University of Alberta

LEARNING AND AGGREGATION OF FUZZY COGNITIVE MAPS – AN EVOLUTIONARY APPROACH

by

Wojciech Stach

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

Fall 2010
Edmonton, Alberta

Examining Committee

Lukasz Kurgan, Electrical and Computer Engineering

Witold Pedrycz, Electrical and Computer Engineering

Scott Dick, Electrical and Computer Engineering

Marek Reformat, Electrical and Computer Engineering

Jozef Szymanski, School of Mining & Petroleum Engineering

Gopal Achari, Department of Civil Engineering, University of Calgary

In gratitude to Dr. Lukasz Kurgan for your profound impact on the course of my life.

Your encouragement and dedication has been a gift throughout my Ph.D. program.

Thank you for the meaningful time you spent with me to offer true help and support.

Abstract

Fuzzy Cognitive Maps (FCMs) are a widely used, neuro-fuzzy based qualitative approach for the modeling of dynamic systems, which allow for both static and dynamic analyses. They are capable of modeling complex systems with nonlinearities and unknown physical behaviour. FCMs describe a given system by means of concepts connected by quantified cause-effect relationships. This dissertation contributes to the subject of computer-driven generation of FCMs that can be used to perform an accurate dynamic analysis of the modeled system. The dynamic analysis provides insights into the degree of presence, and dependencies between the concepts in successive iterations of the simulation of a given FCM model. Such simulation studies could be used to analyze "what-if" scenarios in the context of decision support and to perform time series predictions. Two research directions within the framework of FCM development, which concern the *learning* of FCMs from historical data and an *aggregation* of FCMs that were proposed by multiple experts, are investigated. Several new automated computational methods for data-driven learning and aggregation of FCMs are introduced and empirically evaluated. These methods utilize real-coded genetic algorithms (RCGA)-based optimization. This choice of the optimization vehicle was motivated by their well-documented efficiency in searching large and continuous search spaces, which are inherent to our problem. Experimental evaluation demonstrates that the proposed RCGA-based learning method outperforms modern existing approaches when the dynamic analysis is considered. A novel divide and conquer-based learning strategy to improve scalability of the RCGA approach, is also proposed. This strategy is shown to be competitive or even better than solutions based on the parallelization of the underlying genetic algorithm. The RCGA-based learning method is further extended to provide improved FCMs when the number of connections of the map is known a priori. Experimental evaluation shows that the density-based learning method outperforms the generic RCGA-based approach when using a relatively accurate density estimate, and that both methods are equivalent when the estimate is inaccurate. In addition, a novel method for the aggregation of multiple input FCMs, is proposed. When compared to existing aggregation approaches, this method provides solutions that are more accurate when dynamic analysis is the objective.

Acknowledgments

First and foremost, I would like to express my deep gratitude to my supervisors Dr. Lukasz Kurgan and Dr. Witold Pedrycz, for their friendly, continuous, and valuable guidance during my entire Ph.D. program.

I would like to thank the members of my examination committee, Dr. Scott Dick, Dr. Marek Reformat, Dr. Jozef Szymanski, and Dr. Gopal Achari, for their helpful and constructive comments.

I would like to extend my gratitude to my fellow lab members for their collaboration, support, and time spent together. Special thanks to my dear friend, Dr. Rafal Rak, who has accompanied me through my entire academic career.

I am greatly indebted to Famida Devji for her detailed and prompt proofreading this dissertation.

I would like to thank my parents, family and friends for their unconditional and continuous support in each dimension of my life.

Last but not least, I would like to thank my wife, Monika, for her understanding, encouragement, and for making me a better person.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Fuzzy Cognitive Maps (Kosko, 1986) are a convenient tool for qualitative modeling and simulation of dynamic systems. They describe a given system as a collection of concepts connected by cause-effect relationships. FCMs are useful for performing both *static* and *dynamic* analyses of the modeled system. The former type of analysis includes identification of cycles to uncover nontrivial relationships between concepts, calculations of the model density to obtain an indication of its complexity, and an analysis of the importance of individual nodes. On the other hand, dynamic analysis is concerned with the simulation of the FCM model and provides insights into the degree of presence, interactions and dependencies between the concepts in successive iterations. Consequently, it allows exploring "what-if" scenarios and is used to support decision making (Stylios and Georgopoulos, 2008) and/or predict future states (Stach et al., 2008a) of the system. FCMs have been recognized as a useful and flexible technique in problem solving where many decision variables are causally interrelated. Their main advantages include easy model representation, simulation, and adaptability to a given domain (Aguilar, 2005). The wide-spread, cross-disciplinary and numerous applications of FCMs show that there is an appeal for this modeling technique (see Table 2.1).

Designing an accurate FCM for a given system is a challenging task (Aguilar, 2005). For many years the only available solution was to utilize human expertise in the area of application. These *expert-based* methods have gained their popularity due to the relatively simple FCM representation, which makes it possible to manually draw its graph-based structure using only a pencil and a sheet of paper. FCMs allow for the aggregation of models in case a number of experts develop different models of the same

system; this in turn potentially increases the credibility of the combined FCM model. Existing methods for aggregation of FCMs are based on structural properties of individual FCMs. This may lead to inaccuracies when dynamic analysis is considered. To alleviate this problem, in this dissertation we introduce a novel method to aggregate FCMs.

Although the area of expert-based methods for developing FCMs has been well-established, these methods are inevitably associated with a number of drawbacks, such as experts' bias and difficulties in handling large and complex systems. Recently, a few computational learning methods to support (*semi-automated methods*) or to replace experts' (*fully-automated methods*) have been proposed (Stach et al., 2010). These methods use data available as time-series and a learning algorithm to develop an FCM model. However, the existing solutions, based mostly on Hebbian-learning, are not able to provide accurate solutions when considering dynamic simulations. To this end, we propose a new method for learning FCMs from data that generates maps more accurate in terms of simulations when compared to other existing approaches.

The empirical evaluation of the methods for the FCMs development is challenging. A comprehensive evaluation should cover both static (concerning structure) and dynamic (concerning simulations) map properties. The only available gold standards for structural comparison are maps developed by human experts. These models, however, are vulnerable to bias, since they are usually not sufficiently validated using real data. Therefore, in our evaluation we addressed this issue by performing two types of experiments. The first group includes learning FCMs from data generated from (potentially biased) real or synthetic FCM models, whereas the second group utilizes real data to build FCMs in the areas where such maps were not yet developed. The first group of experiments allow a multidimensional evaluation that concerns both structures and simulations of the maps, while the second one validates the accuracy of the dynamic modeling performed on real data. Additionally, the latter experiment allows for a comparison between the accuracy of FCMs modeling and other existing modeling techniques.

1.2 Goals and thesis statements

In this work several new computational intelligence-based methods for data-driven and automated learning and aggregation of FCMs are proposed and empirically evaluated. It can be noted that they provide improved quality of dynamic simulations, when compared with models derived with current automated methods.

More specifically, we introduce (1) a new automated learning method, which is based on genetic optimization, to develop FCMs from data that aims to provide accurate FCM models for dynamic analysis; and (2) a new approach to aggregate FCMs that preserves dynamic properties of individual input FCM models. Moreover, we extend and improve the proposed method for data-driven learning of FCMs by (3) utilizing additional a priori knowledge of the structure of the desired map and (4) applying a divide and conquer paradigm to improve efficiency in handling large and complex systems that involve a few dozens of concepts.

