the input data, and data generated by simulating the candidate FCM with the same initial state vector as for the input data. The criterion is defined as a normalized average error between corresponding concept values at each iteration between the two state vector sequences:

$$\text{error\_initial} = \frac{1}{(K-1)\cdot N}\sum_{t=1}^{K-1}\sum_{n=1}^{N}|C_n(t) - \hat{C}_n(t)|, \tag{11}$$

where $C_n(t)$ is the value of a node n at iteration t in the input data, $\hat{C}_n(t)$ the value of a node n at iteration t from simulation of the candidate FCM, K the input data length and N is the number of nodes.

2. *Model behavior error criterion*, which measures generalization capabilities of the candidate FCM. To compute this criterion, both the input and the candidate FCMs are simulated from P randomly chosen initial state vectors. Next, the error_initial value is computed for each of the simulations to compare state vector sequences generated by the input and the candidate FCM, and an average of these values is computed:

$$\text{error\_behavior} = \frac{1}{P \cdot (K-1) \cdot N} \sum_{p=1}^{P} \sum_{t=1}^{K-1} \sum_{n=1}^{N} |C_n^p(t) - \hat{C}_n^p(t)|, \tag{12}$$

where $C_n^p(t)$ is the value of a node n at iteration t for data generated by input FCM model started from pth initial state vector, $\hat{C}_n^p(t)$ the value of a node n at iteration t for data generated by candidate FCM model started from pth initial state vector, K the input data length, N the number of nodes and P is the number of different initial state vectors.

Both of these criteria are used for both synthetic and real-life data tests to quantify quality of learning candidate FCM. In order to put the quality of the proposed learning method into a perspective, a set of baseline results were computed. The baseline was generated by computing these criteria values for randomly generated FCMs and comparing them with the input FCM. Ten random FCM were generated for each test category, and the average value of the two criteria was reported as the baseline.

The quality of learning depends on the quality of the input data. In case of FCMs, for the learning to be successful, the limit cycle or fixed point attractor must not appear in the early few simulation iterations of the input FCM. The state vector values after the limit cycle follow a cycle, which is already described by proceeding state vector values, while state vector values after the fixed point attractor are constant. Therefore, learning algorithm uses state vector values, but only these which are proceeding the limit cycle or fixed point attractor. As a result, if they appear early in the sequence, the input data may be too short to accurately learn candidate FCM that corresponds to the input FCM. In this case, it is possible to learn candidate FCM that generates state vector sequence identical to the input data, but cannot be considered as a good solution for this problem because lacking of fulfillment the generalization criterion, i.e. error_initial value is small, yet error_behavior is relatively high. To avoid this problem, the minimum number of iterations before limit cycle or fixed point attractor, called *min_data_length*, was set to 20 for all tests. Also, the *max_data_length* parameter that specifies the maximum number of iterations before limit cycle or fixed point attractor, which was set to 50, was introduced to reduce simulation time required to perform experiments. In general, bigger length of the input data improves possibility of learning high-quality candidate FCM, but also increases computational time.

Several other minor assumptions were also made. For both, the input and candidate FCMs, all weight values smaller than 0.05 were rounded down to 0, since no real-life map considers such weak relationships. Moreover, during simulations all nodes values are rounded to two digits after decimal point, which results from a trade-off between model comprehensibility and accuracy of relationship representation.

4.1.1. RCGA parameters

The parameters, which were defined in the Section 3.2, were instantiated with specific values based on simulations and learning performance for experiments performed with different FCM sizes and densities.

The experiments were performed by generating a random FCM, which was used to generate input data. Next, the RCGA algorithm was used to generate a candidate FCM based on the input data. The goal was to find values, which lead to convergence regardless of the size and density of the FCM in reasonable running time. Totally, 50 experimental learning results were visually inspected to establish the values, which are reported below:

- recombination method—single-point crossover;
- mutation method—randomly chosen from random mutation, non-uniform mutation, and Mühlenbein's mutation;
- selection method—randomly chosen from roulette wheel and tournament;
- probability of recombination: 0.9;
- probability of mutation: 0.5; A high value of mutation probability is due a large number of sub-optimal solutions for each FCM model. Low mutation value leads to slow exploration of the search space, and, in consequence, the algorithm may get stuck in sub-optimal solution, which is very good in terms of the error_initial criterion yet carries a substantial error_behavior value. The completed experiments show that high mutation rates ensure better performance of the final model.
- *population_size*: 100 chromosomes;
- *max_generation*: 300,000;
- *max_fitness*: 0.999;
- *fitness function* – based on L_1, L_2, and L_∞ norms, see Section 3.2.2 for details;

All experiments were performed with logistic transformation function, see Section 2.1, as a generalization of *discrete-type* function, which results in more flexible representation of node activation degrees. This function is parameterized and given below:

$$f(x) = \frac{1}{1+e^{-5x}}. \tag{13}$$

The parameter value of 5 is commonly used in simulations performed with FCM models [25].

A set of experiments focused on comparing different fitness functions, which is described in Section 4.2.1, was preceded by establishing parameter *a* for the auxiliary function for all considered fitness functions, see formula (10). This was done by performing 20 simulations for each function and observing convergence to the desired solutions. As a result, for fitness functions based on L_1 and L_∞ norm *a* was set to 100, whereas for function based on L_2 norm, it was set to 10,000. The final formula for all considered fitness functions is presented below:

$$\text{Fitness\_function\_}L_p = \frac{1}{a \cdot \text{Error\_}L_p + 1}, \tag{14}$$

where $p = 1, 2, \infty$ is the norm that is applied to fitness function, Error_L_P the formula introduced in Section 3.2.2, $a = 100$ for L_1 and L_∞ norms; $a = 10,000$ for L_2 norm.

Two examples of FCM learning experiments with fitness function based on L_2 norm that use the above parameters are plotted in Figs. 6 and 7. The figures show fitness value of the best chromosome, the average fitness value through the entire population during the learning process, with respect to the iteration number, and the average weights change of best solution between two consecutive iterations every 3000 iterations. Please note that weights change bars are plotted using logarithmic scale on *y*-axis, which is shown on the right-hand side of each plot.

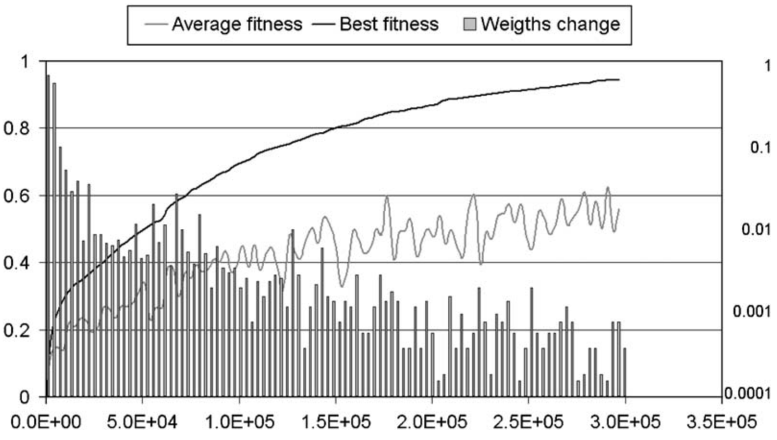


Fig. 6. FCM with 8 nodes and 40% density.

