Learning of Fuzzy Cognitive Maps Using Density Estimate

Wojciech Stach, *Member, IEEE*, Witold Pedrycz, *Fellow, IEEE*, and Lukasz A. Kurgan, *Member, IEEE*

***Abstract*—Fuzzy cognitive maps (FCMs) are convenient and widely used architectures for modeling dynamic systems, which are characterized by a great deal of flexibility and adaptability. Several recent works in this area concern strategies for the development of FCMs. Although a few fully automated algorithms to learn these models from data have been introduced, the resulting FCMs are structurally considerably different than those developed by human experts. In particular, maps that were learned from data are much denser (with the density over 90% versus about 40% density of maps developed by humans). The sparseness of the maps is associated with their interpretability: the smaller the number of connections is, the higher is the transparency of the map. To this end, a novel learning approach, sparse real-coded genetic algorithms (SRCGAs), to learn FCMs is proposed. The method utilizes a density parameter to guide the learning toward a formation of maps of a certain predefined density. Comparative tests carried out for both synthetic and real-world data demonstrate that, given a suitable density estimate, the SRCGA method significantly outperforms other state-of-the-art learning methods. When the density estimate is unknown, the new method can be used in an automated fashion using a default value, and it is still able to produce models whose performance exceeds or is equal to the performance of the models generated by other methods.**

***Index Terms*—Fuzzy cognitive maps (FCMs), real-coded genetic algorithms (RCGAs).**

I. Introduction

FUZZY COGNITIVE maps (FCMs), introduced by Kosko [16], have found applications in modeling and simulation of dynamic systems. They represent a given system as a collection of concepts (events, actions, values, goals, etc.) represented by cause–effect dependencies. Any FCM is depicted as a fuzzy causal graph [16], in which nodes represent concepts, whereas directed edges between the concepts denote causal relationships present between them. The graph structure of FCMs allows for *static analysis*, while its execution model allows for *dynamic analysis* of the modeled system [46]. FCMs are convenient in handling issues of knowledge representation and reasoning, which are essential to intelligent systems [22]. This modeling technique comes with a number of desirable properties, such as abstraction, flexibility, adaptability, and fuzzy reasoning [2], [44]. The areas of applications of FCMs are diversified and include problems in electrical engineering, medicine, political science, international relations, etc. [2]. Examples of specific applications include medical diagnosis [14], analysis of electrical circuits [41], failure modes effects analysis [29], fault management in distributed network environment [24], modeling of software development projects [33], and many others.

One of the recently extensively researched topics in FCMs concerns their design process [2], [39]. Two main categories of the design have been investigated. *Expert-based methods*, which originate from *deductive modeling*, are performed manually and rely solely on human expert knowledge. Their main drawbacks are model subjectivity and limited human perception and ensuing quantification of perception when it comes to dealing with complex systems [39]. As an alternative approach, *computational methods* have been recently introduced to support (*semiautomated* methods) or to replace (*fully automated* methods) human expert(s) [2], [39]. These methods are aimed at developing a model from available data using some learning mechanisms. Therefore, they are also referred to as *inductive modeling* techniques. The two major types of the computational methods include the following: 1) Hebbian-based methods, which were proposed shortly after the introduction of FCMs, and 2) evolutionary algorithm-based methods, which are currently gaining momentum [35], [39]. The computational methods face a substantial challenge, which is the existence of numerous suboptimal solutions. These solutions provide similar simulation results (dynamic analysis), while they might be structurally different and, subsequently, not suitable to support static analysis [2]. This problem is partially addressed by semiautomated methods, in which experts impose constraints on certain properties of the FCM, like the weight associated with the individual causal relationships. However, an involvement of human experts exposes the model to the same problems as in the case of expert-based development methods. Therefore, while this type of learning can be efficient in certain applications, semiautomated methods cannot be considered as a silver-bullet solution, e.g., they could not be used to model large maps due to the difficulty in establishing the corresponding large number of constraints.

The motivation of this paper comes from structural comparison of models generated by existing fully automated learning approaches against real-world models developed by human experts. We show that maps developed by computational methods are much denser than those constructed by humans. Table I summarizes the values of densities, defined as the ratio of the number of nonzero weights to the total number of all possible weights for a given map size (number of nodes in the graph), of the FCMs reported in the literature.

Manuscript received January 10, 2010; revised September 18, 2010 and July 19, 2011; accepted December 6, 2011. Date of publication February 14, 2012; date of current version May 16, 2012. This work was supported in part by the Alberta Ingenuity, the Alberta Informatics Circle of Research Excellence (iCORE), and the Natural Sciences & Engineering Research Council of Canada (NSERC). This paper was recommended by Associate Editor M. Huber.

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: wstach@ualberta.ca; lkurgan@ece.ualberta.ca; wpedrycz@ualberta.ca).

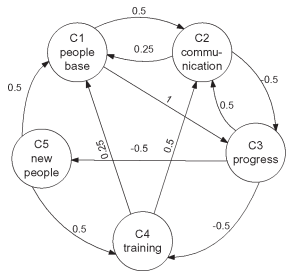
Digital Object Identifier 10.1109/TSMCB.2011.2182646

TABLE I
EXAMPLES OF FCMS REPORTED IN LITERATURE ALONG WITH THE NUMBER OF NODES (#NODES) AND DENSITY OF CONNECTIONS (DENSITY)

Reference	Application area	# nodes	Density
[1]	social science (model of a country)	5	35%
[45]	engineering (process control)	5	40%
[33]	engineering (software development)	5	44%
[44]	engineering (heat exchanger)	5	50%
[19]	engineering (industrial control)	5	60%
[44]	engineering (supervisor model for heat exchanger performance)	5	75%
[30]	engineering (evidence of multiple suspicious events)	6	20%
[23]	engineering (EMU taxes and transfers model)	6	27%
[8]	social science (strategy formation model)	6	33%
[20]	social science (public health issues)	7	24%
[23]	social science (crime and punishment model)	7	36%
[46]	engineering (e-business company)	7	40%
[43]	engineering (system for direct control of a process)	8	23%
[23]	engineering (EMU and the risk of war model)	8	30%
[42]	engineering (plant supervisor model)	9	25%
[17]	engineering (virtual squad of soldiers)	10	34%
[49]	engineering (military planning)	12	38%
[4]	engineering (slurry rheology)	13	39%

Most of the models are characterized by a relatively low level of density, in the range of 30%–40%. The average model density from Table I is approximately 37%, with a standard deviation of 14%. We use the map concerning the slurry rheology [4], which is a larger size map with similar to average density, to demonstrate the level of densities obtained by application of several learning methods. In this experiment, we simulated the expert-derived model to obtain data that were next used in the learning, which, in turn, would try to reproduce the input FCM model. We performed the learning five times, and the average densities of the generated maps are equal 95% for the real-coded genetic algorithm (RCGA) method [35], 93% for the nonlinear Hebbian learning (NHL) method [26], and 92% for the data-driven NHL (DD-NHL) method [37] (these methods are described in Section II-A).

We propose a learning method, *sparse RCGA* (SRCGA), using which we develop sparse FCMs. It incorporates a parameter (*density estimate*) to guide the learning process and utilizes a modified RCGA algorithm to implement the process. The choice of the RCGA method was implied by the nature of the learning problem, which involves a search through a large solution space with continuous variables and many suboptimal solutions. Moreover, learning FCMs with the RCGA approach has been shown to outperform other fully automated methods [34]. The basic RCGA method was modified to accommodate the *density estimate* parameter. Comparative experimental analysis that involves three other learning methods shows that, if an accurate density estimate is available, the SRCGA method learns models of high transparency in terms of both static and dynamic properties. Analysis of results demonstrates that the differences are statistically significant. The results also



	C1	C2	C3	C4	C5
C1	0	0.5	1	0	0
C2	0.25	0	-0.5	0	0
C3	0	0.5	0	-0.5	-0.5
C4	0.25	0.5	0	0	0
C5	0.5	0	0	0.5	0

Fig. 1. Example of FCM that models a software development project. (a) FCM graph. (b) Connection matrix.

demonstrate that the quality of the generated FCM model is lower in the absence of a proper density estimate; however, it is still better or equal to the quality of models learned by the other fully automated methods.

This paper is organized as follows. Section II presents a required background on FCMs and a summary of the existing FCM learning approaches. It also includes a description of the RCGAs, which are used as the proposed learning method. Furthermore, in Section III, we present detailed description of the SRCGA. Section IV covers experimental results for synthetic and real-world systems. Finally, key findings are summarized in Section V.

II. BACKGROUND

A. Overview of FCMs

The techniques for modeling and analysis of dynamic systems can be divided into two major groups, namely, quantitative and qualitative approaches [7]. FCMs, which are successors and generalizations of cognitive maps [3], fall into the rubric of qualitative modeling and are characterized by simplicity of both model representation and its execution. They define a given system as a collection of concepts that influence each other through cause–effect relationships, which are quantified and usually normalized to the $[-1, 1]$ interval [16]. Positive values describe promoting effect, whereas negative ones describe inhibiting effect. The value of -1 represents the highest negative interaction, and $+1$ represents the highest positive interaction, while 0 denotes a neutral relationship (no interaction). Other values correspond to different intermediate levels of the causal effect. The term "fuzzy" in FCMs refers to the relationship values, which are "fuzzified" when compared to the values present in cognitive maps [16].

FCMs can be conveniently represented either by a graph, which is easily understood by a human, or by a connection matrix, which is useful for computational purposes when running simulations. In the graph representation, concepts are represented as nodes, and relationships are depicted by directed edges between the nodes. Each edge is also associated with a number (weight) that quantifies the strength of the corresponding relationship. In the matrix representation, concepts are represented by successive rows/columns, and cells store all relationships values. Fig. 1 shows both of these equivalent representations using an example model of a software development project [33].