To summarize, this dissertation presents the following thesis statements:

1. Learning Fuzzy Cognitive Maps from data using genetic optimization is an efficient approach to obtain high quality models in terms of their dynamic properties.

    a. Using additional a priori information of the map structure results in further improvements in the quality of the derived model

    b. Using the divide and conquer strategy is an effective method of improve scalability of the proposed learning method

2. The aggregation of Fuzzy Cognitive Maps using genetic optimization is an efficient method to obtain a model that preserves dynamic properties of the input FCM models.

1.3 Outline

The dissertation is organized as follows: Chapter 2 includes all required background information. In particular, it introduces the theoretical foundations of Fuzzy Cognitive Maps, their development practices, and highlights selected areas of applications. Chapters 3 and 4 present the main contributions and introduce and empirically validate the new learning method and its extensions, as well as the new aggregation method. The dissertation is summarized in Chapter 5 by presenting results for a case study, enumerating contributions and findings as well as outlining future directions.

Chapter 2

Background and Related Work

The subject of dynamic system modeling has been investigated using various techniques (Campello and Amaral, 2003). Considering the method of knowledge representation, two main groups of modeling approaches can be distinguished. The first group concerns *quantitative* techniques, which can be applied to both well-understood systems, such as mathematical programming techniques of operations research, and to less well-defined ones, such as statistical regression based methods. However, significant effort and specialized knowledge from outside of the domain of interest, which involves linear or nonlinear differential or difference equations, is required to develop these models. Furthermore, they may not provide satisfactory performance in their application to complex systems, such as those with strong nonlinearities, significant feedback, and unknown physical behaviour (Pedrycz, 1995; Aguilar, 2005).

The techniques from the second group, which use a *qualitative* approach, provide an alternative solution to system modeling and are free from the limitations of the first group. Numerous qualitative methods have been introduced in the framework of the Artificial Intelligence area (De Kleer and Brown, 1984). Modeling techniques from this group are focused on providing a qualitative description of a given system (Pedrycz, 1995), in which the set of the system's variables is described with a set of modeling landmarks and expressed by a limited set of descriptors, such as positive, zero, negative, etc. This approach has been extended by applying fuzzy sets to represent these landmarks (D'Ambrosio, 1989). Two classes of models have gained significant importance within this group of modeling: *fuzzy models* (Yager and Filev, 1994; Pedrycz, 1996; Hellendoorn and Driankov, 1997), and *neural networks* (Kosko, 1992; Haykin, 1999). Fuzzy Cognitive Maps (Kosko, 1986) fall into the second group of modeling techniques

and exhibit many appealing properties, such as easy of use, flexibility, and adaptability to a given domain (Aguilar, 2005).

2.1 Overview of Fuzzy Cognitive Maps

2.1.1 Historical perspective

Cognitive Maps, which are precursors of Fuzzy Cognitive Maps, were first introduced in 1976 by a political scientist named Robert Axelrod (Axelrod, 1976). They were presented as a tool to help analyze systems that include concepts interrelated by complex relationships. In particular, Cognitive Maps were applied to represent social scientific knowledge. A Cognitive Map may be expressed in a simple digraph form consisting of nodes and edges whereby each node corresponds to a concept or variable relevant to a domain of application. This set of nodes is connected by directed edges that represent mutual relationships among concepts. Each edge is associated with a positive or negative sign that expresses a specific type of relationship. A positive sign for an edge from a node *A* to a node *B* indicates promoting influence on *B* exerted by *A*. It means that an increase in the value of node *A* will lead to an increase in the value of node *B*, and vice-versa. A negative sign for an edge from node *A* to node *B* reflects an inhibitory type of relationship. It describes the situation where increasing the value of *A* leads to decreasing the value of *B*. If there is no edge between the given two concepts, it indicates that there is no direct cause-effect correlation linking these concepts. However, the results of this modeling technique turned out to be insufficient to describe more complex systems due to the limited representation of relations. Typically, causality is not Boolean (two-valued, yes-no) in real-life systems, i.e., the relationships are too complex to be described with only a sign. This was the motivation to extend the theory of Cognitive Maps.

Ten years later, in 1986, Kosko (Kosko, 1986) introduced Fuzzy Cognitive Maps (FCMs). The fundamentals of this method, in terms of model representation as a set of concepts connected by relationships, are the same as in the Axelrod's approach. In comparison to Cognitive Maps, the most significant generalization of FCMs lies in the way causal relationships between concepts are represented. Instead of using only a sign, each edge is associated with a number (weight) that defines the strength of a given causal relationship. FCMs describe relationships in fuzzy terms, such as weak, medium, strong, or very strong. In other words, a weight associated with directed edge from a node *A* to

node *B* quantifies "how much" concept *A* causes *B*. The strength of a relationship between two nodes (i.e., weight value) is usually normalized on the [–1, 1] interval. The value of –1 represents maximum negative, whereas value of the +1, maximum positive influence. Zero denotes no causal effect. Other values correspond to intermediate levels of influence. As a result, a FCM is fully described by a set of nodes (concepts) and edges (cause-effect relationships), which are represented by weights. Apart from the graph representation, for computational purposes, FCMs can be equivalently defined by a square matrix, called a *connection matrix*, which stores all weight values of edges between corresponding concepts that are represented by rows and columns. Therefore, a system with *N* nodes can be represented by an *N* x *N* connection matrix.

2.1.2 Mathematical foundations

Mathematically, a Fuzzy Cognitive Map *F* is a 4-tuple (C, **E**, **C**, f) (Kosko, 1986), where

1. $C = \{C_1, C_2, ..., C_N\}$ is the set of *N* concepts forming the nodes of a graph.

2. $\mathbf{E} : (C_i, C_j) \rightarrow e_{ij}$ is a function associating e_{ij} with a pair of concepts (C_i, C_j), with e_{ij} equal to the weight of edge directed from C_i to C_j where $e_{ij} \in K = [-1,1]$. Thus, $\mathbf{E}(NxN) = e_{ij} \in K^{NxN}$ is the connection matrix.

3. $\mathbf{C} : C_i \rightarrow C_i(t)$ is a function that associates each concept C_i with the sequence of its activation degrees such as for $t \in N, C_i(t) \in L$ 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* is a transformation function, which includes recurring relationship on $t \geq 0$ between $\mathbf{C}(t+1)$ and $\mathbf{C}(t)$

$$\forall i \in \{1, ..., N\}, C_j(t+1) = f\left(\sum_{i=1}^{N} e_{ij} C_i(t)\right) \tag{2.1}$$

where $f : \mathbf{R} \rightarrow L$ is a transformation function of **R** to the set of activation degrees *L* normalizing the activation.

Formula 2.1 describes a functional model of FCMs, which is used to perform simulation of the system dynamics. Simulation consists of computing the 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. The value of a given node is calculated from the preceding iteration of values of nodes, which exert influence on the given node through a cause-effect relationship (nodes that are connected to the given node).