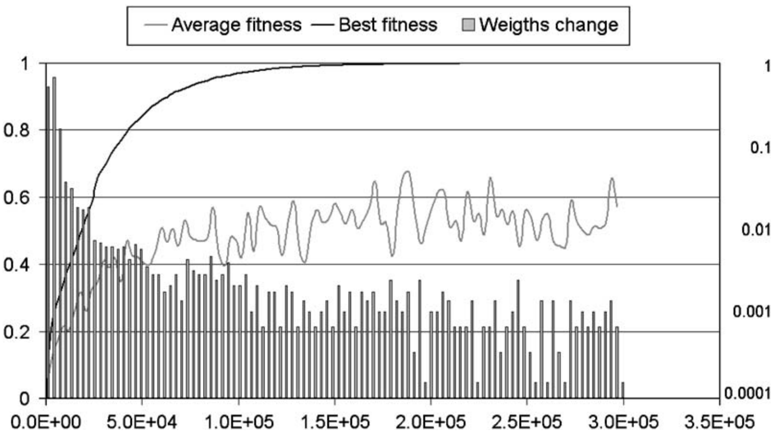


Fig. 7. FCM with 6 nodes and 60% density.

The results show that the RCGA algorithm gradually converges into a high-quality candidate FCM.

4.2. Experiments outcome

4.2.1. Experiments to select the best fitness function

The goal of these experiments is to choose fitness function for the genetic learning of FCM. The set of tests, similar to experiments with synthetic data described in Section 4.2.2, is carried out with all three functions proposed in Section 3.2.2. In order to achieve reliable results, several different FCM model configurations are used in these experiments, i.e. model sizes of 6 and 8 nodes, and densities of 20%, 40%, 60% and 80%. In order to compare between different fitness functions, for each of them a baseline value was estimated. The baseline was computed by choosing 10 different randomly generated FCM models and calculating the corresponding error coefficients, see formula (8), and taking an average value.

Table 4
Experiment results with different fitness functions

FCM parameters		Fitness function L_1		Fitness function L_2		Fitness function L_∞	
# Nodes	Density (%)	Error_L_1	Ratio (%) Error_L_1 /Baseline_L_1	Error_L_2	Ratio (%) Error_L_2 /Baseline_L_2	Error_L_∞	Ratio (%) Error_L_∞ /Baseline_L_∞
6	20	1.30E − 03	3.60E − 01	2.40E − 06	1.17E − 03	1.48E − 02	1.77E + 00
6	40	2.92E − 03	8.06E − 01	5.00E − 08	2.44E − 05	1.30E − 02	1.55E + 00
6	60	3.78E − 03	1.04E + 00	2.00E − 07	9.76E − 05	9.83E − 03	1.18E + 00
6	80	3.39E − 03	9.35E − 01	5.00E − 08	2.44E − 05	9.97E − 03	1.19E + 00
8	20	5.53E − 03	1.53E + 00	1.38E − 05	6.73E − 03	3.15E − 02	3.77E + 00
8	40	8.25E − 03	2.28E + 00	1.07E − 05	5.22E − 03	3.29E − 02	3.94E + 00
8	60	5.60E − 03	1.55E + 00	2.04E − 05	9.95E − 03	2.78E − 02	3.32E + 00
8	80	5.59E − 03	1.54E + 00	1.62E − 05	7.90E − 03	1.58E − 02	1.89E + 00

Each considered FCM, in terms of the number of nodes and density, was simulated 10 times with the three fitness functions, which totalled in 240 experiments. Table 4 reports the results, which include average error rates and the corresponding baselines. The baselines for the corresponding error rates, which have been obtained experimentally, have the following values:

Baseline_L_1: 0.362,
Baseline_L_2: 0.205,
Baseline_L_∞: 0.836.

Table 4 reports the ratio of error in each of the metrics with respect to the corresponding baseline, which expresses accuracy of the learned FCM with respect to the randomly generated solution.

By analyzing the corresponding ratio values, the results show that the best performance is achieved with fitness function based on L_2 norm. In order to further verify this hypothesis, the results for L_1 and L_∞ based fitness functions were expressed using the L_2 norm. This allows for a direct comparison among the three fitness functions, see Table 5.

Fig. 8 shows relationship between FCM parameters, i.e. number of nodes and density, and ratio values in L_2 for each fitness function.

Analyzing the results, the best convergence is consistently achieved with L_2-based fitness function. Error rates for the two other functions are bigger and increase more rapidly with the increasing FCM size. The effectiveness of the L_2-based fitness function comparing to the L_1-based one comes from the fact that the former one is more sensitive on errors, which leads to faster convergence. On the other hand L_∞ based norm gives reasonably good results for small FCMs, yet is ineffective when the search space increases.

We note that these experiments show that proper selection of fitness function is of great importance for genetic algorithms. Considering the above conclusions, the remaining experiments, i.e. for both synthetic and real-life data, are carried out with the best fitness function, which is based on L_2 norm.

Table 5
Comparison of learning quality for different fitness functions

FCM parameters		Fitness function L_1		Fitness function L_2		Fitness function L_∞	
# Nodes	Density (%)	Error_L_2	Ratio (%) Error_L_2/ Baseline_L_2	Error_L_2	Ratio (%) Error_L_2/ Baseline_L_2	Error_L_2	Ratio (%) Error_L_2/ Baseline_L_2
6	20	3.30E − 04	1.61E − 01	2.40E − 06	1.17E − 03	4.87E − 03	2.38E + 00
6	40	9.88E − 03	4.82E + 00	5.00E − 08	2.44E − 05	2.49E − 03	1.21E + 00
6	60	2.13E − 02	1.04E + 01	2.00E − 07	9.76E − 05	4.98E − 04	2.43E − 01
6	80	8.18E − 03	3.99E + 00	5.00E − 08	2.44E − 05	1.66E − 03	8.09E − 01
8	20	3.16E − 02	1.54E + 01	1.38E − 05	6.73E − 03	7.78E − 02	3.80E + 01
8	40	2.63E − 02	1.28E + 01	1.07E − 05	5.22E − 03	6.77E − 02	3.30E + 01
8	60	1.18E − 02	5.74E + 00	2.04E − 05	9.95E − 03	1.95E − 02	9.50E + 00
8	80	2.54E − 03	1.24E + 00	1.62E − 05	7.90E − 03	2.22E − 03	1.08E + 00

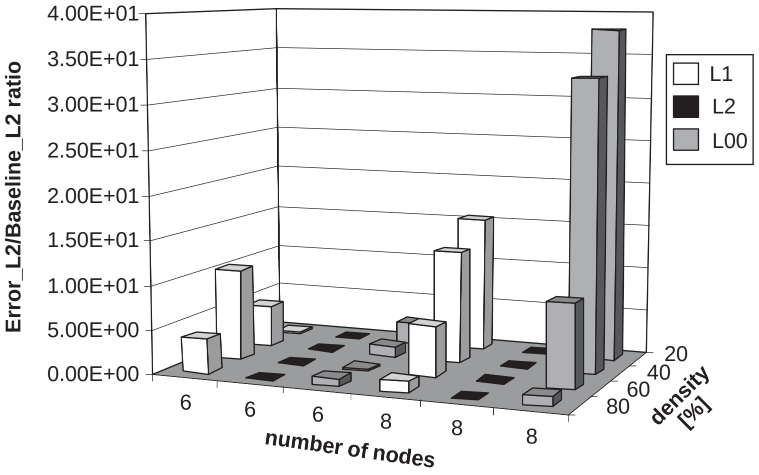


Fig. 8. Influence of different fitness functions on learning quality.