The two main approaches to develop FCMs are based on the deductive modeling (expert-based methods) and the inductive

modeling (computational methods) [39]. The expert-based methods exploit human domain knowledge. In this case, the models are developed based solely on the understanding of the modeled system by an expert or a group of experts. These methods suffer from a number of drawbacks. They require a knowledgeable expert who is also familiar with the FCM formalism. In addition, the model is vulnerable to the bias exhibited by the expert, especially in terms of an accurate weight assignment. The FCM development by a group of experts may increase the model's reliability; however, additional parameters, such as credibility of each expert, need to be considered, which adds to the complexity of the entire process. Also, development of larger maps is difficult as the number of weighs that have to be established exhibits quadratic growth with the size of the problem. These problems motivated investigations into alternative methods for the development of FCMs. A number of learning techniques, which aim at supporting or replacing human expert(s), have been recently examined. They use available historical data and an algorithm to develop a model of a given system. Section II-B presents a comprehensive overview of the existing learning methods.

Once the map has been developed (either by an expert or through learning), a user can perform *static* analysis of the model using graph theory techniques (as adopted by Tsadiras [48]). The FCM may also be used to complete simulations and, based on them, draw conclusions as to the *dynamic* behavior of the system. Simulation boils down to calculating system states over successive iterations. System state is defined by the degree of activation of all the concepts, which is usually limited to the [0, 1] interval [16]. The value of zero suggests that a given concept is not present in the system at a particular iteration, whereas the value of one indicates that a given concept is present to its maximum degree. Other values correspond to intermediate levels of activation. The activation level of each concept depends on its value at the preceding iteration as well as on the preceding values of all concepts that exert influence on it through nonzero relationships. Consequently, the simulation requires knowledge of an *initial state vector* in order to determine successive states of the model, which is carried out using the following expression [2], [16]

$$C_j(t+1) = f\left(\sum_{i=1}^{N} e_{ij} C_i(t)\right) \tag{1}$$

where $C_i(t)$ is the value of the ith node in the tth iteration, e_{ij} is the edge weight between nodes C_i and C_j, N is the number of nodes, and f is a transformation function.

Therefore, expression (1) is used to calculate each concept activation degree over the successive iterations. The transformation function is used to restrict the unbounded weighted sum to a certain range. This hinders quantitative analysis but allows for more qualitative comparative analysis. The most commonly encountered functions are continuous; however, some authors utilize binary functions. The latter type of functions limits dynamic analysis of concepts just to two values, which correspond to linguistic terms: inactive and active. A comparison of different transformation functions for FCMs was carried out in two recent papers authored by Tsadiras [47] and by Bueno and Salmeron [6]. Fig. 2 shows a sample simulation of the model

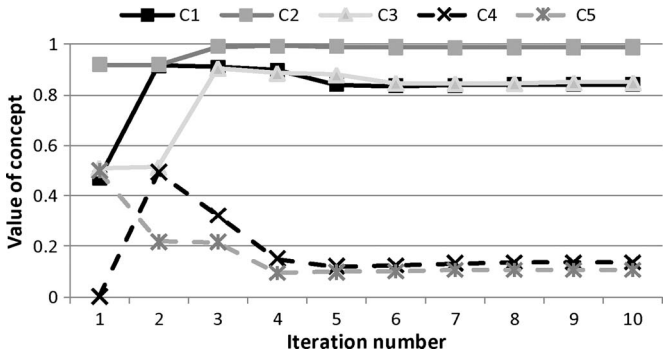


Fig. 2. Sample FCM simulation of the model from Fig. 1.

present in Fig. 1. The simulation was carried out with the use of logistic transformation function [see formula (2)], which is the most commonly used function [2] and offers significantly greater advantages than other functions [6]. Function (2) is continuous and returns values from the (0, 1) interval that corresponds to the activation degrees of the concepts

$$f(x) = \frac{1}{1 + e^{-Cx}} \tag{2}$$

where C is a parameter used to determine the degree of fuzzification of the function. In many practical applications, its value is equal to 5 [2].

Simulations allow for an analysis of several aspects of FCMs, such as concept activation levels at the final state (if there is any) and changes/trends in the activation levels throughout the simulation concerning either all concepts or a subset of concept that is of interest to the user, and discovery of cycles (intervals and concept activation levels within the cycle). This type of analysis allows investigating "what-if" scenarios by performing simulations of a given model for different initial state vectors. Simulations offer a description of the dynamic behavior of the system that can be used to support decision making or predictions about its future states [4], [32].

B. Computational Methods for Learning FCMs

Computational methods utilize historical data available for a given system to establish an FCM model. *Semiautomated* methods require a relatively limited human intervention, whereas *fully automated* design approaches are able to develop FCMs based solely on the historical data, i.e., without any human interaction. These algorithms can be categorized into two groups based on the learning paradigm used, i.e., Hebbian-based and evolutionary algorithm-based learners [39].

Dickerson and Kosko proposed the first method that utilized a simple *differential Hebbian learning (DHL)* algorithm [9], [10], which is based on Hebbian learning. During DHL learning, the values of weights are iteratively updated until the desired structure is found. The weights of outgoing edges for each concept in the connection matrix are modified only when the corresponding concept value changes:

$$e_{ij}(t+1) = \begin{cases} e_{ij}(t) + c_t\left[\Delta C_i \Delta C_j - e_{ij}(t)\right] & if\ \Delta C_i \neq 0 \\ e_{ij}(t) & if\ \Delta C_i = 0 \end{cases} \tag{3}$$

where e_{ij} denotes the weight for relation from concept C_i to 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 coefficient. The learning coefficient is a small constant, whose values usually decrease as the learning progresses.

The main drawback of this learning method is that the formula updates weights between each pair of concepts, taking into account only these two concepts and ignoring the influence coming from other concepts. An improved version of DHL learning was introduced by Huerga [13]. The algorithm called *balanced differential algorithm (BDA)* eliminates one of the limitations of the DHL method by taking into account all the concept values that change at the same time when updating the weights. More specifically, the modified formula for $e_{ij}(t+1)$ takes into consideration changes in all concepts if they occur at the same iteration and exhibit the same direction. Experimental comparison between DHL and BDA demonstrates that the latter method improves quality of the learned maps [13]. The BDA algorithm was applied only to binary FCMs, i.e., maps with binary transformation functions, which limits its application areas.

A year later, Papageorgiou introduced an NHL algorithm [26]. While this algorithm originates from the same learning principles, it uses a nonlinear extension to the basic Hebbian rule [25] by modifying the weight update formula. The NHL learning method has been designed as a semiautomated approach that requires initial human intervention. Experts are required to suggest nodes that are directly connected, and only these edges are updated during learning. In addition, the experts have to indicate a sign of each edge according to its physical interpretation. This algorithm updates the corresponding weights while preserving their initial signs. In short, the NHL algorithm allows obtaining a model that retains initial graph structure imposed by the expert(s), and therefore, it requires human intervention before the learning process starts. In addition, the experts have to define output concepts and specify range of values that these concepts can take. The latter is used after every update of the learned model's weights to validate the model. The validation is based on checking whether the model state satisfies these constrains.

The same research group proposed active Hebbian algorithm (AHL) in 2004 [27]. This approach realizes the task of determination of the sequence of activation concepts. Expert(s) determines a desired set of concepts, initial structure, and interconnections of the FCM structure, as well as the sequence of activation concepts. A seven-step AHL procedure, which is based on Hebbian learning, is iteratively used to adjust the weights until predefined stopping criteria are satisfied.

In the recent work, Stach et al. proposed an improved version of the NHL method [37]. The algorithm, called DD-NHL, is based on the same learning principle as NHL, but it takes advantage of historical data (a simulation of the actual system) and uses output concepts to improve the learning quality. An empirical comparative study has shown that, if historical data are available, then the DD-NHL method produces better FCM models when compared with those developed by using the generic NHL method [37].

In 2001, Koulouriotis applied the genetic strategy to learn the FCM's model structure from the data [18]. In their method, the learning process is based on a collection of input/output pairs, referred to as *examples*. The learning requires historical data consisting of multiple sequences of state vectors (multiple simulations of the system). The algorithm computes the structure of the FCM that is able to generate state vector sequences that transform the input vectors into the output vectors. The main drawback of this approach is that it requires multiple state vector sequences, which might be difficult to obtain in some of the application domains.

Particle swarm optimization (PSO) method, proposed by Parsopoulos, belongs to the class of swarm intelligence algorithms [28]. This method aims at learning the FCM structure based on historical data that converge to a desired final state. The PSO is a population-based algorithm, which performs a search for the solution by maintaining and transforming a population of individuals. The learning requires human knowledge to specify adequate constraints, which would guarantee that the relationships within the FCM model retain the physical meaning defined by the expert(s).

The algorithm proposed by Khan and Chong aims to accomplish a different learning objective [15]. Instead of learning the structure of the FCM model, their goal was to find an initial state vector (initial condition) that leads a given model to the specified end state. Their method employed genetic algorithms (GAs) to find the initial state.

A fully automated method for learning FCMs, which is based on RCGAs, was introduced by Stach in 2005 [35]. The RCGA is a floating-point extension [12] to GAs [11]. This extension was used to allow finding of the floating point weights instead of weights that take on a limited set of values. The core of this approach is a learning module which exploits RCGA to find the FCM structure that is capable of mimicking the given input historical data. This approach is flexible in terms of the input data as it can use either one or multiple sets of concept values over successive iterations. A follow-up of this work includes analysis of the quality of the RCGA-based learning depending on the amount of the available historical data [36]. It demonstrates that the RCGA-based method can generate FCM models that are identical to models proposed by a domain expert, given the training data of sufficient size.