The *transformation function* is used to reduce an unbounded 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). The three most commonly used transformation functions include:

- Bivalent

$$f(x) = \begin{cases} 0, & if \quad x \le 0 \\ 1, & if \quad x > 0 \end{cases} \tag{2.2}$$

- Trivalent

$$f(x) = \begin{cases} -1, & if \quad x \le 0.5 \\ 0, & if \quad -0.5 < x < 0.5 \\ 1, & if \quad x \ge 0.5 \end{cases} \tag{2.3}$$

- Logistic

$$f(x) = \frac{1}{1 + e^{-Mx}} \tag{2.4}$$

where M is a parameter used to determine the degree of fuzzification of the function.

A comparison of different transformation functions for FCMs was recently carried out by Tsadiras (Tsadiras, 2008). It includes performance evaluations of different functions, with

respect to the nature of the problem, the required representation capabilities of the problem and the level of inference required by the case. Overall, the *logistic function* has been found to be the most flexible and effective in providing an accurate description of the dynamics of the modeled system. Therefore, in this dissertation all experiments are carried out with the logistic transformation function.

2.1.3 Development strategies

There are two mainstream techniques for developing FCMs. The first group denoted as *expert-based methods* includes techniques that exploit only human knowledge. For a long time this was practically the only way to establish FCMs, mainly because of the lack of automated or semi-automated approaches that would support this process. Recently, several attempts were made to develop *computational methods*. They aim to substitute or help the expert and learn the model's structure in either an automated or semi-automated fashion using historical data (Stach et al., 2005a; Stach et al., 2010).

Expert-based methods for Fuzzy Cognitive Maps development rely entirely on human expertise and domain knowledge, and therefore are classified as *deductive modeling*. The relatively simple model representation makes it possible to simply manually draw the graph that corresponds to an FCM using only a pencil and a sheet of paper. The experts are also required to have a rudimentary knowledge of the FCM theory to understand the meaning of the weights and the direction of the causal effects. In order to increase the credibility of the model, a group of experts as opposed to a single person may be involved in the development process. Experts can work together or design maps individually to represent their understanding of a given system. In the latter case, the individual maps can be combined into a single model.

The expert-based development of FCMs usually consists of the following three steps (Kosko, 1986; Khan and Quaddus, 2004):

1. Identification of important concepts.

2. Identification of causal relationships among these concepts.

3. Estimation of the strength of the causal relationships.

In the first step, the decision of which available concepts should be included in the model must be made. The most intuitive strategy is to create a list of all relevant concepts and remove the insignificant ones. In the second step, all cause-effect direct relationships amongst the remaining concepts have to be identified, including their directions. Usually this is accomplished by focusing on one pair of concepts at a time, and hence the expert is relieved of the task of coming up with hidden or indirect cause–effect relationships. These relationships become apparent later through analyses carried out using the completed FCM. These first two steps result in a structural design which consists of a graph with nodes and directed edges.

The main challenge in expert-based development of FCMs is to accurately estimate the strength of the relationships. We note that the number of weights exhibits quadratic growth with the number of concepts, which may lead to difficulties in developing maps with several dozens of concepts. Following the original paper (Kosko, 1986), each relationship strength value (weight) is expressed by a real number from the [-1,1] interval. The value of 0 denotes no relationship and is implicitly assigned at the end of the second step. Higher absolute values represent stronger relationships, whereas the sign defines the type: promoting (positive numbers) or inhibiting (negative numbers). Theoretically, each weight can take on an infinite numbers of values. Consequently, this step is potentially susceptible to subjective interpretation of a given expert. A common practice to facilitate the estimation of the weight values is to first describe each relationship by a linguistic term and then to transform these terms into numerical values. The corresponding work can be divided into the following three steps (Kosko, 1986; Khan and Quaddus, 2004):

1. Determining the sign of each relationship.

2. Describing each relationship by means of linguistic terms, e.g. weak, medium, strong and very strong.

3. Mapping the linguistic terms to numerical values, e.g. weak to 0.25, medium to 0.5, strong to 0.75, and very strong to 1.0.

The use of the linguistic expressions to describe the degrees of causality in relationships allows the experts to avoid the difficult task of specifying the precise numerical values before a draft model is established. Additionally, analytical procedures, such as

Analytical Hierarchy Process (Saaty, 1980), may be helpful in finding the numerical values used in the last step of the weight estimation procedure.

Computational methods for Fuzzy Cognitive Maps development utilize available data and a learning algorithm to develop or to support the development of the corresponding model. Since they use available data to learn the model, they are classified as *inductive modeling*. A number of methods have been proposed within this research framework. The detailed description of these approaches is included in Section 2.2.1, where the related background work is presented.

2.1.4 Areas of applications

Fuzzy Cognitive Maps have been utilized in numerous research and industrial areas. They can be roughly divided into applications in medicine, earth and environmental sciences, engineering, and economics, business and management. The breadth and number of applications provide motivation for the proposed research and thus a brief summary of selected applications, which are organized in chronological order for each application domain, is presented in the next few subsections.

Medicine

Georgopoulos et al. (Georgopoulos et al., 2001; Georgopoulos et al., 2003; Georgopoulos et al., 2005; Georgopoulos and Stylios, 2008; Stylios and Georgopoulos, 2008) applied FCMs to help with the diagnosis of Specific Language Impairment (SLI) and to support the speech therapist in the diagnosis process. Stylios et al. (Stylios et al., 2001) proposed a model based on FCMs that utilized the evaluation of cardiotocographic signals with other physiological data to create an advanced decision-making support system. Papageorgiou et al. (Papageorgiou et al., 2002) used FCMs in radiotherapy treatment planning to estimate the radiation dose. The follow-up of this work (Papageorgiou et al., 2003a) was published in 2003 and included a more advanced hierarchical approach to the same problem that, in addition to the previous model, supervised and evaluated the radiotherapy process prior to treatment execution. Castelfranchi et al. (Castelfranchi et al., 2003) performed a case study of trust evaluation between a doctor and medical automatic system. Innocent and John (Innocent and John, 2004) proposed a framework based on FCMs to consider computer aided medical diagnoses in a clinical context that supports the estimation of the stage of a disease. The follow-up of this work (John and

Innocent, 2005) included a description of a fuzzy process that is applied to the diagnosis of two similar diseases. Papageorgiou et al. (Papageorgiou et al., 2006) used FCMs to develop an advanced diagnostic method for urinary bladder tumor grading. In 2007, Giles et al. (Giles et al., 2007; Giles et al., 2008) proposed an FCM-based approach to address the issue of diabetes among Aboriginal communities in Canada, whereby current conventional methods of treatment are ineffective. Papageorgiou et al. (Papageorgiou et al., 2007) introduced a method using FCM to support medical decision-making incase different types of medical and/or clinical data are available. The same research group proposed a new method of brain tumor characterization including an accurate identification of its grade (Papageorgiou et al., 2008a) as well as an FCM based decision-making support system for thyroid disease (Papageorgiou et al., 2008b). One of Papageorgiou and colleagues' recent works includes FCM's application to model the problem of pulmonary infections during a patient's admission into the hospital or Intensive Care Unit (Papageorgiou et al., 2009).