4.2.2. Synthetic data

These tests use randomly generated input FCM models to generate synthetic input data, which is used to learn candidate FCM. A wide range of input FCM models was considered to accommodate different types of real-life FCM models reported in Table 1. As a result, the tests were performed with FCMs with 4, 6, 8, and 10 nodes, and with densities of 20%, 40%, 60% and 80%, which results in 16 test configurations. Each test was carried out according to the following routine:

1. *Set the parameters*: *number of nodes N, density D, number of experiments T, number of tests P, current experiment* $t = 1$
2. *Generate a random FCM model with N nodes and D density*
3. *Generate random initial state vector*

```
 4. Perform simulation for the generated initial state vector to generate input data
 5. IF the input data length is NOT between min_data_length and max_data_length THEN go to 2
 6. Apply RCGA algorithm to learn candidate FCM model
 7. Compute parameter error_initial for the candidate FCM model,
 8. Compute error_behavior parameter by performing P simulations for input and candidate FCMs with
    new, randomly generated initial state vectors
 9. IF $t < T$ THEN increase t by one AND go to 2
10. Report average values and standard deviations for error_initial and error_behavior
```

Parameter *T*, which specifies number of experiments carried for a given configuration, and *P*, which specifies number of tests with a different initial state vector in each case, were set to 10. Thus, the average values and standard deviations for the 10 experiments are reported.

A total of almost 200 tests were performed. For each FCM size and density 10 tests were performed, and average results are reported. Additionally, for each of the 10 tests, 10 simulations to compute the error_behavior value, which bring the number of simulations to almost 2000, were performed, and the average values are reported.

Following, selected results achieved by the RCGA algorithms, in terms of comparison of state vector sequences generated by candidate and input FCMs for several different initial state sequences, and error defined by the difference between the input data and the generated sequence, are presented. Figs. 9 and 10 show data, which were used to compute values of the error_initial and error_behavior criteria. They present comparison, in terms of plots and errors, between the input data and state vector sequence generated by the candidate FCM for the input state vector defined in the input data; see Figs. 9a, b, e, f and 10a, b, e, f. They also present comparison between the best and the worst state vector sequences, in terms of similarity to sequences generated by the input FCM, generated by the candidate FCM, and state vector sequences generated by the input FCM for experiments performed using 10 new, randomly generated input state vectors; see Figs. 9c, d, g, h and 10c, d, g, h. The former results were used to compute the error_initial value, while the latter were used to compute error_behavior value.

Figs. 9 and 10 present results of two representative experiments, one for 8 nodes and 40% density, and another for 6 nodes and 60% density. For each experiment, the best and the worst, in terms of computing the error_initial value, experiments were selected and presented using four consecutive figures. The figures show:

1. the input data and the result of simulation of candidate FCM started from initial condition defined in the input data (Figs. 9a,e and 10a,e),
2. error rate expressed as an absolute difference between corresponding state vector values between the two sequences described above (Figs. 9b,f and 10b,f),
3. results of the best among ten experiments performed with a new initial state vector for both input and candidate FCM model (Figs. 9c,g and 10c,g),
4. results of the worst among ten experiments performed with a new initial state vector for both input and candidate FCM model (Figs. 9d,h and 10d,h).

The following rules characterize the figures:

- Plots shown in black depict simulations of input FCM, while grey plots concern simulations performed with the candidate FCM. Ideally, they should overlap, and in such a case only grey lines are visible.

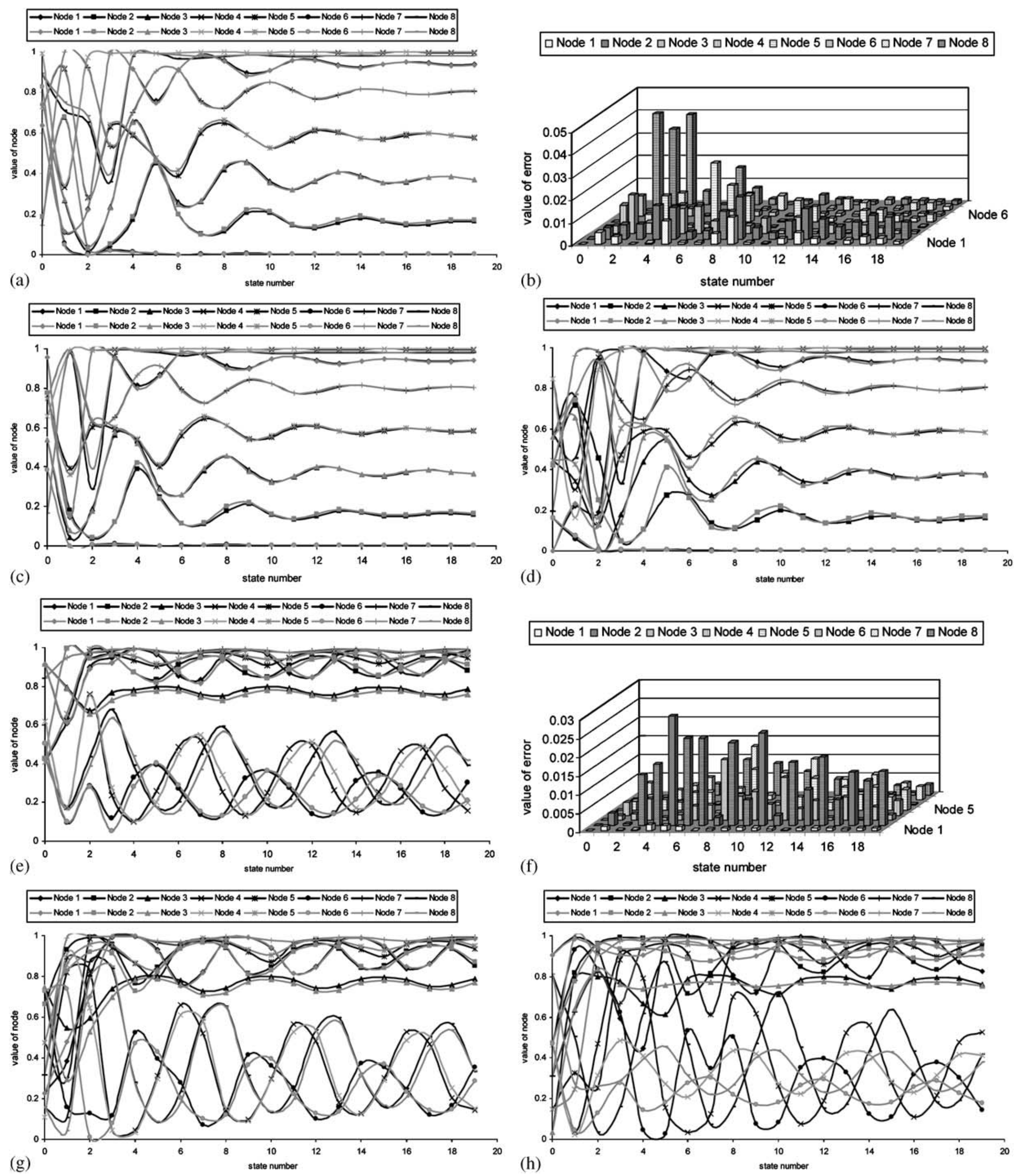


Fig. 9. Experiments for nodes = 8 and density = 40%: (a) Input data and plot obtained from simulation of candidate FCM; (b) Difference between corresponding states vector values for plots from (a); (c) Comparison of the best simulation of candidate and input FCMs with new state vector; (d) Comparison of the worst simulation of candidate and input FCMs with new state vector; (e) Input data and plot obtained from simulation of candidate FCM; (f) Difference between corresponding states vector values for plots from (e); (g) Comparison of the best simulation of candidate and input FCMs with new state vector; (h) Comparison of the worst simulation of candidate and input FCMs with new state vector.