Recently, the same research group has introduced a parallel RCGA-based method that targets learning of large maps that consist of dozens of concepts [34]. The method was reported to be up to four times faster than the sequential RCGA learning when executed on eight processors, and it allows learning maps that include several dozens of concepts within a few hours. This publication was followed by another method that aims at improving learning time. This work is based on the divide and conquer strategy, where the input data are divided into several subsets that are used simultaneously to learn submodels, which are later merged into a single final model [38].

Table II summarizes the existing computational methods for learning FCMs. It includes several aspects, such as learning goal, involvement of a domain expert, input data type, and learning strategy used.

A more detailed comparison of expert-based methods with computational methods for learning FCMs is presented in a recent work [40]. It concludes that computational methods are more suitable to perform accurate dynamic analysis of a given system, whereas expert-based methods provide a more accurate

TABLE II
COMPARISON OF INDUCTIVE METHODS USED IN LEARNING OF FCMS. THE "NO. OF NODES" COLUMN REPORTS MAP SIZES THAT HAVE BEEN LEARNED USING A GIVEN METHOD AND REPORTED IN THE ORIGINAL REFERENCE, WHEREAS THE "LEARNING ALGORITHM" COLUMN SHOWS A CATEGORY OF THE LEARNING METHOD THAT A GIVEN TECHNIQUE BELONGS TO

Algorithm	Reference	Learning objective	Human input	Type of data used[1)]	Transformation function	No. of nodes	Learning algorithm
DHL	[9][10]	Connection matrix	No	Single	N/A	N/A	Hebbian
BDA	[13]	Connection matrix	No	Single	Binary	5, 7, 9	Modified Hebbian
NHL	[26]	Connection matrix	Yes and No[2)]	Single	Continuous	5	Modified Hebbian
DD-NHL	[37]	Connection matrix	Yes and No[2)]	Single	Continuous	5	Modified Hebbian
AHL	[27]	Connection matrix	Yes and No[2)]	Single	Continuous	8	Modified Hebbian
GS	[18]	Connection matrix	No	Multiple	Continuous	7	Genetic
PSO	[28]	Connection matrix	No	Single	Continuous	5	Swarm
GA	[15]	Initial vector	N/A	N/A	Continuous	11	Genetic
RCGA	[35]	Connection matrix	No	Single	Continuous	4, 6, 8, 10	Genetic
Parallel RCGA	[34]	Connection Matrix	No	Single	Continuous	10, 20, 40, 80	Parallel Genetic
Sparse RCGA	This paper	Connection matrix	No	Single	Continuous	5, 10, 20, 40	Modified Genetic
Divide and Conquer RCGA	[38]	Connection matrix	No	Single	Continuous	5, 8, 10, 12, 13, 20, 24, 40	Modified Genetic

[1)] Single – historical data consisting of one sequence of state vectors, Multiple – historical data consisting of several sequences of state vectors, for different initial conditions

[2)] Initial human intervention is necessary but later when applying the algorithm it is not required

static analysis. Semiautomated methods, in which experts supervise the learning, provide partial solution to this problem. However, the existing semiautomated methods for learning FCMs are not as good for the static analysis as the models obtained from the expert-based methods. They are worse in the context of the dynamic analysis when compared with the fully automated methods.

C. Figures

GAs have been used in various problem domains to perform optimization and search tasks [11]. They have numerous advantages such as a broad applicability, ease of use, and ability to find global solutions. GAs maintain a population of chromosomes that evolves over time using operators inspired by evolutionary biology, such as mutation, selection, and crossover. Each chromosome encodes a solution to a given problem, and its quality is quantified by a fitness value that comes from a fitness function. The fitness function is custom defined for a given genetic representation (problem dependent) by the designer. The GAs usually start from a randomly generated population, and they evolve it to produce subsequent generations. In each generation, a new population is formed by using genetic operators and by exploiting fitness values of the chromosomes. The idea is to produce better solutions to the problem over the successive generations. GAs succeed in application to large, complex, and poorly understood search spaces, in which classical tools are often inappropriate. The details about GAs can be found in [11] and [31].

The RCGA [12] is a floating-point extension to basic GAs. This means that, in contrary to GAs that use binary vectors, RCGA represents chromosomes as floating-point vectors. This makes RCGA more effective in tackling optimization problems with continuous variables. The genetic operators used in the GAs have also been revised in order to handle floating-point values in the RCGA method. Nevertheless, the underlying principles of these two optimization techniques are the same. A comprehensive review of the RCGAs is presented in [12].

III. PROPOSED APPROACH

The proposed method, called *SRCGA*, is a computational method for learning FCMs. It takes the *input data* and *density estimate* parameter and carries out the learning process until the final model, called *candidate FCM*, is found. Input data are values of concept activations at successive time points (iterations). For a system with N concepts and K data points, the input data form a $K \times N$ matrix, which is called *input data matrix*. Each row of this matrix, given as $\mathbf{C}(t) = [C_1(t), C_2(t), \ldots, C_N(t)]$, where $t = 1 \ldots K$, stores values of activations of the concepts at tth iteration. For the purpose of the genetic-based learning, the problem is transformed into an optimization task. The objective of this optimization is to find the best set of weights of the candidate FCM, i.e., $\mathbf{E} = [e_{11}, e_{12}, \ldots, e_{N1}, \ldots, e_{NN}]$, where e_{ij} is the relationship strength from concept C_i to concept C_j, such that they maximize a predefined quality function (*fitness function*). The quality function evaluates the candidate FCM based on its ability to approximate the input data. Hence, the learning problem boils down to determining the values of N^2 parameters that can take any value from the $[-1, 1]$ range. The optimization process in the SRCGA method is based on the modified RCGA method.

The modified RCGA method itself, similar to a generic RCGA method, has a number of parameters, which need to be fixed before running the simulations. In our experiments, they have been set up consistently with [35] and [36]. They include population size, maximum fitness value (*max fitness*), maximum number of generation (*max generation*), recombination method, recombination probability, mutation method, mutation probability, fitness function (*fitness*), and selection method. The latter parameters are discussed in the following. They are fixed for all of the experiments reported in the following discussion, and thus, they should be treated as being transparent to the user. In contrast to the generic RCGA method, the SRCGA learning needs an additional parameter—the density estimate. In case when the density is unknown, a default value of 37%, as discussed in Section I, should be used. This parameter is used to guide the learning

toward solutions with a predefined structure (candidate FCM density).

The *SRCGA* method is a six-step procedure described as follows:

STEP 1. *Initialize and evaluate population*

First, the population of chromosomes is initialized. Each gene, which represents one of the weights in a given connection matrix (represented as the chromosome), is reset to the value of zero with probability (1—*density estimate*). All remaining genes are initialized with the value chosen randomly from a uniform distribution $[a, 1]$ and are assigned the negative sign with the 50% probability. In other words, all nonzero values are chosen from the uniform distribution in the interval $[-1, -a]$ and [a, 1], while the values from [−a, a] are rounded down to 0. a is a rounding cutoff, and it is equal to 0.05, as suggested in the previous studies [35], [36]. Second, each chromosome is evaluated by assigning to it a fitness function value (STEP 5).

STEP 2. *Check stop criteria*

IF the fitness value of the best individual (candidate FCM) is

greater than *max fitness*

OR current generation is greater than *max generation*

THEN stop the learning process.

Candidate FCM is a chromosome with the highest fitness function value in the current generation.

STEP 3. *Recombine population*

Recombination is performed using crossover operation. In our experiments, we exploited a simple one-point crossover. It carries a low computational cost, yet as demonstrated through experiments [35], it effectively handles the optimization problem.

STEP 4. *Mutate population*

For each gene selected for mutation, the following procedure is applied:

First, the density of the chromosome that includes the given gene is calculated (defined as the number of nonzero genes over the total number of genes). If this number is lower than or equal to the *density estimate*, then mutation is carried out using a randomly chosen mutation operator. The random mutation, nonuniform mutation, and Mühlenbein's mutation operators are considered, and all of them are normalized such that they return numbers between -1 and 1. This is to ensure that each gene's value is a valid relationship strength of the corresponding FCM. If the chromosome's density is higher than the *density estimate*, then the gene is reset to zero. In other words, the mutation operator is dynamically adjusted each time to guide the optimization toward solutions with the predefined density. This way of modifying mutation operation has been reported as an effective approach to boost convergence of GAs [5], [21]. It allows intensifying search in certain areas of the search space.

STEP 5. *Evaluate population*

Similar to our previous works [34]–[36], the design of the fitness function in the proposed learning method takes advantage of a specific feature of the FCM theory. Specifically, at each iteration during the model simulation, the current state vector depends only on the state vector at the preceding iteration and does not depend on any other state vectors [see formula (1)]. We use this property to group each two adjacent state vectors in the input data matrix to form $K - 1$ input data pairs. Within each pair, we call the antecedent a system stimulus, whereas the decedent is called a system response. These input data pairs store information about the system's dynamics, and consequently, they are used in the fitness function definition. This approach was reported to provide better convergence when compared to a method that uses the entire data set at once [35].