Earth and Environmental Sciences

Banini and Bearman (Banini and Bearman, 1998) presented an FCM approach to study the interaction between various factors that affect slurry rheology. Satur and Liu (Satur and Liu, 1999a; Satur and Liu, 1999b; Liu and Satur, 1999) used FCMs to develop a supportive tool to analyze data from the Geographic Information System (GIS). Hobbs et al. (Hobbs et al., 2002) applied the FCM as a method to study the objectives of ecological rehabilitation focusing on the analysis of responses from the ecosystem to management. Fons et al. (Fons et al., 2003; Fons et al., 2004) performed an FCM-based analysis of the impact of establishing an eco-industrial park. Ozesmi and Ozesmi (Ozesmi and Ozesmi, 2003) used FCMs to support the management plan for establishing a lake. The same authors (Ozesmi and Ozesmi, 2004) proposed a multi-step FCM approach to create ecological models with both expert and the residential people's expertise. Giordano et al. (Giordano et al., 2005) applied FCMs to describe conflicts of interests in the area of water management. FCMs were used to analyze the decrease in pollution created by products' recycling. Pan et al. (Pan et al., 2005) developed this disassembly model and generated recycle sequences for a product. A shallow lake ecosystem model was presented and analyzed by Tan and Ozesmi (Tan and Ozesmi, 2006) in order to help understand its complex structure and dynamics. In 2007, an FCM model to study the impact of deteriorating pipes on water quality so that decision making process for pipe renewal

could be supported (Sadiq et al., 2007) was developed. An emergency management plan for a nuclear power plant using an FCM modeling was presented by Espinosa-Paredes and colleagues in 2008 (Espinosa-Paredes et al., 2008) to support the decision making process during abnormal situations in a nuclear power plant. In 2009, the same research group used an FCM to model a risk scenario for a nuclear power plant (Espinosa-Paredes et al., 2009). Recent work in this area, introduced by Kok (Kok, 2009), discusses modeling and predicts the dynamics of deforestation in the Brazilian Amazon with the use of FCMs. Gras and colleagues (Gras et al., 2009) used an FCM to simulate an evolving predator-prey ecosystem.

Engineering

In one of the first applications of FCMs in engineering, Styblinski and Meyer (Styblinski and Meyer, 1988) explored their usefulness to the qualitative electrical circuit analysis. Pelaez and Bowles (Pelaez and Bowles, 1995; Pelaez and Bowles, 1996) applied FCMs to describe the failure modes of a system. Lee et al. (Lee et al., 1996) proposed a system that performed on-line fault diagnoses with the use of FCMs. Another example using FCM technique, fault management and diagnosis, was authored by Ndousse and Okuda (Ndousse and Okuda, 1996). Subsequent applications are related to the robotics field. Hashimoto and Yamaguchi, followed by Ando et al. (Hashimoto and Yamaguchi, 1997; Hashimoto, 2000; Ando et al., 2001), proposed an application of FCMs to describe a system with an agent. As an example they used a mobile robot and modeled its emotions. Modeling of emotions in autonomous agents and robotics with the use of FCMs was also studied by Ayesh (Ayesh, 2004) and Golmohammadi and colleagues (Golmohammadi et al., 2007). Pipe et al. (Pipe et al., 1995) used FCMs as a supportive tool in discovering a near-optimal path between start and goal positions of a particular maze for a mobile robot. Another application of FCMs to robot systems was presented by Subramanian and Dagli (Subramanian and Dagli, 2003). They used this technique to simulate the positioning of the robots and perform simple robots movements. Returning back to the mid 1990s, in 1997 Stylios et al. (Stylios et al., 1997) presented an application of FCMs to distributed control systems. The base modeled system included a tank and three valves that controlled the amount of liquid in the tank. Based on this initial work, several follow-ups were presented by the same authors (Stylios and Groumpos, 1999; Groumpos and Stylios, 2000; Stylios and Groumpos, 2004), which demonstrated the usefulness of FCM modeling in more complex control process problems. In particular, their work included

the modeling of the supervisor of a control system and examples of its application in real-life systems, such as a heater exchanger. Macedo (Macedo, 1999) introduced a new computer-aided design system, using FCMs, that supports organizational re-engineering. Satish Jamadagni (Satish Jamadagni, 2000) employed FCMs to describe mobility management issues in telecommunication networks. Lee and Han (Lee and Han, 2000) applied FCMs to help with the design of Electronic Data Interchange (EDI) controls, whereas in follow-ups to this work (Lee et al., 2004; Lee and Lee, 2007), the evaluation of the performance of EDI with the use of FCMs was carried out. An example of an intelligent intrusion detection system in a computer network that exploits FCMs as a supportive tool for decision making was proposed by Siraj et al. (Siraj et al., 2001). Xin et al. (Xin et al., 2003) used FCMs in the same area to detect complex network attack scenarios, whereas Mu et al. (Mu et al., 2004) presented an FCM-based decision support mechanism that set up the intrusion response plan. Lee et al. (Lee et al., 2002) applied FCMs in the field of web-mining as part of a novel web-mining inference amplification mechanism. Christova et al. (Christova et al., 2003) proposed an integrated hierarchical method based on FCMs, to model industrial plants. Stach and Kurgan (Stach and Kurgan, 2004; Stach et al., 2004a) applied FCMs in the software project management domain. This modeling technique was used to describe and analyze factors that affect the pace of work progress during software project development. In 2006, Xirogiannis and colleagues (Xirogiannis et al., 2006) presented a decision support framework for assisting the process of urban design. Several examples of FCM application as well as guidelines on how to use this technique in Engineering and Technology Management are discussed by Jetter (Jetter, 2006). Rodriguez-Repiso and co-workers (Rodriguez-Repiso et al., 2007) developed an FCM model to classify and evaluate the indicators of success in IT projects. An analysis of electronic circuits troubleshooting using FCMs was presented by Perusich (Perusich, 2007*;* Perusich, 2008) whereas mobile robot troubleshooting was discussed by Yeap et al. (Yeap et al., 2008). Risk management in a distributed system using an example of distributed databases was modeled and analysed by de Korvin et al. (De Korvin et al., 2007). Distributed wireless peer to peer networks were modeled with an FCM by Li et al. (Li et al., 2009) to help develop a novel peer selection scheme.

Economics, Business and Management

Kardaras and Karakostas (Kardaras and Karakostas, 1999) studied the application of FCMs as an alternative method to existing Strategic Planning of Information Systems.

Koulouriotis et al. (Koulouriotis et al., 2001a) performed a stock market analysis and prediction with the use of FCMs, which were used to construct a stock market model. Jetter (Jetter, 2003) applied FCMs to model the early phases of a new product development, which was used to support decision making. Borrie and Ozveren (Borrie and Ozveren, 2004) utilized FCMs to describe a dynamic market environment using an example of the electricity market in the United Kingdom. Another illustration of using FCM-based methodology in economics was presented by Carvalho and Tome (Carvalho and Tome, 2004) who developed a model of tax rate evolution in Europe. Xirogiannis and Glykas (Xirogiannis and Glykas, 2004) developed an FCM that supplemented the strategic planning and business analysis phases of typical redesign projects. The same authors (Glykas and Xirogiannis, 2005) applied FCMs as a tool for modeling factors related to geographically dispersed financial enterprises that influence overall business performance. Grant and Osei-Bryson (Grant and Osei-Bryson, 2005) investigated how changes in one area of a company affect other areas by using FCM to model impacts and interactions. Modeling a system to support e-business strategy formulation with the use of FCMs was presented by Xirogiannis and Glykas (Xirogiannis and Glykas, 2007). An application of FCMs to business planning modeling and analysis was discussed by Niskanen (Niskanen, 2007). FCMs were applied to model a supply chain risk analysis in order to achieve continuous customer satisfaction by Feyzioglu et al. (Feyzioglu et al., 2007). Enterprise resource planning tools using FCMs was proposed by Bueno and Salmeron (Bueno and Salmeron, 2008) to support decision making in a company. FCMs were also utilized to model and forecast stock markets by Froelich and Wakulicz-Deja (Froelich and Wakulicz-Deja, 2008). Schlager and Pernul (Schlager and Pernul, 2008) studied trust and its role in e-commerce by developing and analysis of a corresponding FCM model. In 2009, an application of FCMs to credit risk evaluation was proposed by Buyukozkan and Vardaloglu (Buyukozkan and Vardaloglu, 2009), whereas Noori et al. (Noori et al., 2009) applied FCMs as a tool for modeling factors related to organizational conflict management.