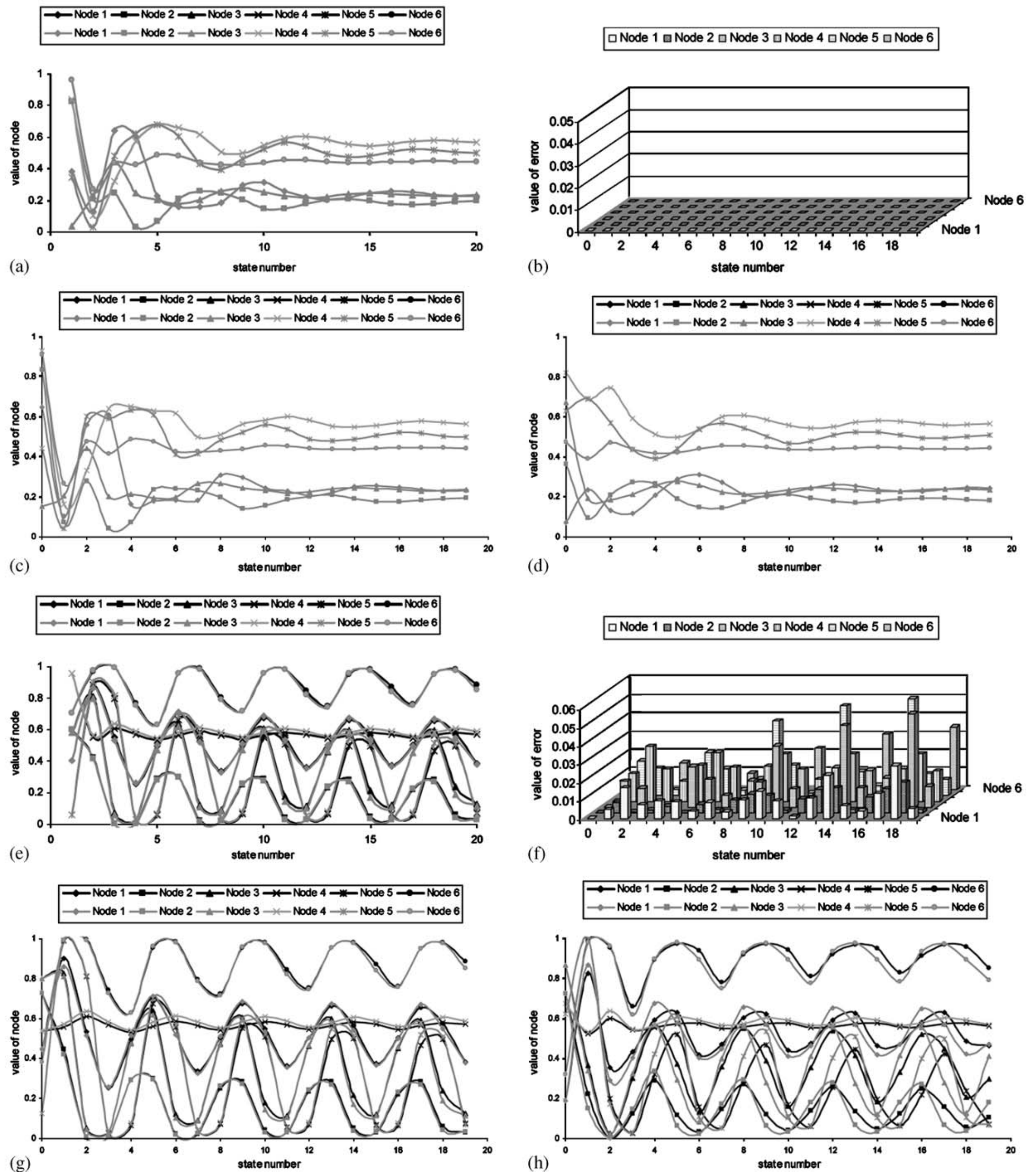


Fig. 10. Experiments for nodes = 6 and density = 60%: (a) Input data and plot obtained from simulation of candidate FCM; (b) Difference between corresponding states vector values for plots from (a); (c) Comparison of the best simulation of candidate and input FCMs with new state vector; (d) Comparison of the worst simulation of candidate and input FCMs with new state vector; (e) Input data and plot obtained from simulation of candidate FCM; (f) Difference between corresponding states vector values for plots from (e); (g) Comparison of the best simulation of candidate and input FCMs with new state vector; (h) Comparison of the worst simulation of candidate and input FCMs with new state vector.

Table 6
Experimental results for the synthetic data

Nodes	Density (%)	Error_initial ± stdev	Error_behavior ± stdev
4	20	0.000 ± 0.000	0.001 ± 0.001
4	40	0.000 ± 0.000	0.000 ± 0.000
4	60	0.000 ± 0.000	0.000 ± 0.000
4	80	0.000 ± 0.000	0.000 ± 0.000
6	20	0.005 ± 0.005	0.017 ± 0.030
6	40	0.005 ± 0.006	0.018 ± 0.027
6	60	0.004 ± 0.004	0.022 ± 0.029
6	80	0.003 ± 0.003	0.011 ± 0.016
8	20	0.057 ± 0.043	0.102 ± 0.086
8	40	0.015 ± 0.021	0.052 ± 0.063
8	60	0.014 ± 0.020	0.056 ± 0.060
8	80	0.006 ± 0.008	0.036 ± 0.060
10	20	0.088 ± 0.095	0.168 ± 0.147
10	40	0.037 ± 0.048	0.094 ± 0.102
10	60	0.026 ± 0.039	0.085 ± 0.125
10	80	0.006 ± 0.009	0.089 ± 0.138

Nodes—number of nodes of the input FCM; density (%)—ratio of non-zero weights to the total number of weights; error_initial, error_behavior—average value of corresponding evaluation criteria; stdev—standard deviation of the corresponding reported criterion.

- The bar plots show the error, in terms of an absolute difference, between corresponding state vector values generated by the input and the candidate FCMs.

The following conclusions can be drawn based on analysis of the above figures:

- The error_initial values were relatively low in all considered cases, which confirms the overall results. In one case, the candidate FCM was able to perfectly recover the input data, which resulted from its identical connection matrix when compared to the connection matrix of the input FCM, see Fig. 9a and b.
- In general, better results, in terms of recovering the input data, were obtained for cases, in which the sequence in the input data heads for the fixed-point attractor, see Figs. 9a and 10a. In such cases, the errors decrease with the simulation time and stabilize at certain, low level. On the other hand, models that head for limit cycle turned out to have higher error rate, see Figs. 9e and 10e.
- Simulation of input and candidate FCMs for experiments with new initial state vectors again show that fixed-point attractor sequences give better learning results when compared to limit cycle sequences. We observed that in general FCM models have an inclination to head for the same fixed-point attractor vector value regardless of the value of the initial state vector value, which was also reported in [37]. Therefore the results for the fixed-point attractor sequences perform on average better; compare Figs. 9d and h, and 10d and h.

Table 6 shows summary of the results for all performed experiments, considering different number of concept nodes and densities.