The fitness function for each chromosome is calculated by performing one-step simulations of the corresponding FCM starting from each system stimulus. The state vectors obtained from simulations, which constitute FCM responses, are compared against the corresponding true system responses. The obtained error value is used to calculate the fitness value. An auxiliary function (4) is introduced to ensure that better chromosomes have a higher fitness value and to normalize the fitness value to the (0, 1] interval [34]–[36]. As a result, the following formula (4) is used to calculate the fitness function for each chromosome:

$$fitness(Error) = \frac{1}{b \cdot Error + 1} \tag{4}$$

where

$$Error = \frac{1}{(K-1) \cdot N} \sum_{t=1}^{K-1} \sum_{n=1}^{N} \left| C_n(t) - \hat{C}_n(t) \right|^2 \tag{5}$$

$\mathbf{C}(t) = [C_1(t), C_2(t), \ldots, C_N(t)]$—original system state at time point t (from input data), $\hat{\mathbf{C}}(t) = [\hat{C}_1(t), \hat{C}_2(t), \ldots, \hat{C}_N(t)]$—state of the candidate FCM obtained from performing one-step simulation from $\mathbf{C}(t-1)$ initial state vector, and b—fitness function coefficient (set to 10 000 as used in [34]–[36]).

STEP 6. *Select next population*

Two popular selection strategies, i.e., roulette wheel and tournament selection, are randomly applied. Next, *go to STEP 2*.

The SRCGA method, by using modified population initialization and mutation operators (STEP 1 and STEP 4), guides the learning toward solutions of a given density. Since the ultimate goal is to optimize the candidate FCM in terms of its dynamic properties, the fitness function (STEP 5) compares the simulation result of the candidate FCM against the input data. The overall architecture of the RCGA optimization, its parameters, and the fitness function formula are consistent with our previous works [35], [36].

We note that the SRCGA method is capable of handling missing states since the fitness function exploits pairs of consecutive states. In other words, the algorithm splits the data into input–output pairs of successive states, which are used in RCGA-based optimization. Therefore, if the input data matrix has missing rows, then in STEP 5, when calculating the *Error* value, only rows $\boldsymbol{C}(t)$ having defined $\boldsymbol{C}(t+1)$ would be used in the formula (5). In such case, the value of K in the same

formula would be updated and equal to the number of existing $\boldsymbol{C}(t) - \boldsymbol{C}(t+1)$ pairs in the input data matrix.

IV. Experiments

The experiments evaluate the quality of the *SRCGA* method on synthetic and real-world models. Similarly as in [34]–[37], we use a known FCM connection matrix to generate the input data, which are next used by the learning method to reconstruct this matrix. The evaluation was performed based on several criteria that measure both the static and dynamic properties of the generated candidate FCMs against the known model and the input data (as well as additional data simulated from the known FCM), respectively. We carry out a comparative analysis with other state-of-the-arts methods for learning FCMs, which include both Hebbian-based and genetic-based approaches.

A. Evaluation Criteria

Several quality criteria, which include in sample and out of sample errors that evaluate the dynamic properties of the candidate FCM and matrix error, specificity, sensitivity, and SS mean that evaluate the static properties, were measured.

1) *In sample error*—measures the difference between the input data (given as the input data matrix) and the data generated by simulating the candidate FCM from the first row in the input data matrix (initial state vector). The criterion is defined as a normalized average of absolute errors between corresponding concept values at each iteration:

$$In\ sample\ error = \frac{1}{(K-1)\cdot N}\sum_{t=1}^{K-1}\sum_{n=1}^{N}\left|C_n(t) - \hat{C}_n(t)\right| \tag{6}$$

where $C_n(t)$ is the value of node n at iteration t in the input data, $\hat{C}_n(t)$ is the value of a node n at iteration t from simulation of the candidate FCM from $\boldsymbol{C}(0)$ initial state vector, K is the input data length, and N is the number of nodes.

2) *Out of sample error*—evaluates generalization capabilities of the candidate FCM. To compute this criterion, both the input model and the candidate FCMs are simulated from ten randomly chosen initial state vectors that were not used to learn the candidate FCM. Subsequently, the error values, which compare state vector sequences generated from the input FCM and the candidate FCM, are computed for each of the ten simulations, and an average of these values is reported:

$$Out\ of\ sample\ error = \frac{1}{P\cdot(K-1)\cdot N}\sum_{p=1}^{P}\sum_{t=1}^{K-1}\sum_{n=1}^{N}\left|C_n^p(t) - \hat{C}_n^p(t)\right| \tag{7}$$

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

3) *Execution time*—measures the time (in seconds) to learn the candidate FCM from the given data. All methods were implemented in-house using C language and were executed on the same hardware platform. The hardware used to execute the experiments was a state-of-the-art 12-way IBM p570 server with POWER5 processors. However, it should be noted that this measure is not suitable to compare computational complexity of the learning methods since it just reports average time needed to learn FCM for a given setup.

4) *Matrix error*—evaluates the candidate FCM structure against the input model. It is defined as a normalized average of absolute errors between corresponding weights:

$$Matrix\ error = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}|e_{ij} - \hat{e}_{ij}| \tag{8}$$

where e_{ij} is the edge weight between nodes C_i and C_j in the input model and $\hat{e}_{ij}$ is the edge weight between nodes C_i and C_j in the candidate FCM.

The matrix error measure does not provide sufficient insights into the map structure in terms of its density. Therefore, in addition, the structural evaluation was extended and transformed into a binary classification problem with two classes, namely, *zeros* and *nonzeros*, which are defined for all map weights. Each weight from both the original FCM and the candidate FCM is assigned to one of the two classes. Each comparison of the corresponding weights from the input FCM with these from the candidate FCM results in one of the four outcomes, TP—correctly identified zero, TN—correctly identified nonzero, FP—nonzero (in the input model) incorrectly identified as zero (in the candidate FCM), and FN—zero (in the input model) incorrectly identified as nonzero (in the candidate FCM). Two measures, i.e., sensitivity and specificity, are used to evaluate the classification quality based on the cardinalities of the four outcomes over the entire connection matrix:

$$Sensitivity = \frac{TP}{TP+FN} \quad Specificity = \frac{TN}{TN+FP}. \tag{9}$$

In addition, the harmonic mean of sensitivity and specificity $SS-mean$ is calculated:

$$SSmean = \frac{2\cdot Sensitivity\cdot Specificity}{Sensitivity + Specificity}. \tag{10}$$

This measure is a weighted average of the sensitivity and specificity which ranges between 0 and 1, where 1 corresponds to a perfect result.

B. Data sets

We used both synthetic and real-world data. In the first scenario, the data for each experiment (the input data matrix) were obtained by simulating randomly generated FCMs (input models) and starting from a certain random initial vector. Next, ten randomly chosen initial state vectors were generated to perform the out of sample tests. The experiments were realized for FCMs of size 5, 10, 20, and 40 concepts and densities 20% and 40%. For each setup, they were repeated five times with

TABLE III
EXPERIMENTAL RESULTS WITH SYNTHETIC DATA. #NODES AND DENSITY COLUMNS DEFINE THE INPUT MODEL. THE SUBSEQUENT COLUMNS CORRESPOND TO EVALUATION MEASURES DESCRIBED IN SECTION IV-A. TIME, IN SAMPLE, AND MATRIX, SPECIFICITY, SENSITIVITY, AND SS MEAN VALUES ARE AVERAGED ACROSS FIVE EXPERIMENTS FOR EACH SETUP (WITH CORRESPONDING STANDARD DEVIATIONS), WHILE OUT OF SAMPLE VALUES ARE ADDITIONALLY AVERAGED ACROSS TEN EXPERIMENTS PERFORMED FROM DIFFERENT INITIAL VECTORS

	# nodes	Density	Time [s] Avg (Std)	In sample Avg (Std)	Out of sample Avg (Std)	Matrix Avg (Std)	Spec	Sens	SS Mean
Sparse RCGA	5	20%	720 (28.04)	0.004 (0.004)	0.010 (0.012)	0.002 (0.002)	0.85	0.93	0.89
	5	40%	732 (28.61)	0.004 (0.005)	0.008 (0.011)	0.003 (0.003)	0.87	0.91	0.89
	10	20%	1479 (34.16)	0.005 (0.005)	0.123 (0.141)	0.105 (0.198)	0.58	0.90	0.71
	10	40%	1455 (36.28)	0.005 (0.005)	0.124 (0.151)	0.168 (0.200)	0.69	0.85	0.76
	20	20%	3963 (47.69)	0.007 (0.006)	0.142 (0.143)	0.122 (0.219)	0.49	0.86	0.62
	20	40%	3968 (45.33)	0.006 (0.008)	0.146 (0.148)	0.203 (0.295)	0.59	0.79	0.68
	40	20%	13141 (57.11)	0.017 (0.020)	0.166 (0.183)	0.135 (0.235)	0.36	0.84	0.50
	40	40%	13155 (61.82)	0.019 (0.022)	0.164 (0.196)	0.245 (0.288)	0.48	0.72	0.58
RCGA	5	20%	547 (29.75)	0.005 (0.006)	0.017 (0.017)	0.321 (0.381)	0.93	0.12	0.20
	5	40%	536 (27.18)	0.005 (0.005)	0.012 (0.018)	0.361 (0.372)	0.96	0.09	0.16
	10	20%	1181 (33.81)	0.006 (0.007)	0.135 (0.137)	0.398 (0.322)	0.96	0.11	0.19
	10	40%	1169 (36.63)	0.006 (0.007)	0.131 (0.140)	0.385 (0.316)	0.94	0.11	0.20
	20	20%	3337 (47.22)	0.008 (0.009)	0.151 (0.149)	0.426 (0.346)	0.94	0.09	0.16
	20	40%	3384 (49.58)	0.008 (0.008)	0.152 (0.149)	0.413 (0.376)	0.94	0.08	0.14
	40	20%	11151 (58.29)	0.019 (0.022)	0.171 (0.187)	0.453 (0.385)	0.94	0.08	0.15
	40	40%	11130 (60.06)	0.020 (0.022)	0.167 (0.189)	0.436 (0.368)	0.96	0.08	0.15
DD-NHL	5	20%	1470 (110.88)	0.197 (0.186)	0.199 (0.209)	0.317 (0.345)	0.96	0.06	0.11
	5	40%	1445 (117.62)	0.189 (0.197)	0.197 (0.203)	0.381 (0.333)	0.96	0.07	0.12
	10	20%	4368 (227.36)	0.191 (0.188)	0.201 (0.181)	0.412 (0.356)	0.96	0.07	0.13
	10	40%	4505 (222.72)	0.211 (0.184)	0.192 (0.189)	0.423 (0.348)	0.93	0.05	0.10
	20	20%	9632 (336.14)	0.203 (0.245)	0.201 (0.222)	0.464 (0.316)	0.93	0.08	0.14
	20	40%	9812 (346.43)	0.203 (0.245)	0.203 (0.224)	0.436 (0.348)	0.94	0.07	0.13
	40	20%	22248 (376.36)	0.194 (0.168)	0.198 (0.199)	0.468 (0.388)	0.96	0.07	0.12
	40	40%	21168 (403.52)	0.187 (0.160)	0.206 (0.209)	0.465 (0.384)	0.93	0.06	0.12
NHL	5	20%	1584 (138.24)	0.184 (0.196)	0.201 (0.195)	0.345 (0.349)	0.94	0.07	0.13
	5	40%	1552 (142.56)	0.186 (0.188)	0.209 (0.197)	0.346 (0.306)	0.93	0.05	0.10
	10	20%	4264 (188.87)	0.204 (0.201)	0.200 (0.215)	0.420 (0.348)	0.93	0.07	0.12
	10	40%	4018 (192.61)	0.202 (0.211)	0.206 (0.217)	0.435 (0.301)	0.93	0.06	0.11
	20	20%	9682 (269.66)	0.201 (0.192)	0.199 (0.222)	0.461 (0.348)	0.93	0.07	0.13
	20	40%	9830 (272.44)	0.187 (0.179)	0.201 (0.222)	0.468 (0.322)	0.93	0.05	0.10
	40	20%	21416 (405.01)	0.205 (0.216)	0.211 (0.190)	0.489 (0.388)	0.95	0.08	0.15
	40	40%	21703 (388.97)	0.205 (0.206)	0.213 (0.193)	0.498 (0.389)	0.94	0.06	0.12