Other Application Areas

Applications of FCMs are not limited to the areas listed in the last four subsections. They have also been used in other fields. Chen and Huang (Chen and Huang, 1995a; Chen and Huang, 1995b) studied guard heuristic in Chinese chess using FCMs. Perusich (Perusich, 1996; Perusich and McNeese, 2005) investigated the application of FCMs as an

intelligent analyst incase of a terrorist attack. Several publications presented applications of FCM in domain of education (Cole and Persichitte, 2000; Georgiou and Makry, 2004; Hossain and Brooks, 2008; Salmeron, 2009). FCMs were also applied in political sciences (Tsadiras et al., 2001; Andreou et al., 2005), pattern recognition (Hu et al., 2004; Pajares and De la Cruz, 2006; Papakostas et al., 2008), document classification (Peng et al., 2008; Zhou and Zhang, 2008), protein structure prediction (Kurgan et al., 2007; Nguyen et al., 2008), time series prediction (Stach et al., 2008a) which is described in details in Section 3.4, and military planning (Yaman and Polat, 2009). Most recent applications include modeling of agro-ecosystems (Rajaram and Das, 2010), cryovolcanic activity on Titan (Furfaro et al., 2010), and defense planning in combating terrorism (Akgun et al., 2010).

Table 2.1 presents reported applications of Fuzzy Cognitive Maps. The table includes information about the area of application, the number of corresponding journal and conference papers, and the years of publications.

Table 2.1: Summary of FCM applications

Area	Journal papers	Conference papers	Years (number of publications)
Medicine	11	8	2001, 2002 (x2), 2003 (x3), 2004, 2005 (x3), 2006, 2007 (x2), 2008 (x4), 2009
Earth and Environmental Sciences	13	4	1998, 1999 (x3), 2002, 2003 (x2), 2004 (x2), 2005 (x2), 2006, 2007, 2008, 2009 (x3)
Engineering	16	21	1998, 1995 (x2), 1996 (x3), 1997 (x2), 1999 (x2), 2000 (x5), 2001 (x2), 2002, 2003 (x3), 2004 (x6), 2006 (x2), 2007 (x5), 2008 (x2), 2009
Economics, Business, and Management	6	9	1999, 2001, 2003, 2004 (x3), 2005 (x2), 2007 (x2), 2008 (x3), 2009 (x2)
Others	13	9	1995 (x2), 1996, 2000, 2001, 2004 (x2), 2005 (x2), 2006, 2007, 2008 (x6), 2009 (x2) , 2010 (x3)

The above summary of FCM applications indicates that researchers use this modeling technique in various disciplines demonstrating its adaptability and flexibility. The number of recent publications shows that there is a continuous interest in Fuzzy Cognitive Maps, which motivates and rationalizes further research in this area.

2.1.5 Example of FCM – case study

A real-life FCM application of this particular modeling technique in Mineral Engineering was selected as a case study. More specifically, FCMs were used to model factors that affect slurry rheology (Banini and Bearman, 1998). This choice was made due to the fact that the original reference presents an interesting application of FCM to a relatively large system that includes thirteen concepts and is the only contribution that provides all necessary information to define a case study that can be utilized consistently throughout the entire dissertation. In particular, this map was developed by an aggregation of input maps developed by three experts, whereby the authors provide all the relevant connection matrices. Moreover, the design of the map follows typical practices used for FCM developments (Aguilar, 2005). Following the original paper, there is a step-by-step description of the design procedure used in the original reference.

Defining concepts

Based on their knowledge from the area of application and previous works, three experts agreed on the following 13 concepts that define the modeled system: gravity (C1), mechanical properties of particles (C2), physicochemical interaction (C3), hydrodynamic interaction (C4), effective particle concentration (C5), particle-particle contact (C6), liquid viscosity (C7), effective particle shape (C8), effective particle size (C9), temperature (C10), inter-particle attraction (C11), floc/structure formation (C12), and shear rate (C13). This set represents important factors that affect the slurry rheology process and that interact with each other through cause-effect relationships.

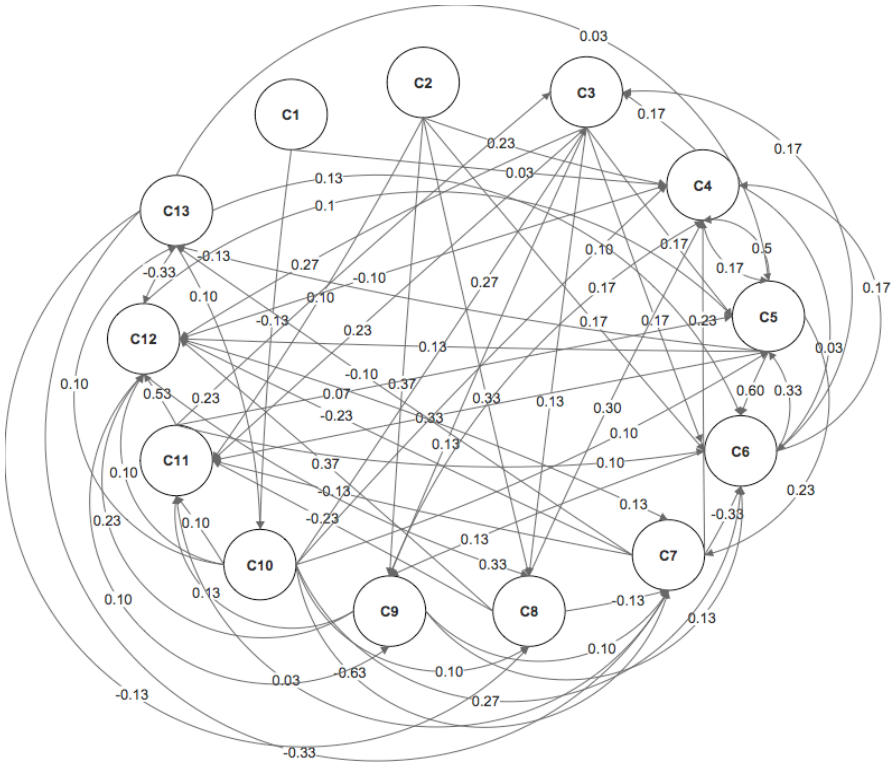
Defining relationships

Each expert was asked to draw an FCM model that consists of these 13 concepts and to establish the relationships between the concepts based on his or her beliefs. Consequently, three models have been proposed (Banini and Bearman, 1998). The experts chose one of the twenty one values (-1,-0.9,…,0,…0.9,1) to define each

relationship. Since the model consists of 13 concepts, each expert had to define 169 relationships. The exact procedure is not explained, though the paper includes a statement that they consulted their opinions at the local Mineral Research Center. The experts' maps were sparsely connected and they included 18, 20, and 44 non-zero relationships, respectively.