To easy the analysis of the results, a relation between FCM parameters, i.e. size and density, and each of the evaluation criteria was presented in Figs. 11 and 12. Each of these figures includes a table with the

		Number of nodes			
		4	6	8	10
Density	20%	0.000	0.005	0.057	0.088
	40%	0.000	0.005	0.015	0.037
	60%	0.000	0.004	0.014	0.026
	80%	0.000	0.003	0.006	0.006
	baseline	0.391			

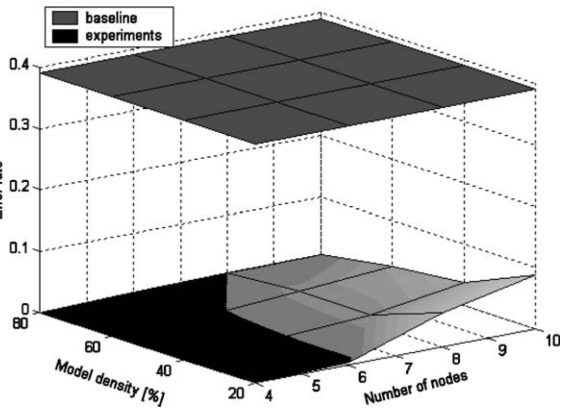


Fig. 11. Error_initial values as a function of number of nodes and FCM density.

		Number of nodes			
		4	6	8	10
Density	20%	0.001	0.017	0.102	0.147
	40%	0.000	0.018	0.052	0.102
	60%	0.000	0.022	0.056	0.125
	80%	0.000	0.011	0.036	0.130
	baseline	0.391			

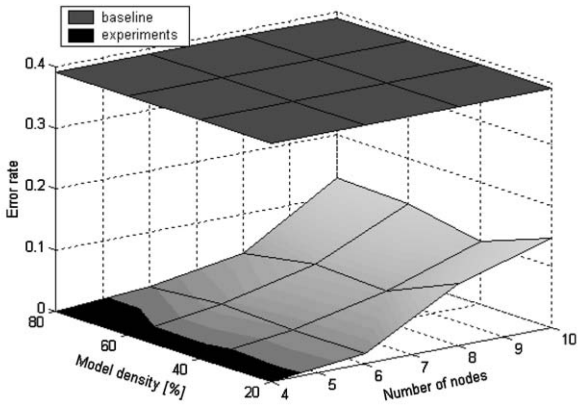


Fig. 12. Error_behavior values as a function of number of nodes and FCM density.

average values of the corresponding criterion across different sizes and densities of the input FCMs. The content of the tables is also represented as graphs, which additionally include the baseline results.

The error_initial values, which describe the error between the input data and state vector sequence generated by the candidate FCM, are relatively small for all considered experiments. We note that they slightly increase with the increasing size and decreasing density of the input FCM, but even for the 10 nodes and 30% dense FCM the value indicates finding high-quality candidate FCM. Since this criterion was used to develop fitness function of the RCGA algorithm, low values indicate that the learning method is effective in finding a good quality solution. In general, this result indicates that the proposed learning method is able to find FCM that can closely mimic the input data for the same initial state vector.

The second criterion, i.e. error_behavior, shows how the RCGA algorithm copes with preventing overfitting the initial data, or in another words with finding a well-generalized solution, i.e. it finds FCM that fits the input data well, but also behaves similarly to the input FCM for different input state vectors. High values of this criterion would indicate poor quality of the candidate FCM. Compared to the values of the error_initial criterion, the rate of deterioration of error_behavior values is also linear, but it is more rapid. We note that error_behavior values increase with the increasing size of the input FCM, and are approximately constant for different densities. For some experiments, slightly better results were obtained for denser FCM models. This indicates that the quality of the learning method deteriorates with the increasing size of the maps, but is almost independent on the density. Comparison of the achieved results with respect to the baseline values reveals that the proposed learning algorithm generates candidate FCMs with good quality. We observe that the RCGA algorithm generates very high quality candidate FCM for problems involving 4 and 6 concept nodes, while the quality for larger problems is still acceptable. In general, we conclude that the experimental results proved usefulness of the genetic algorithm based learning for this problem.

4.2.3. Real-life data

These tests were performed with two FCM models that have been reported in literature. Larger models, with 7 and 10 concept nodes, were selected to show quality of the proposed learning method for harder domains. In this case the FCM was predefined by the original author (domain expert). The experiments involved simulating the input FCM to generate input data, and later using the input data to generate a candidate FCM. Next, the error_initial was computed for the candidate FCM, and similarly as for the synthetic data, 10 experiments with new, randomly generated state vectors were performed to compute the error_behavior value.

Two experiments with real-life FCM were performed. First experiment involves a 7 nodes map, while the other a 10 nodes map.

4.2.3.1. Experiment 1—e-business company: First experiment was performed with FCM model proposed by Tsadiras, which concerns business industry and financial activities [44]. This FCM describes relationships among seven concepts, which were identified as important in the strategic planning process of a e-business company. The following concepts were considered: e-business profits, e-business sales, prices cutoffs, customer satisfaction, staff recruitments, impact from international e-business competition, and better e-commerce services. The density of the considered FCM was 40%. The input data for learning, shown in Fig. 13, was generated with a randomly chosen initial state vector, and reaches the fixed-point attractor state after 10 iterations. The input data was applied to the proposed RCGA learning algorithm, resulting in a learning progress shown in Fig. 14.

The learning results were evaluated identically as in case of the experiments with the synthetic data. The error_initial ($\pm$stdev) value was 0.004 ($\pm$0.005), while error_behavior($\pm$stdev) value was 0.01 ($\pm$0.02).

Following, selected results achieved by the RCGA algorithm, in terms of comparison of state vector sequences and error defined by the difference between the input data and the generated sequence, for the e-business FCM, are presented; see Fig. 15. They follow the same rules, as defined for the experiments with the synthetic data.

The following conclusions can be drawn based on analysis of the results of this experiment:

- The quality of the candidate FCM is very good in terms of both criteria, despite that the input data length was shorter than assumed for synthetic data experiments.
- Results from simulation with new initial state vector show that for certain cases the results obtained from the candidate FCM simulation are almost identical with those from original FCM. For others, some differences are present in several initial states, but the system stabilizes in the same state.

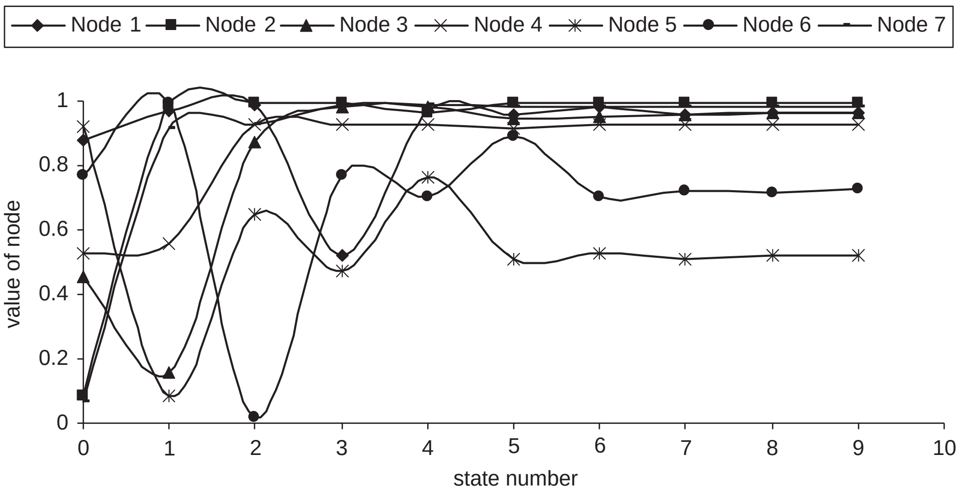


Fig. 13. Input data for e-business company FCM.