different input models and state vectors. We report the average and standard deviations from these five repeats.

In the second scenario, three large FCMs reported in literature were used. In order to minimize any potential bias, we selected FCMs that were developed as consensus based on minimum three experts' opinions rather than by a single expert knowledge. The models involve 9, 12, and 13 concepts that describe, correspondingly, plant supervisory [42], military planning [49], and slurry rheology [4]. These models were simulated from initial vectors suggested by the authors to generate the input data matrix. Similarly to experiments with the synthetic data, the out of sample error was computed by simulating the input model and the candidate FCM from ten randomly chosen initial vectors.

C. Results

Table III summarizes the experimental results with the synthetic data. The density estimate value was set equal to the actual density of the known FCM used to generate the input data. The columns reporting the execution time, in sample error, matrix error, specificity, sensitivity, and SS mean are averaged over the five independent experiments performed with each setup, whereas the out of sample error was additionally averaged over the ten experiments performed with different initial vectors. The results were compared with three other recent learning methods, namely, RCGA [35], NHL [26], and DD-NHL [37]. In order to perform unbiased comparison, the same initial population was used for RCGA and SRCGA methods (in the latter case, the gene reset was carried out after the initialization; see STEP 1 of the SRCGA). Since the NHL and DD-NHL methods use just a single initial connection matrix as their starting point (in contrary to the RCGA and SRCGA that use the entire population of 100 chromosomes), 100 experiments were carried out with these methods—each experiment was performed with a different initial matrix that was taken from the initial population utilized by the RCGA and SRCGA methods. The best (based on the in sample error), among the 100 experiments, results were reported in Table III for both the NHL and DD-NHL methods. Consequently, we report cumulative time, over the 100 experiments, for the NHL and DD-NHL methods.

The results in Table III show that SRCGA outperforms other methods in terms of the out of sample and the matrix errors. In order to facilitate the comparative analysis, we used paired t-test to investigate statistical significance of differences be-

TABLE IV
TESTS OF STATISTICAL SIGNIFICANCE OF DIFFERENCES BETWEEN THE SRCGA AND OTHER METHODS GIVEN THAT THE DENSITY ESTIMATE VALUE WAS SET EQUAL TO THE ACTUAL DENSITY OF THE KNOWN FCM USED TO GENERATE THE INPUT DATA. THE NUMBER OF PLUS SIGNS IN CELLS REFLECTS DIFFERENT CONFIDENCE LEVELS, I.E.,, 98% (+ + +), 95% (++), AND 90% (+), FOR THE SIGNIFICANCE TESTS

# nodes	Density	Out of sample[)]			Matrix		
		RCGA	DD-NHL	NHL	RCGA	DD-NHL	NHL
5	20%	+++	+++	+++	+++	+++	+++
5	40%	+++	+++	+++	+++	+++	+++
10	20%	+++	+++	+++	+++	+++	+++
10	40%	+++	+++	+++	+++	+++	+++
20	20%	++	+++	+++	+++	+++	+++
20	40%	++	+++	+++	+++	+++	+++
40	20%	+	+++	+++	+++	+++	+++
40	40%	+	+++	+++	+++	+++	+++

tween the proposed method and other methods. The obtained results are summarized in Table IV.

The analysis of the out of sample (see Table IV) shows that the SRCGA is significantly better at 98% confidence level than the three other methods under consideration for the small and medium size maps (containing five and ten nodes). For larger maps (with 20 and 40 nodes), the improvements at 98% are true when the map is compared against the DD-NHL and NHL approaches, whereas the confidence level drops to 95% (20 nodes) and 90% (40 nodes) when the SRCGA is compared with the RCGA. The results of the proposed method (see Table III) show that the maximal difference between the out of sample error for different map densities for a given map size is equal to 0.004. The matrix error (see Table IV) generated by the SRCGA method is statistically significantly better at 98% confidence level than the error produced by the three existing methods for all setups. The results from Table III suggest that the matrix error is smaller for sparser maps, i.e., 20%, when compared with the denser maps. This trend is more evident for larger maps, i.e., 0.135 versus 0.245 for maps consisting of 40 nodes, which is due to a larger number of zeros in the sparser maps and the fact that the proposed method enforces zeros in its solution. We further investigate this observation by calculating baselines using ten randomly generated maps of a given density and size 40, with their nonzero weights set at random. The average matrix error for these maps is equal to 0.18 for a density of 20% and 0.34 for a density of 40%. This means that the SRCGA improves over the baseline by 25% for the sparser and by 28% for the denser maps, respectively.

Additional insights concerning structural quality of the candidate FCM can be obtained from the analysis of the three last columns in Table III. Specificity determines the ratio of correctly assigned "nonzero" values, whereas sensitivity determines the ratio of correctly assigned "zero" values in the connection matrix of the candidate FCM when compared to the input model. Using an approach described in the previous paragraph, we calculated the baselines for these measures. The baseline specificities are equal to 0.18 (20%) and 0.37 (40%), the baseline sensitivities are equal to 0.82 (20%) and 0.63 (40%), and the baseline SS means are 0.30 (20%) and 0.47 (40%), respectively. In addition, ten random maps without any density restrictions were generated for each setup, and the following baseline was obtained: specificities of 0.95 (20%) and 0.94 (40%), sensitivities of 0.05 (20%) and 0.06 (40%), and SS means of 0.10 (20%) and 0.11 (40%). The analysis of SS mean values, which combine specificity and sensitivity, shows that the SRCGA method outperforms the other considered approaches. This demonstrates that structurally accurate solutions (evaluated not only by the number of nonzero relationships but also based on their placement) are found by guiding the learning process. Hence, this suggests that focusing the search on a subspace with similar (density-driven) models is beneficial for the learning of maps characterized by good quality that is measured by the SS mean. Moreover, the proposed method is between five times better (for large 40-nodes maps) and about nine times better (for small five-node maps) than the baseline that does not consider the density, and it improves over the baseline with the known density by 40% and 16% for the sparser and denser maps of size 40, respectively. When compared against the RCGA, NHL, and DD-NHL, the SRCGA provides substantially higher sensitivity and lower levels of specificity. We note that both of these measures are relatively balanced in the case of real RCGA, while the other methods are characterized by high specificity (due to the fact that they predict virtually all weights with nonzero values) and very low specificity between 5% and 12%. Although RCGA and SRCGA methods obtain the out of sample errors that differ "only" at 90% significance for the 40 nodes maps (see Table IV), the SS mean values of the proposed method are over three times higher, and they demonstrate that the corresponding maps are more useful for the static analysis. Overall, the results for smaller maps suggest that, by using the additional density parameter, the SRCGA method is capable of learning maps with high structural quality. This parameter guides the evolutionary learning process and effectively eliminates these maps that could provide accurate dynamic modeling being structurally significantly different from the original map, refer to the results in Table III for RCGA, NHL, and DD-NHL methods. For larger maps, even though the structural quality of SRCGA solutions deteriorates, this method still outperforms the other approaches by focusing the search only on a subset of the search space.