FCMs aggregation

A commonly used method for aggregating FCMs that averages relationship strengths across all individual models (see Section 2.2.2 for details) was used to obtain the combined model. The models from the three experts were aggregated to obtain a single, final model of the slurry rheology system, see Figure 2.1.



	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
C1				0.03							-0.13		
C2				0.23		0.17		0.33	0.37		0.1		
C3					0.17	0.17		0.13	0.13		0.23	0.27	
C4			0.17		0.17	0.03						-0.1	
C5				0.5		0.6	0.23				0.33	0.13	-0.13
C6			0.17	0.17	0.33				0.13				
C7				0.23		-0.33					-0.13	-0.23	-0.10
C8				0.3			-0.13				-0.23	0.37	
C9				0.17		0.13	0.1				0.13	0.23	
C10			0.27	0.1	0.1	0.27	-0.63	0.1			0.1	0.1	0.1
C11			0.23		0.07	0.1	0.03					0.53	
C12					0.1		0.13	0.33	0.1				
C13					0.03	0.13	-0.33	-0.13		0.1		-0.33	

Figure 2.1: Fuzzy Cognitive Map model for the slurry rheology system along with its connection matrix. Empty cells indicate values of zero.

FCM analysis

The static analysis of the final model included discovering concepts that play an important role in the modeled system (Tsadiras et al., 2001). This was performed by taking the sum of absolute values of all non-zero relationships that enter (incoming edges) as well as leave (outgoing edges) a given concept. The following concepts were determined as the most important in the system: C4, C5, C6, C7, C8, C11, and C12, since the values of the corresponding sums were greater than those of the other concepts.

The dynamic analysis was aimed at performing what-if scenarios by simulating the system from different initial conditions. For instance, by simulating the model from the initial state where C1=1 and all other concepts are reset to the value zero, the authors investigated the system behaviour in case of gravity change (C1). Since the system headed to a stable state with most active concepts C1, C4, C5, C8, C9, C12, the conclusion was that these concepts will be mostly affected by the change.

The results obtained for both types of analyses agreed with previous works and experiments in this area (Banini and Bearman, 1998).

2.2 Related work

2.2.1 Computational methods for FCMs development

In one of the first attempts, Dickerson and Kosko proposed a simple *Differential Hebbian Learning (DHL)* (Dickerson and Kosko, 1993; Dickerson and Kosko, 1994) method, which is based on Hebbian's theory (Hebb, 1949). During DHL learning, the values of weights are iteratively updated until the desired structure is found. In general, the weights of outgoing edges for each concept in the connection matrix are modified only when the corresponding concept value changes,

$$e_{ij}(t+1) = \begin{cases} e_{ij}(t) + c_t\left[\Delta C_i(t)\Delta C_j(t) - e_{ij}(t)\right] & if \quad \Delta C_i(t) \neq 0 \\ e_{ij}(t) & if \quad \Delta C_i(t) = 0 \end{cases} \tag{2.5}$$

where e_{ij} denotes the weight for relation from concept C_i to C_j, $\Delta C_i(t)$ 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 whereby values generally decrease as the learning progresses. The main drawback of this learning method is that the formula updates the weights between each pair of concepts taking into account only these two concepts and ignoring the influence from others.

An improved version of DHL learning was introduced by Huerga (Huerga, 2002). The new algorithm, called *Balanced Differential Algorithm (BDA)* eliminates one of the limitations of the DHL method by taking into account all concept values that change simultaneously when the weights are updated. More specifically, the modified formula for $e_{ij}(t+1)$ takes into consideration the changes in all concepts if they occur at the same iteration and have the same direction. An empirical comparison between DHL and BDA demonstrates that the latter method improves the quality of the learned maps (Huerga, 2002). On the other hand, the BDA algorithm was applied only to binary FCMs, i.e., maps with binary transformation functions, which limits its application areas.

One year later, Papageorgiou and colleagues introduced a *Nonlinear Hebbian Learning (NHL)* algorithm (Papageorgiou et al., 2003b). While this algorithm originates from the same learning principles, it uses a nonlinear extension to the basic Hebbian rule (Oja et al., 1991) by introducing a modified weight update formula. The formula updates each weight proportionally to the product of the corresponding concepts' activation values. The learning algorithm takes an initial FCM and iteratively modifies the model until the desired map is found. Learning terminates when the activation values of a selected subset of concepts achieve corresponding predefined levels.

The same research group proposed *Active Hebbian Learning* algorithm *(AHL)* in 2004 (Papageorgiou et al., 2004). This approach introduces and exploits the task of determining the sequence of activation concepts. Expert(s) determine(s) the desired set of concepts, initial structure and the 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 to satisfy the predefined stopping criteria.

In recent work, Stach and coworkers proposed an improved version of the NHL method (Stach et al., 2008b). The algorithm, called *Data-Driven Nonlinear Hebbian Learning (DD-NHL),* is based on the same learning principle as the NHL, but it takes advantage of all 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 using the generic NHL method (Stach et al., 2008b).

In 2001, Koulouriotis and colleagues applied the Genetic Strategy (GS) to learn FCM's model structure, i.e., weights of relationships, from data (Koulouriotis et al., 2001b). In their method, the learning process is based on a collection of input/output pairs, which are 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 an FCM that is able to generate state vector sequences that transform the input vectors into the output vectors. The main drawback to 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 and coworkers, belongs to the class of Swarm Intelligence algorithms (Parsopoulos et al., 2003). This method aims at learning FCM based on historical data that converge to a desired final state. 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 that is used to specify adequate constraints, which would guarantee that the relationships within the FCM model retain the physical meaning defined by the expert(s).

Another method based on swarm optimization, was proposed by Petalas and colleagues in 2009 (Petalas et al., 2009). This method, called memetic particle swarm optimization (MPSO), combines swarm intelligence memetic algorithm with both deterministic and stochastic local search schemes, for fuzzy cognitive maps learning tasks. The method needs an initial set of weights, which are usually imposed by an expert or group of experts, to perform optimization.

The next algorithm proposed by Khan and Chong aims to accomplish a different learning objective (Khan and Chong, 2003). 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 to find the initial state.

Table 2.2 summarizes the above-mentioned learning approaches. The comparison is made based on several factors, such as the learning goal, involvement of a domain expert, input data, and learning type.

Table 2.2:Summary of learning methods for FCMs

Algorithm	Learning goal	Human intervention	Type of data used [1)]	Transformation function	Learning type
AHL	Connection matrix	Yes&No[2)]	Single	Continuous	Modified Hebbian
BDA	Connection matrix	No	Single	Binary	Modified Hebbian
DD-NHL	Connection matrix	No	Single	Continuous	Modified Hebbian
DHL	Connection matrix	No	Single	N/A	Hebbian
GA	Initial vector	N/A	N/A	Continuous	Genetic
GS	Connection matrix	No	Multiple	Continuous	Genetic
MPSO	Connection matrix	Yes&No[2)]	Single	Continuous	Swarm
NHL	Connection matrix	Yes&No[2)]	Single	Continuous	Modified Hebbian
PSO	Connection matrix	No	Multiple	Continuous	Swarm

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 there is no human intervention needed

In summary, research on learning FCM models from data has resulted in a number of interesting approaches. One group of methods, which includes *NHL*, *AHL*, and *MPSO*, is aimed at providing tools that would support the expert(s) in the development of accurate models based on his or her knowledge about the modeled system. The remaining algorithms from the other groups are oriented toward eliminating human intervention from the entire development process, i.e., they only need historical data to establish FCM models for a given system. Early works applied mostly Hebbian learning, whereas recently genetic-based optimization techniques gain the momentum.