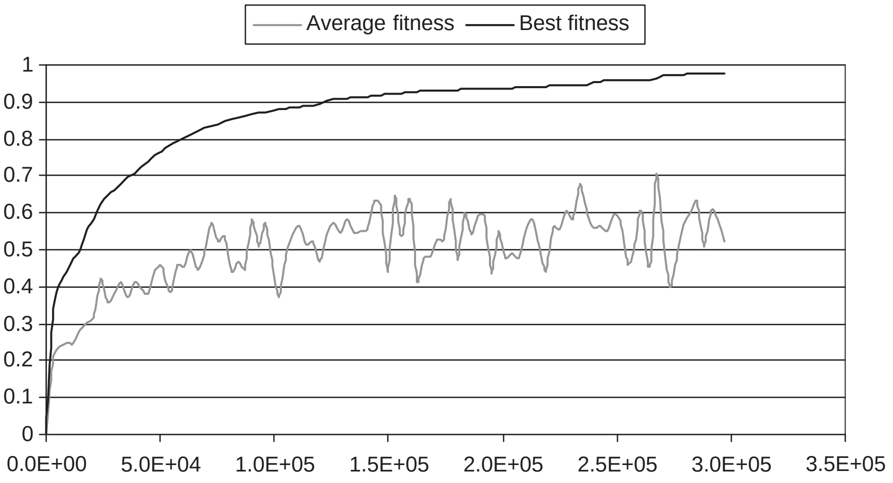


Fig. 14. Learning process for the e-business company FCM.

4.2.3.2. Experiment 2—squad of soldiers in combat: Second experiment was performed with a FCM model proposed by Kosko, which describes squad of soldiers in combat, for details see [19]. The maps included 10 nodes, which describe behavior of soldiers: cluster, proximity of enemy, receiving fire, presence of authority, firing weapons, peer visibility, spread out, taking cover, advancing, and fatigue. The density of the considered FCM was 34%. The input data for learning, shown in Fig. 16, was generated with a randomly chosen initial state vector, and reaches the fixed-point attractor state after 12 iterations.

The input data was applied to the proposed RCGA learning algorithm, resulting in a learning progress shown in Fig. 17.

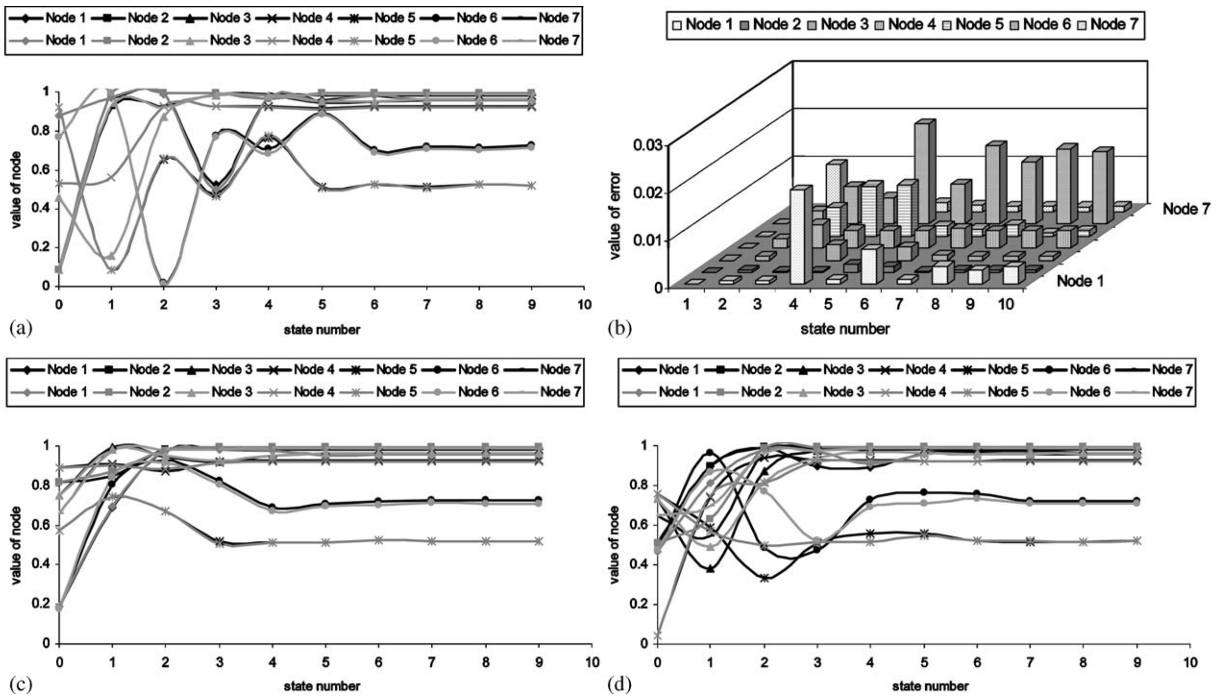


Fig. 15. Selected results achieved for the e-business FCM: (a) Input data and plot obtained from simulation of candidate FCM; (b) Difference between corresponding states vector values for plots from (a); (c) Comparison of the best simulation of candidate and input FCMs with new state vector; (d) Comparison of the worst simulation of candidate and input FCMs with new state vector.

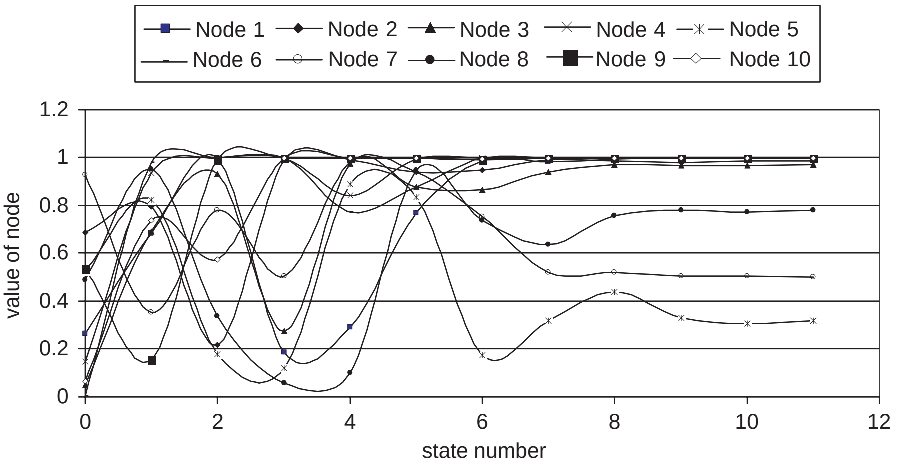


Fig. 16. Input data for squad of soldiers FCM.