The time to compute the solutions is similar for all considered methods. We observe a moderate, about 20%–30%, increase between the SRCGA and the RCGA methods. The time for the NHL and DD-NHL method concerns learning of 100 models. Since these methods can learn a single model (in contrast to RCGA-based methods that operate over a population of 100 models), they provide a faster solution if the user would be satisfied with a lower quality of the generated map. We note that the quality of these faster solutions would be likely

TABLE V
EXPERIMENTAL RESULTS WITH REAL-WORLD MODELS. THE DENSITY EST COLUMN REPORTS THE DENSITY ESTIMATE PARAMETER USED WITH THE SRCGA LEARNING. THE SUBSEQUENT COLUMNS CORRESPOND TO EVALUATION MEASURES DESCRIBED IN SECTION IV-A. TIME, IN SAMPLE, AND MATRIX, SPECIFICITY, SENSITIVITY, AND SS MEAN VALUES ARE AVERAGED ACROSS FIVE EXPERIMENTS FOR EACH SETUP (WITH CORRESPONDING STANDARD DEVIATIONS), WHILE OUT OF SAMPLE VALUES ARE ADDITIONALLY AVERAGED ACROSS TEN EXPERIMENTS PERFORMED FROM DIFFERENT INITIAL VECTORS. ABBREVIATIONS IN THE METHOD COLUMN: SRCGA—SPARSE RCGA, D&C—DIVIDE AND CONQUER, SRCGA 37%—SRCGA WITH THE DEFAULT DENSITY ESTIMATE VALUE, I.E., 37%

# nodes	Method	Density Est	Time [s] Avg	In sample Avg (Std)	Out of sample Avg (Std)	Matrix Avg (Std)	Spec	Sens	SS Mean
13	SRCGA	10%	2148	0.006 (0.006)	0.141 (0.138)	0.190 (0.317)	0.10	0.96	0.17
		15%	2142	0.006 (0.007)	0.139 (0.139)	0.187 (0.322)	0.20	0.93	0.32
		20%	2141	0.006 (0.006)	0.137 (0.141)	0.169 (0.316)	0.39	0.92	0.55
		25%	2148	0.006 (0.007)	0.136 (0.132)	0.150 (0.307)	0.47	0.89	0.61
		30%	2145	0.006 (0.007)	0.134 (0.133)	0.138 (0.269)	0.55	0.86	0.67
		35%	2148	0.006 (0.006)	0.132 (0.131)	0.120 (0.245)	0.63	0.84	0.72
		40%	2143	0.006 (0.007)	0.132 (0.130)	0.117 (0.247)	0.65	0.84	0.73
		45%	2152	0.006 (0.006)	0.136 (0.134)	0.123 (0.265)	0.70	0.72	0.71
		50%	2143	0.006 (0.008)	0.138 (0.135)	0.129 (0.265)	0.72	0.62	0.67
		55%	2147	0.006 (0.007)	0.139 (0.132)	0.135 (0.311)	0.76	0.56	0.65
		60%	2142	0.006 (0.006)	0.139 (0.134)	0.166 (0.319)	0.80	0.46	0.58
		65%	2143	0.006 (0.007)	0.140 (0.131)	0.179 (0.318)	0.82	0.41	0.55
		70%	2144	0.006 (0.007)	0.143 (0.138)	0.219 (0.349)	0.84	0.33	0.47
		75%	2148	0.006 (0.006)	0.144 (0.139)	0.239 (0.329)	0.87	0.30	0.44
		80%	2143	0.006 (0.007)	0.149 (0.141)	0.265 (0.346)	0.92	0.19	0.31
		85%	2148	0.006 (0.007)	0.149 (0.144)	0.305 (0.359)	0.94	0.15	0.26
		90%	2141	0.007 (0.008)	0.149 (0.149)	0.329 (0.376)	0.95	0.10	0.18
		95%	2143	0.007 (0.008)	0.151 (0.137)	0.346 (0.388)	0.95	0.09	0.17
		100%	2143	0.007 (0.009)	0.151 (0.141)	0.392 (0.386)	0.95	0.08	0.15
	D&C SRCGA	39%	1125	0.006 (0.007)	0.136 (0.133)	0.133 (0.131)	0.62	0.82	0.71
	SRCGA 37%	37%	2146	0.006 (0.006)	0.138 (0.142)	0.121 (0.221)	0.57	0.82	0.67
	D&C SRCGA 37%	37%	1149	0.006 (0.007)	0.140 (0.127)	0.134 (0.137)	0.52	0.81	0.63
	RCGA	N/A	2049	0.007 (0.009)	0.151 (0.141)	0.392 (0.386)	0.95	0.08	0.15
	D&C RCGA	N/A	1031	0.007 (0.009)	0.154 (0.143)	0.411 (0.427)	0.95	0.10	0.18
	DD-NHL	N/A	5156	0.191 (0.165)	0.197 (0.165)	0.426 (0.382)	0.94	0.07	0.13
	NHL	N/A	5015	0.195 (0.180)	0.199 (0.184)	0.436 (0.379)	0.94	0.06	0.11
12	SRCGA	38%	1928	0.005 (0.005)	0.125 (0.140)	0.118 (0.143)	0.59	0.89	0.71
	D&C SRCGA	38%	995	0.006 (0.006)	0.131 (0.128)	0.121 (0.119)	0.55	0.87	0.67
	SRCGA 37%	37%	1998	0.005 (0.005)	0.129 (0.128)	0.121 (0.143)	0.59	0.88	0.71
	D&C SRCGA 37%	37%	1002	0.006 (0.006)	0.132 (0.118)	0.122 (0.123)	0.55	0.82	0.66
	RCGA	N/A	1789	0.005 (0.005)	0.145 (0.151)	0.405 (0.316)	0.96	0.10	0.18
	D&C RCGA	N/A	902	0.006 (0.005)	0.148 (0.140)	0.399 (0.388)	0.91	0.10	0.18
	DD-NHL	N/A	4998	0.188 (0.245)	0.191 (0.188)	0.411 (0.346)	0.93	0.08	0.15
	NHL	N/A	4989	0.186 (0.005)	0.193 (0.151)	0.405 (0.316)	0.96	0.10	0.18
9	SRCGA	25%	1187	0.005 (0.005)	0.121 (0.137)	0.102 (0.151)	0.65	0.87	0.74
	D&C SRCGA	25%	596	0.006 (0.005)	0.124 (0.111)	0.112 (0.117)	0.64	0.85	0.73
	SRCGA 37%	37%	1183	0.005 (0.005)	0.127 (0.120)	0.112 (0.149)	0.69	0.73	0.71
	D&C SRCGA 37%	37%	591	0.006 (0.005)	0.129 (0.122)	0.119 (0.118)	0.66	0.72	0.69
	RCGA	N/A	995	0.005 (0.005)	0.131 (0.141)	0.395 (0.348)	0.95	0.09	0.16
	D&C RCGA	N/A	511	0.007 (0.005)	0.136 (0.123)	0.358 (0.325)	0.94	0.09	0.16
	DD-NHL	N/A	4177	0.187 (0.184)	0.189 (0.197)	0.438 (0.316)	0.94	0.07	0.13
	NHL	N/A	4169	0.188 (0.201)	0.191 (0.201)	0.419 (0.372)	0.95	0.08	0.15

lower than the values reported in Table III since these numbers correspond to the best solution among the 100 experiments. These quantified execution time values are affected by the initial conditions and the quality of the source code, and thus, these results should be treated as rough estimates, and they may not reflect the underlying computational complexity. Estimation of the computational complexity of the considered learning methods is a challenging task that falls beyond the scope of this paper and which deserves a separate contribution.

Table V shows the experimental results for the real-world models. Results were also compared to those produced by the divide and conquer strategy [38] applied to both RCGA and SRCGA methods. We used divide and conquer setup with two processors, i.e., the input data were split into two subsets since the learning quality decreases statistically significantly for the considered map sizes when using more processors [38]. For the SRCGA method, the experiments with the correct density estimate value, i.e., the actual density of the original map, are supplemented by experiments with the default density estimate value, i.e., 37%. We used the divide and conquer approach with both SRCGA setups as well as with generic RCGA method. In addition to the experiments that are analogous to the experiments performed with the synthetic data, we also carried out analysis of sensitivity of the proposed method to the *density estimate* parameter for the largest considered real-world map. The value of this parameter was varied between 10% and 100% with 5% increments, and we investigated the influence of the setting of the parameter values on the quality of the candidate FCM.

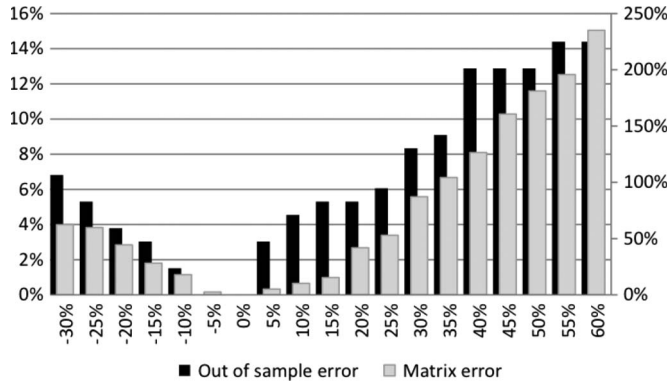


Fig. 3. Sensitivity of the proposed method to the setting of the density estimate parameter. Bars show relative increase of the out of sample error (scale to the left) and the matrix error (scale to the right) with respect to the lowest error obtained for the density estimate that is equal to 40%. Labels on the horizontal axis correspond to the difference between a given value of the density estimate and the value for the density of 40%.

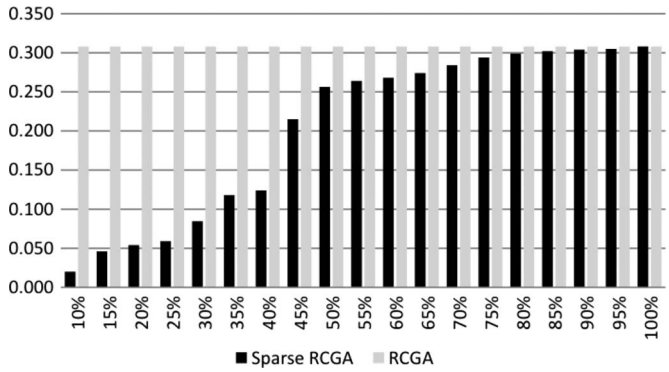


Fig. 4. Weight difference as a function of the density estimate. The baseline (gray bars) was estimated by the weight difference value for the RCGA method, which is independent of the density estimate.