2.2.2 Aggregation methods for FCMs

Fuzzy Cognitive Maps allow for a relatively simple aggregation of knowledge obtained from several experts. The aggregation of FCMs aims at improving the reliability of the final model which is less susceptible to potentially erroneous beliefs of a single expert. There are a couple of procedures for combining multiple FCMs into a single, final model. They involve simple matrix operations, such as summations and multiplications by a number (Kosko, 1988), which are computed using individual connection matrices developed by different experts. It is not uncommon that experts decide on a different number of concepts. Consequently, the sizes of the corresponding matrices may not be the same and/or the corresponding rows/columns may refer to different concepts. In such a case, the first step towards the aggregation of maps is to equalize their sizes. Each connection matrix is augmented, if necessary, by including any missing concept(s), when compared to any other map, through the addition of extra rows and columns in the connection matrix filled with zeros. In other words, "dummy" concepts are added to the model. If the total number of distinct concepts for all input FCMs equals N, then each individual connection matrix is augmented to the matrix of $N \times N$ size (Khan and Quaddus 2004). Consequently, all individual maps have the same dimensions.

Chronologically, the first approach to aggregate FCMs was introduced by Kosko (Kosko, 1988). It simply takes an average of each relationship across individual maps. Therefore, the connection matrix of the final FCM is established through simple matrix operations:

$$\mathbf{E} = \frac{1}{K}\left(\mathbf{E}_1 + \mathbf{E}_2 + \ldots + \mathbf{E}_K\right) \tag{2.6}$$

where **E** is the connection matrix of the aggregated model, $\mathbf{E}_k$ is the connection matrix of the model developed by k^{th} expert, and K is the number of experts.

In other words, in this approach, each expert contributes equally to the final model. This method does not require any additional information except the knowledge of the input connection matrices. On the other hand, it gives each expert the same credibility, i.e., uses a simple, non-weighted average of the input matrices, which may be undesirable. The assumption for the formula above is that the individual matrices have the same number of dimensions and the same concepts. If this is not the case, a pre-processing of matrix augmentation as described in the previous paragraph, is necessary.

The basic method has been extended in order to accommodate for the credibility factors of individual maps (Kosko, 1988), by replacing the ordinary average with a weighted average. The weights w_i are assigned to experts proportionally to their reliability and take values from range [0,1]. Formula 2.7 is used to calculate the aggregated map using this method:

$$\mathbf{E} = \frac{1}{\sum_{k=1}^{K} w_k} \left(w_1 \mathbf{E}_1 + w_2 \mathbf{E}_2 + ... + w_K \mathbf{E}_K \right) \tag{2.7}$$

where **E** is the connection matrix of the aggregated model, $\mathbf{E}_k$ is the connection matrix of the model developed by k^{th} expert, w_k is the credibility of k^{th} expert, and K is the number of experts.

Hence, experts with higher credibility have a higher influence on the structure of the aggregated map than those with lower credibility. The applicability of this method is limited by difficulties in estimating the credibility coefficients (Stach et al., 2005a). Similar to the previous method, a pre-processing step of unifying individual matrices may be required.

The most recent method (Lin, 2007; Lin, 2008), for aggregating Fuzzy Cognitive Maps utilizes Dempster-Shafer (Dempster, 1967; Shafer, 1976) evidence theory, which is a tool for multi-expert opinions combination. It requires defining a frame of discernment that is used to convert the weights defined by experts to basic probability assignments. The

frame of discernment is a set of fuzzy sets that correspond to linguistic descriptors, which are later mapped to a numerical value. Each weight defined by an expert is changed to basic probability assignments based on the membership functions of the linguistic variables. Next, these probability assignment functions are combined according to the combination rule of evidence theory, and the target type is determined by the rule of belief evaluation. Finally, a weighted average of all the elements of the frame, according to the integrative basic probability assignment, is applied. The main difficulty in applying this method is its requirement for predefined frames of discernment, which includes both the number of the fuzzy sets and their membership functions. Although this choice affects all subsequent steps of the algorithm, the original paper does not give any guidelines on how they should be defined.

Table 2.3 summarizes methods for aggregating Fuzzy Cognitive Maps.

Table 2.3: Summary of aggregating methods for FCMs

Method	Type	Additional information needed
Average	Structural	No
Weighted average	Structural	Yes, credibility weights
Dempster-Shafer	Structural	Yes, frame of discernment

All of them operate on a structural level, i.e., they solely use the connection matrices of individual FCM models and aggregate them using certain formulas. The only method that does not require any additional information is *average*. Both other approaches require supplementary knowledge, which may be difficult or impossible to obtain.

Chapter 3

Learning Fuzzy Cognitive Maps from Data

3.1 Introduction

Designing a reliable and accurate Fuzzy Cognitive Map model for a given system is a challenging task (Aguilar, 2005; Stach et al., 2005a). Disadvantages related to expert-based development methods, such as bias and limited complexity of the model, encouraged researchers to work towards a systematic approach for FCM development (Stach et al., 2005a). In this realm, a number of computational methods that utilize available data to develop an accurate model for a given system have been recently proposed.

Computational methods for FCM development can be divided into two groups: *semi-automated* and *fully-automated* (Stach et al., 2005a; Stach et al., 2010). Methods from the former group require limited human expertise in the area of application, which typically involves sketching an initial map or defining a subset of relationships. Therefore, these methods are used to *support* the development process rather than to replace human experts. On the other hand, fully-automated methods *learn* the model from available data using a learning algorithm. Hence, they are intended to *replace* human experts.

Since FCMs include feedback loops and nontrivial transformation functions, forming the models from data is a complex task (Aguilar, 2005; Stach et al., 2005a). A differential Hebbian learning algorithm was the first one used to tackle this problem, see Table 2.2. It served as a cornerstone for a number of algorithms based on a modification of the Hebbian learning rule. Siminar research, within this framework, includes approaches that

use evolutionary algorithms, such as genetic algorithms or particle swarm optimization. Recently, these methods have gained momentum.

3.2 Detailed goals

In this chapter, a new fully-automated method for learning FCM from data is introduced. The method is a natural continuation of research conducted in the domain of FCM learning (refer to Table 2.2). It draws conclusions from the methods proposed in the past, and provides substantial advancements. The research goals are stated below:

- to introduce a new, fully-automated, learning method for Fuzzy Cognitive Maps with a continuous transformation function that is capable of learning high quality models using a single input state vector sequence (input data);
- to compare several designs of the proposed learning method and select the most effective solution;
- to carry out well-organized, thorough tests, considering the large number of diverse types of FCMs, and come up with firm design guidelines;
- to examine the influence of the input data length on the learning quality;
- to evaluate the method against other automated approaches to learn FCMs;
- to validate the proposed method on real data in application to time series prediction and to compare its performance to other fuzzy-based predictors;
- to investigate and propose improvements to the scalability of the method with the aim of applying it to a large system consisting of a few dozen of concepts;
- to propose a modification to the proposed learning method, which utilizes a priori knowledge of the model structure, that leads to improved quality of the learned FCMs.