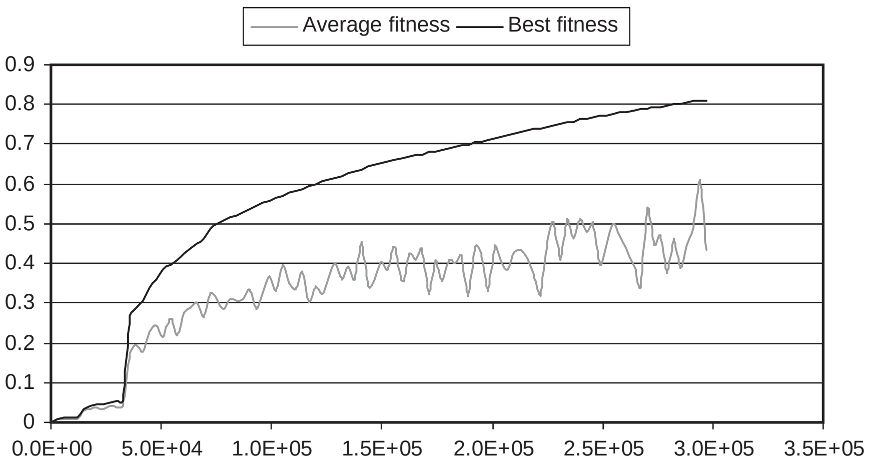


Fig. 17. Learning process for the squad of soldiers FCM.

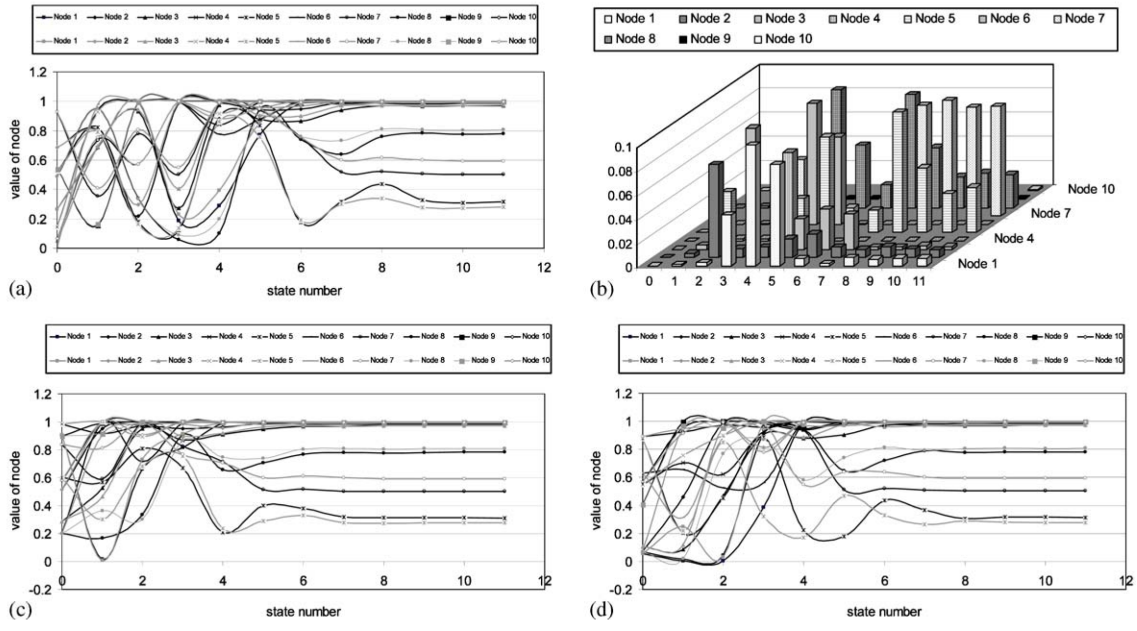


Fig. 18. Selected results achieved for the squad of soldiers in combat FCM: (a) Input data and plot obtained from simulation of candidate FCM for squad of soldiers FCM; (b) Difference between corresponding states vector values for plots from (a); (c) Comparison of the best simulation of candidate and input FCMs with new state vector for squad of soldiers FCM; (d) Comparison of the worst simulation of candidate and input FCMs with new state vector for squad of soldiers FCM.

For the experiments with the squad of soldiers in combat FCM, the error_initial(±stdev) value was 0.02 (±0.03), while error_behavior (±stdev) value was 0.04 (±0.09). Other selected results, similarly as in case of experiment 1, are presented in Fig. 18.

On the basis of the experiment with squad of soldiers FCM the following conclusions can be drawn:

- Quality of results is good, but slightly worse than those for the e-business FCM. The reason is greater number of nodes, which implies more complex models, and agrees with results for synthetic data.
- Generalization of model expressed by examining similarity of plots obtained from new initial vectors is satisfactory; visible differences occur only in few initial states, but the system stabilizes in the same state;

Experiments performed with real-life data were focused on examining the proposed learning approach when dealing with real systems reported in literature. The results obtained from simulations show high usefulness of this learning method. The quality of candidate FCM, which is expressed by two error values, is comparable with the corresponding results obtained from synthetic data.

5. Conclusions and further directions

In this study, we have developed a comprehensive learning environment for the development of fuzzy cognitive maps. It has been shown how genetic optimization helps construct maps on a basis of experimental numeric data. We demonstrated the feasibility and effectiveness of the evolutionary approach.

The paper discusses relevant work, and proposes and tests a novel learning strategy, based on a real-coded genetic algorithm. The method is able to generate a FCM model from input data consisting of a single sequence of state vector values. First set of experiments aimed to design high-quality genetic-algorithm-based learning strategy based on different fitness functions. In order to achieve this goal, three different functions were examined, and the best among them was selected. Later, a comprehensive set of tests was performed with the selected function, including experiments with both synthetic and real-life data, and different sizes and densities of FCMs. The results show that the proposed learning method is very effective, and generates FCM models that can almost perfectly represent the input data. We note that the quality of learning deteriorates with the increasing size of the maps. In general, the proposed method achieved excellent quality for maps up to 6 nodes, while for maps up to 10 nodes that quality is still satisfactory. Since many different configurations of FCMs have been tested, the produced results could provide some guidelines for other learning methods.

The future work will concern on the improvement of the proposed learning method, especially in terms of its scalability (computational complexity) and convergence. One of interesting and open issues worth pursuing would be to associate parameters of the genetic algorithm with the characteristics of a given experimental data. Another interesting direction concerns the use of the learning method in a context of practical applications. Those could involve such areas as, e.g., stock exchange or sports bets.

Acknowledgements

The authors would like to thank anonymous reviewers for constructive comments. This research was supported in part by the Natural Sciences & Engineering Research Council of Canada (NSERC).

References

[1] J. Aguilar, Adaptive random fuzzy cognitive maps, Proc. 8th Ibero-American Conf. on AI, 2002, pp. 402–410.

[2] J. Aguilar, A dynamic fuzzy-cognitive-map approach based on random neural networks, Internat. J. Comput. Cognition 1 (4) (2003) 91–107.

[3] J. Aguilar, A survey about fuzzy cognitive maps papers (Invited Paper), Internat. J. Comput. Cognition 3 (2) (2005) 27–33.

[4] R. Axelrod, Structure of Decision: The Cognitive Maps of Political Elites, Princeton University Press, Princeton, NJ, 1976.

[5] C. Carlsson, R. Fuller, Adaptive fuzzy cognitive maps for hyperknowledge representation in strategy formation process, Proc. Internat. Panel Conf. on Soft and Intelligent Computing, 1996, pp. 43–50.

[6] K. Deb, An introduction to genetic algorithms, Sadhana 24 (4) (1999) 205–230.

[7] J.A. Dickerson, B. Kosko, Fuzzy virtual worlds, Artif. Intel. Expert 7 (1994) 25–31.

[8] J.A. Dickerson, B. Kosko, Virtual worlds as fuzzy cognitive maps, Presence 3 (2) (1994) 173–189.