The SRCGA approach with the correct density estimate consistently provides the smallest out of sample errors across the three real-world maps. The second best errors are obtained by the SRCGA with the default density estimate and by the generic RCGA method. As expected, the application of the divide and conquer strategy to each of the three RCGA methods (SRCGA with correct density estimate, SRCGA with default density estimate, and generic RCGA) increases the error value, but this increase is not statistically significant at 90% confidence level, as measured using the paired t-test. The values of the quality measures are consistent with the experiments obtained for the synthetic data that are reported in Table III.

Next, we analyze the results from Table V that concern the SRCGA learning of the largest considered map with 13 nodes with varying density estimate values. Both out of sample and matrix errors have the lowest values when the density estimate is equal to 40%, which is close to the actual density of 39%. These results are similar to the results for the synthetic maps with size of 10, which is comparable to the size of the slurry rheology map. Fig. 3 shows the influence of the density estimate on these two criteria for the learning of the slurry rheology map.

Both errors are upper bounded by the solution obtained from the RCGA method which does not utilize information about the density. This means that, even if the estimate of the density is incorrect, the maps generated by the SRCGA will be still better or at least equivalent, both in terms of the out of sample and matrix errors, when compared to the maps generated by the RCGA, NHL, and DD-NHL methods. We note that the RCGA is equivalent to the SRCGA with the density estimate of 100%, which is confirmed by the results in Table V. A relatively small increase in the out of sample error (up to 15%) across the entire range of the density estimate demonstrates that there are many structurally different maps (of different densities) that exhibit similar dynamic behavior. The classification results reported in Table V show that the SRCGA outperforms other methods in terms of the SS mean values. In order to put these numbers into perspective, we compute the baseline by averaging ten randomly generated maps with a density of 39%. The baseline specificity, sensitivity, and SS mean are equal to 0.35, 0.63, and 0.45, respectively. Table V shows that, when considering the SS mean values, the results obtained with the SRCGA for the density estimates between 20% and 70% are better than the baseline. Therefore, the density estimate could be off by as much as -20% or $+30\%$ with respect to the actual value, and the SS mean would still be better than this baseline that is calculated for the actual density.

Finally, we investigate the relationship between the density estimate and the convergence of the method to the same candidate FCM. This experiment aims at finding whether similar solution maps would be found if the input data (data matrix), which correspond to the same underlying FCM, would change. Solutions from the previous experiment were taken as a reference (reference models). We generated ten new random initial vectors, used them to generate ten different data matrices from the actual slurry rheology map, and performed learning for each input data matrix. Next, we calculated the average difference per concept (weight difference) for a given method according to the following expression:

$$Weight\ difference = \frac{1}{NP}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{p=1}^{P}|e_{ij} - \hat{e}_{ij}(p)| \quad (11)$$

where e_{ij} is the edge weight between nodes C_i and C_j in the reference model, $\hat{e}_{ij}(p)$ is the edge weight between nodes C_i and C_j in the candidate FCM obtained from pth learning, $N = 13$ is the number of nodes, and $P = 10$ is the number of experiments.

Fig. 4 shows the relationship between the density estimate value and weight difference. It shows that the SRCGA method is capable of finding similar solutions for a given system when the density estimate is lower or the same as the true density. For the estimate of 40%, which is the closest to the actual density of 39%, the SRCGA is over two times more consistent in finding the same solution than RCGA (according to the weight difference value). In the case when the density is overestimated, the consistency of the proposed method decreased, but it is still upper bounded by the weight difference value of the RCGA method.

V. Conclusion

FCMs have received a well-deserved attention in recent years. Numerous successful applications in various research

and industrial domains clearly demonstrate the effectiveness of this modeling technique. However, it seems that further development of FCMs is somewhat constrained by deficiencies that are present in their underlying theoretical framework. One of the issues that have been recently investigated is to provide a systematic approach to efficient design of FCMs.

In this paper, a new method for learning sparse FCMs from data has been proposed. It aims at learning models that are structurally more similar to real-world models when compared with the FCMs obtained from current fully automated learning approaches. Our examination of the published FCMs reveals that the maps are relatively sparsely connected. This information has been utilized to develop a new learning strategy, called *SRCGA*, that guides the learning process toward finding maps of a predefined density, determined by the density estimate parameter. Based on literature review, we found that the average density of the published models is around 37%; hence, this value could be used as a density estimate if a user has no prior knowledge about the modeled system's density.

The proposed method exploits the modified RCGA. This particular choice was motivated by the nature of the underlying optimization problem, which involves search through a large continuous solution space with many suboptimal solutions. We extended the generic RCGA to accommodate the constraints imposed by the density estimate parameter. Experimental analysis of the SRCGA method on synthetic data shows that, given a correct density estimate, it is capable of producing models that are statistically significantly better at the 98% level of confidence than models generated by all other considered learners for all tested setups, except for the 20 and 40 nodes maps when compared to the RCGA method where the proposed method is statistically significantly better at the 95% and 90% level of confidence, respectively. Analysis of the structural quality expressed by the matrix error reveals that the SRCGA approach performs statistically significantly better (at the 98% level of confidence) than all other considered methods for all setups. Experiments with real-world models demonstrate that, when the correct density estimate is unknown, the SRCGA method is still able to develop models of a quality equivalent or better than the quality offered by the other methods. In addition, it converges to more similar maps when using different input data that correspond to the same system.

Learning FCMs from data is a difficult task due to the high complexity and dimensionality of the underlying problem, i.e., the quadratic growth of the number of weights with respect to the number of concepts. The automated learning methods need to cope with a substantial challenge of existence of many suboptimal solutions that are structurally considerably different from each other. Consequently, such maps are not suitable for the static analysis. The proposed SRCGA learning method is the first method that considers structural features (density) of the generated FCMs. The idea behind introducing this method is to search the highly complex continuous space in a sound manner by concentrating efforts only on a fairly small subspace. Experimental results that concern both static and dynamic properties of the FCMs learned with the SRCGA method are promising. These results show that the density estimate can be a useful parameter to improve the learning quality when compared to the generic RCGA method that does not use any prior information concerning the structure of the map. However, further research toward a systematic approach to develop FCMs from data is still needed.

REFERENCES

[1] J. Aguilar, "A dynamic fuzzy-cognitive-map approach based on random neural networks," *Int. J. Comput. Cognition*, vol. 1, no. 4, pp. 91–107, Jan. 2003.

[2] J. Aguilar, "A survey about fuzzy cognitive maps papers," *Int. J. Comput. Cognition*, vol. 3, no. 2, pp. 27–33, Jun. 2005.

[3] R. Axelrod, *Structure of Decision: The Cognitive Maps of Political Elites*. Princeton, NJ: Princeton Univ. Press, 1976.

[4] G. A. Banini and R. A. Bearman, "Application of fuzzy cognitive maps to factors affecting slurry rheology," *Int. J. Mineral Process.*, vol. 52, no. 4, pp. 233–244, Feb. 1998.

[5] D. Bhandari, N. R. Pal, and S. K. Pal, "Directed mutation in genetic algorithms," *Inform. Sci.*, vol. 79, no. 3/4, pp. 251–270, Jul. 1994.

[6] S. Bueno and J. L. Salmeron, "Benchmarking main activation functions in fuzzy cognitive maps," *Expert Systems With Applications*, vol. 36, no. 3, pp. 5221–5229, Apr. 2009.

[7] R. J. G. B. Campello and W. C. Amaral, "Towards true linguistic modelling through optimal numerical solution," *Int. J. Syst. Sci.*, vol. 34, no. 2, pp. 139–157, Feb. 2003.

[8] C. Carlsson and R. Fuller, "Adaptive fuzzy cognitive maps for hyperknowledge representation in strategy formation process," in *Proc. Int. Panel Conf. Soft Intell. Comput.*, 1996, pp. 43–50.

[9] J. A. Dickerson and B. Kosko, "Virtual worlds as fuzzy cognitive maps," in *Proc. Virtual Reality Annu. Int. Symp.*, 1993, pp. 471–477.

[10] J. A. Dickerson and B. Kosko, "Virtual worlds as fuzzy cognitive maps," *Presence*, vol. 3, no. 2, pp. 173–189, 1994.

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

[12] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artif. Intell. Rev.*, vol. 12, no. 4, pp. 265–319, Aug. 1998.

[13] A. V. Huerga, "A balanced differential learning algorithm in fuzzy cognitive maps," in *Proc. 16th Int. Work. on Qualitative Reasoning*, 2002, poster.

[14] R. I. John and P. R. Innocent, "Modeling uncertainty in clinical diagnosis using fuzzy logic," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 6, pp. 1340–1350, Dec. 2005.

[15] M. S. Khan and A. Chong, "Fuzzy cognitive map analysis with genetic algorithm," in *Proc. Indian Int. Conf. Artif. Intell.*, 2003, pp. 1196–1205.

[16] B. Kosko, "Fuzzy cognitive maps," *Int. J. Man-Mach. Stud.*, vol. 24, pp. 65–75, 1986.

[17] B. Kosko, *Fuzzy Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[18] D. E. Koulouriotis, I. E. Diakoulakis, and D. M. Emiris, "Learning fuzzy cognitive maps using evolution strategies: A novel schema for modeling and simulating high-level behavior," in *Proc. IEEE Congr. Evol. Comput.*, 2001, pp. 364–371.

[19] D. E. Koulouriotis, I. E. Diakoulakis, D. M. Emiris, E. N. Antonidakis, and I. A. Kaliakatsos, "Efficiently modeling and controlling complex dynamic systems using evolutionary fuzzy cognitive maps," *Int. J. Comput. Cognition*, vol. 1, no. 2, pp. 41–65, Jun. 2003.

[20] K. C. Lee, W. J. Lee, O. B. Kwon, J. H. Han, and P. I. Yu, "Strategic planning simulation based on fuzzy cognitive map knowledge and differential game," *Simulation*, vol. 71, no. 5, pp. 316–327, Nov. 1998.

[21] J. Lis, "Genetic algorithm with the dynamic probability of mutation in the classification problem," *Pattern Recognit. Lett.*, vol. 16, no. 12, pp. 1311–1320, Dec. 1995.

[22] Y. Miao, "Dynamic cognitive networks—An extension of fuzzy cognitive map," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 5, pp. 760–770, Oct. 2001.

[23] S. T. Mohr, "The use and interpretation of fuzzy cognitive maps," M.S. thesis, Rensselaer Polytechnic Inst., Troy, NY, 1997.

[24] T. D. Ndousse and T. Okuda, "Computational intelligence for distributed fault management in networks using fuzzy cognitive maps," in *Proc. IEEE Int. Conf. Commun. Converging Technol. Tomorrow Appl.*, 1996, vol. 3, pp. 1558–1562.

[25] E. Oja, H. Ogawa, and J. Wangviwattam, "Learning in nonlinear constrained Hebbian networks," in *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas, Eds. Amsterdam, The Netherlands: Elsevier, 1991, pp. 385–390.

[26] E. Papageorgiou, C. Stylios, and P. Groumpos, "Fuzzy cognitive map learning based on nonlinear Hebbian rule," *Lecture Notes Comput. Sci.*, vol. 2903, pp. 256–268, 2003.
[27] E. Papageorgiou, C. D. Stylios, and P. P. Groumpos, "Active Hebbian learning algorithm to train fuzzy cognitive maps," *Int. J. App. Reasoning*, vol. 37, no. 3, pp. 219–249, 2004.
[28] K. E. Parsopoulos, E. I. Papageorgiou, P. P. Groumpos, and M. N. Vrahatis, "A first study of fuzzy cognitive maps learning using particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2003, pp. 1440–1447.
[29] C. E. Pelaez and J. B. Bowles, "Applying fuzzy cognitive maps knowledge representation to failure modes effects analysis," in *Proc. IEEE Ann. Symp. Reliab. Maintainability*, 1995, pp. 450–456.
[30] A. Siraj, S. Bridges, and R. Vaughn, "Fuzzy cognitive maps for decision support in an intelligent intrusion detection system," in *Proc. IFSA World Congr. 20th NAFIPS Int. Conf.*, 2001, vol. 4, pp. 2165–2170.
[31] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. New York: Springer-Verlag, 2007.
[32] W. Stach, L. A. Kurgan, and W. Pedrycz, "Numerical and linguistic prediction of time series with the use of fuzzy cognitive maps," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 1, pp. 61–72, Feb. 2008.
[33] W. Stach and L. Kurgan, "Modeling software development project using fuzzy cognitive maps," in *Proc. 4th ASERC Workshop Quantitative Soft Softw. Eng.*, 2004, pp. 55–60.
[34] W. Stach, L. Kurgan, and W. Pedrycz, "Parallel learning of large fuzzy cognitive maps," in *Proc. Int. Joint Conf. Neural Netw.*, 2007, pp. 1584–1589.
[35] W. Stach, L. Kurgan, W. Pedrycz, and M. Reformat, "Genetic learning of fuzzy cognitive maps," *Fuzzy Sets Syst.*, vol. 153, no. 3, pp. 371–401, Aug. 2005.
[36] W. Stach, L. Kurgan, W. Pedrycz, and M. Reformat, "Learning fuzzy cognitive maps with required precision using genetic algorithm approach," *Electron. Lett.*, vol. 40, no. 24, pp. 1519–1520, Nov. 2004.
[37] W. Stach, L. A. Kurgan, and W. Pedrycz, "Data-driven nonlinear Hebbian learning method for fuzzy cognitive maps," in *Proc. World Congr. Comput. Intell.*, 2008, pp. 1975–1981.
[38] W. Stach, L. A. Kurgan, and W. Pedrycz, "A divide and conquer method for learning large fuzzy cognitive maps," *Fuzzy Sets Syst.*, vol. 161, no. 19, pp. 2515–2532, Oct. 2010.
[39] W. Stach, L. A. Kurgan, and W. Pedrycz, "A survey of fuzzy cognitive map learning methods," in *Issues in Soft Computing: Theory and Applications*, P. Grzegorzewski, M. Krawczak, and S. Zadrozny, Eds. Warszawa, Poland: Exit, 2005, pp. 71–84.
[40] W. Stach, L. A. Kurgan, and W. Pedrycz, "Expert-based and computational methods for developing fuzzy cognitive maps," in *Advances in Theory, Methodologies, Tools and Applications, Studies in Fuzziness and Soft Computing*, vol. 247, M. Glykas, Ed. New York: Springer-Verlag, 2010, pp. 23–41.
[41] M. A. Styblinski and B. D. Meyer, "Signal flow graphs vs. fuzzy cognitive maps in application to qualitative circuit analysis," *Int. J. Man-Mach. Stud.*, vol. 35, no. 2, pp. 175–186, Aug. 1991.
[42] C. D. Stylios and P. P. Groumpos, "Fuzzy cognitive map in modeling supervisory control systems," *J. Intell. Fuzzy Syst.*, vol. 8, no. 2, pp. 83–98, Mar. 2000.
[43] C. D. Stylios and P. P. Groumpos, "Fuzzy cognitive maps: A model for intelligent supervisory control systems," *Comp. Ind.*, vol. 39, no. 3, pp. 229–238, Jul. 1999.
[44] C. D. Stylios and P. P. Groumpos, "Modeling complex systems using fuzzy cognitive maps," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 1, pp. 155–162, Jan. 2004.
[45] C. D. Stylios and P. P. Groumpos, "The challenge of modelling supervisory systems using fuzzy cognitive maps," *J. Intell. Manuf.*, vol. 9, no. 4, pp. 339–345, Sep. 1998.
[46] A. K. Tsadiras, "Using fuzzy cognitive maps for e-commerce strategic planning," in *Proc. 9th Panhellenic Conf. Informat.*, 2003, pp. 142–151.
[47] A. K. Tsadiras, "Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps," *Inform. Sci.*, vol. 178, no. 20, pp. 3880–3894, Oct. 2008.
[48] A. K. Tsadiras, I. Kouskouvelis, and K. G. Margaritis, "Making political decisions using fuzzy cognitive maps: The FYROM crisis," in *Proc. 8th Panhellenic Conf. Inf.*, 2001, vol. 2, pp. 501–510.
[49] D. Yaman and S. Polat, "A fuzzy cognitive map approach for effect-based operations: An illustrative case," *Inf. Sci.*, vol. 179, no. 4, pp. 382–403, Feb. 2009.

Wojciech Stach (M'05) received the M.Sc. degree in automation and robotics (with honors) from the AGH University of Science and Technology, Krakow, Poland, in 2003 and the Ph.D. degree in software engineering and intelligent systems from the University of Alberta, Edmonton, Canada, in 2010.

He is currently working as a Scientific Research Analyst on projects that use data mining and machine learning in auto insurance risk assessment. He was involved in research projects related to software engineering and intelligent systems with special emphasis on software project management, systems modeling, and data mining. His work on fuzzy cognitive maps was published in two book chapters as well as several journal and conference papers.

Witold Pedrycz (M'88–SM'94–F'99) received the M.Sc., Ph.D. and D.Sci. degrees from the Silesian University of Technology, Gliwice, Poland.

He is a Professor and Canada Research Chair (CRC—computational intelligence) with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. In 2009, he was elected as a foreign member of the Polish Academy of Sciences. His main research directions involve computational intelligence, fuzzy modeling and granular computing, knowledge discovery and data mining, fuzzy control, pattern recognition, knowledge-based neural networks, relational computing, and software engineering. He has published numerous papers in this area. He is also an author of 14 research monographs covering various aspects of computational intelligence and software engineering.

Dr. Pedrycz has been a member of numerous program committees of IEEE conferences in the area of fuzzy sets and neurocomputing. He is intensively involved in editorial activities. He is the Editor-in-Chief of *Information Sciences* and the Editor-in-Chief of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A. He currently serves as an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS and is a member of a number of editorial boards of other international journals. In 2007, he received a prestigious Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Council. He was a recipient of the IEEE Canada Computer Engineering Medal 2008. In 2009, he received a Cajastur Prize for Soft Computing from the European Centre for Soft Computing for "*pioneering and multifaceted contributions to Granular Computing*."

Lukasz A. Kurgan (M'02) received the M.Sc. degree in automation and robotics (with honors; recognized by an Outstanding Student Award) from the AGH University of Science and Technology, Krakow, Poland, in 1999 and the Ph.D. degree in computer science from the University of Colorado, Boulder, in 2003.

He is an Associate Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. His research interests include applications of machine learning in structural bioinformatics. He authored and coauthored several machine learning algorithms and methods for high-throughput prediction of protein and short RNA structure and function. He published more than 70 peer-reviewed journal articles. He currently serves as an Editor of *PLoS ONE*, *BMC Bioinformatics*, *Neurocomputing*, *Open Proteomics Journal*, *Journal of Biomedical Science and Engineering*, *Open Bioinformatics Journal*, and *Protein and Peptide Letters* journals.

Dr. Kurgan has been a member of numerous conference committees in the area of bioinformatics, data mining, machine learning, and computational intelligence.