[9] V.C. Georgopoulos, G.A. Malandraki, C.D. Stylios, A fuzzy cognitive map approach to differential diagnosis of specific language impairment, J. Artif. Intel. Med. 29 (3) (2003) 261–278.

[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[11] K. Gotoh, J. Murakami, T. Yamaguchi, Y. Yamanaka, Application of fuzzy cognitive maps to supporting for plant control, Proc. SICE Joint Symp. 15th Systems Symp. and Tenth Knowledge Engineering Symp., 1989, pp. 99–104.

[12] F. Herrera, M. Lozano, J.L. Verdegay, Tackling real-coded genetic algorithms: operators and tools for behavioural analysis, Artif. Intel. Rev. 12 (4) (1998) 265–319.

[13] D. Kardaras, G. Mentzas, Using fuzzy cognitive maps to model and analyse business performance assessment, in: J. Chen, A. Mital (Eds.), Advances in Industrial Engineering Applications and Practice II, 1997, pp. 63–68.

[14] M.S. Khan, A. Chong, Fuzzy cognitive map analysis with genetic algorithm, Proc. 1st Indian Internat. Conf. on Artificial Intelligence (IICAI-03), 2003

[15] M. Khan, M. Quaddus, Group decision support using fuzzy cognitive maps for causal reasoning, Group Decision Negotiation J. 13 (5) (2004) 463–480.

[16] B. Kosko, Fuzzy cognitive maps, Internat. J. Man-Mach. Studies 24 (1986) 65–75.

[17] B. Kosko, Hidden patterns in combined and adaptive knowledge networks, Internat. J. Approx. Reason. 2 (1988) 377–393.

[18] B. Kosko, Neural Networks and Fuzzy Systems, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[19] B. Kosko, Fuzzy Engineering, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[20] D.E. Koulouriotis, I.E. Diakoulakis, D.M. Emiris, Anamorphosis of fuzzy cognitive maps for operation in ambiguous and multi-stimulus real world environments, 10th IEEE Internat. Conf. on Fuzzy Systems, 2001, pp. 1156–1159.

[21] D.E. Koulouriotis, I.E. Diakoulakis, D.M. Emiris, Learning fuzzy cognitive maps using evolution strategies: a novel schema for modeling and simulating high-level behavior, IEEE Congr. on Evolutionary Computation (CEC2001), 2001, pp. 364–371.

[22] D.E. Koulouriotis, I.E. Diakoulakis, D.M. Emiris, E.N. Antonidakis, I.A. Kaliakatsos, Efficiently modeling and controlling complex dynamic systems using evolutionary fuzzy cognitive maps (Invited Paper), Internat. J. Comput. Cognition 1 (2) (2003) 41–65.

[23] K.C. Lee, W.J. Lee, O.B. Kwon, J.H. Han, P.I. Yu, Strategic planning simulation based on fuzzy cognitive map knowledge and differential game, Simulation 71 (5) (1998) 316–327.

[24] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, 1992.

[25] S.T. Mohr, The use and interpretation of fuzzy cognitive maps, Master's Project, Rensselaer Polytechnic Institute, 1997.

[26] H. Mühlenbein, D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm I. continuous parameter optimization, Evol. Comput. 1 (1) (1993) 25–49.

[27] T.D. Ndousse, T. Okuda, Computational intelligence for distributed fault management in networks using fuzzy cognitive maps, Proc. IEEE Internat. Conf. Communications Converging Technologies for Tomorrow's Application, 1996, pp. 1558–1562.

[28] E.I. Papageorgiou, K.E. Parsopoulos, C.D. Stylios, P.P. Groumpos, M.N. Vrahatis, Fuzzy cognitive maps learning using particle swarm optimization, J. Intel. Inform. Systems, in press.

[29] E. Papageorgiou, C.D. Stylios, P.P. Groumpos, Fuzzy cognitive map learning based on nonlinear Hebbian rule, Australian Conf. on Artificial Intelligence, 2003, pp. 256–268.

[30] E. Papageorgiou, C.D. Stylios, P.P. Groumpos, Active Hebbian learning algorithm to train fuzzy cognitive maps, Internat. J. Approx. Reason. 37 (3) (2004) 219–249.
[31] K.E. Parsopoulos, E.I. Papageorgiou, P.P. Groumpos, M.N. Vrahatis, A first study of fuzzy cognitive maps learning using particle swarm optimization, Proc. IEEE 2003 Congr. on Evolutionary Computation, 2003, pp. 1440–1447.
[32] C.E. Pelaez, J.B. Bowles, Applying fuzzy cognitive maps knowledge representation to failure modes effects analysis, Proc. IEEE Annu. Symp. on Reliability and Maintainability, 1995, pp. 450–456.
[33] S. Renals, R. Rohwer, A study of network dynamics, J. Statist. Phys. 58 (1990) 825–848.
[34] T.L. Saaty, The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation, McGraw-Hill, New York, 1980.
[35] M. Schneider, E. Shnaider, A. Kandel, G. Chew, Automatic construction of FCMs, Fuzzy Sets and Systems 93 (2) (1998) 161–172.
[36] A. Siraj, S. Bridges, R. Vaughn, Fuzzy cognitive maps for decision support in an intelligent intrusion detection system, IFSA World Congr. and 20th NAFIPS Internat. Conf., vol. 4, 2001, pp. 2165–2170.
[37] W. Stach, L. Kurgan, Modeling software development project using fuzzy cognitive maps, Proc. 4th ASERC Workshop on Quantitative and Soft Software Engineering (QSSE'04), 2004, pp. 55–60.
[38] W. Stach, L. Kurgan, W. Pedrycz, M. Reformat, Parallel fuzzy cognitive maps as a tool for modeling software development project, Proc. 2004 North American Fuzzy Information Processing Society Conf. (NAFIPS'04), Banff, AB, 2004, pp. 28–33.
[39] M.A. Styblinski, B.D. Meyer, Signal flow graphs versus fuzzy cognitive maps in application to qualitative circuit analysis, Internat. J. Man Mach. Studies 35 (1991) 175–186.
[40] C.D. Stylios, P.P. Groumpos, The challenge of modelling supervisory systems using fuzzy cognitive maps, J. Intel. Manuf. 9 (4) (1998) 339–345.
[41] C.D. Stylios, P.P. Groumpos, Fuzzy cognitive maps: a model for intelligent supervisory control systems, Comput. Ind. 39 (3) (1999) 229–238.
[42] C.D. Stylios, P.P. Groumpos, Fuzzy cognitive map in modeling supervisory control systems, J. Intel. & Fuzzy Systems 8 (2) (2000) 83–98.
[43] C.D. Stylios, P.P. Groumpos, Modeling complex systems using fuzzy cognitive maps, IEEE Trans. Systems Man, Cybern. Part A: Systems Humans 34 (1) (2004).
[44] A.K. Tsadiras, Using fuzzy cognitive maps for e-commerce strategic planning, Proc. 9th Panhellenic Conf. on Informatics (EPY' 2003), 2003.
[45] A.K. Tsadiras, K. Margaritis, An experimental study of the dynamics of the certainty neuron fuzzy cognitive maps, Neurocomputing 24 (1999) 95–116.
[46] A. Vazquez, A balanced differential learning algorithm in fuzzy cognitive maps, Technical Report, Departament de Llenguatges I Sistemes Informatics, Universitat Politecnica de Catalunya (UPC), 2002.
[47] A. Wright, Genetic algorithms for real parameter optimization, Foundations of Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1991, pp. 205–218.