3.3 Proposed approach

3.3.1 Problem statement

The objective of the FCM learning is to establish an FCM connection matrix **E**, see FCM definition in Section 2.1.2, given its set of concepts $C = \{C_1, C_2, ..., C_N\}$ and a sequence of their activation degrees $\left(\hat{\mathbf{C}}(0), \hat{\mathbf{C}}(1), ..., \hat{\mathbf{C}}(T-1)\right)$. The sequence forms a *TxN input data matrix* where *T* is the *input data length* and *N* is the number of concepts. We also assume that the transformation function is given and we carry out the learning task with the most commonly used in practice transformation function (Stach et al., 2005a; Tsadiras, 2008), which is the logistic function, see Formula 2.4, using parameter *M*=5.

3.3.2 Methods used

Learning of FCMs is equivalent to the task of optimization of the underlying connection matrix. The optimization's objective is to determine *NxN* parameters that define an FCM, such that a certain performance index, which is defined based on difference between input data and the simulation of the FCM, is minimized. An extension to the genetic algorithms has been chosen to perform the optimization task. Genetic algorithms (GAs) (Holland, 1975; Goldberg, 1989) are robust, global optimization methods, which originate from the biological realm, i.e., they maintain a population of *candidate solutions* (*chromosomes*) to a problem that evolves over a sequence of generations. They are based on the principle of "survival of the fittest", and therefore the better solutions have a higher probability to be selected for reproduction than poorer solutions. Compared to gradient descent techniques, GAs are superior because the search is not biased towards locally optimal solutions (Patnaik and Mandavilli, 1996). GAs usually start with a randomly generated population of candidate solutions. The *fitness function*, which is problem specific, is used to evaluate each chromosome. Chromosomes with higher fitness values have greater probabilities of being selected for further reproduction. Genetically inspired operators such as *crossover*, *mutation*, and *selection* are applied in each generation to produce "better" chromosomes and prevent the premature convergence of the GAs to suboptimal solutions.

Generic genetic algorithms are better suited to tackle optimization problems with discrete variables due to the chromosomes' binary representation. One of the extensions proposed to GAs, called *real-coded genetic algorithms*, eliminates this inconvenience. Real-coded genetic algorithms (RCGA) (Herrera et al., 1998) are a floating-point extension to GAs, which makes them convenient in their applications to optimization problems with continuous variables. The key difference between these two approaches lays in chromosome representation. Contrary to GAs that use binary strings, floating-point vectors are used in RCGA. When compared to GAs, the genetic operators were revised to handle floating-point values. Nevertheless, the principles of these two optimization techniques are the same. Since our optimization problem involves floating-point parameters, the RCGA method was selected and used in the proposed learning method.

3.3.3 RCGA learning algorithm

Chromosome structure

Chromosome structure needs to be defined in a way that each chromosome represents a solution to a given problem. As a result, for the FCM learning task, the structure must include $N*N$ floating-point values where N is the number of concepts in the modeled system.

$$\hat{\mathbf{E}} = \left[e_{11}, e_{12}, ..., e_{1N}, e_{21}, ..e_{2N}, ..., e_{NN}\right]^T \tag{3.1}$$

where e_{ij} corresponds to the relationship strength from the concept C_i to the concept C_j and is normalized on the [-1,1] interval.

The chromosome's structure can be easily converted from a $1xN^2$ vector to a NxN matrix, which represents a connection matrix of a solution to the learning task. The solution is called *candidate FCM*.

Fitness function

Fitness function definition is considered as one of the most important factors in the successful application of genetic algorithms (Goldberg, 1989; Herrera et al., 1998). The design of the fitness function in the proposed learning method takes advantage of a

specific feature of the FCMs theory. Specifically, we note that at each iteration of model simulation, the state vector ***C(t+1)*** depends only on the state vector at the preceding iteration and does not depend on any other state vectors, see Formula 2.1. We use this property to group each two adjacent state vectors in the input data matrix to form *T-1 input data pairs*

$$\left(\hat{\mathbf{C}}(t-1),\hat{\mathbf{C}}(t)\right) \qquad \forall t = 1,...,T-1 \tag{3.2}$$

where $\hat{\mathbf{C}}(t)$ is the t^{th} state vector in the input data matrix.

Within each pair we call the antecedent $\hat{\mathbf{C}}(t-1)$ a *system stimulus*, whereas the decedent $\hat{\mathbf{C}}(t)$ 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.

Generally speaking, the fitness function for each chromosome is calculated by performing one step simulations of the candidate FCM starting from each system stimulus. The state vectors obtained from simulations, *candidate FCM responses*, are compared against the corresponding *system responses*. The candidate's FCM error function used to calculate fitness value is shown below:

$$error\_L_p = \alpha \sum_{t=1}^{T-1} \sum_{n=1}^{N} \left| C_n(t) - \hat{C}_n(t) \right|^p \tag{3.3}$$

where $\mathbf{C}(t) = \left[C_1(t), C_2(t), ..., C_N(t)\right]$ is the candidate FCM response for $\hat{\mathbf{C}}(t-1)$ system stimulus, $\hat{\mathbf{C}}(t) = \left[\hat{C}_1(t), \hat{C}_2(t), ..., \hat{C}_N(t)\right]$ is the system response for $\hat{\mathbf{C}}(t-1)$ system stimulus, p is the error norm type, and α is the normalization parameter.

We performed an experimental comparison (see Section 3.3.4) of the performance of our learning method for three different p-norms to calculate the error function, i.e., p=1 (taxicab norm), p=2 (Euclidean norm), and $p=\infty$ (maximum norm). The parameter α is

used to normalize the error rate and it is equal to $\frac{1}{N(T-1)}$ for the taxicab and Euclidean norm, and $\frac{1}{(T-1)}$ for the maximum norm.

The fitness function in the proposed learning approach is defined as follows:

$$fitness = h(Error\_L_p) \tag{3.4}$$

where h is an auxiliary function, and $h(x) = \frac{1}{ax+1}$, a is a parameter established experimentally.

The auxiliary function h was introduced for the following reasons:

- to ensure that better individuals have a higher fitness value and to normalize the fitness value onto the (0,1] interval;
- to embed non-linearity that rewards chromosomes that are close to the optimal solution.

Genetic operators

The following genetic operators need to be defined for the RCGA method: crossover, mutation, and selection (Herrera et al., 1998). We performed an experimental evaluation, described in the following subsection, of different configurations, which is based on the results we chose the recommended setup.

Stopping conditions

The following stopping condition has been implemented:

- fitness function value of the best chromosome is greater than or equal to its certain maximum value

OR

- maximum number of generations is reached

These maximum values have been established during experiments described in the following subsection.

Detailed procedure for the proposed RCGA learning method

The above-mentioned discussion is summarized below where all steps of the proposed RCGA learning algorithm for Fuzzy Cognitive Maps (Stach et. al., 2005c) are presented.

Given

A set of concepts $C = \{C_1, C_2, ..., C_N\}$ and a sequence of their activation degrees $\left(\hat{\mathbf{C}}(0), \hat{\mathbf{C}}(1), ..., \hat{\mathbf{C}}(T-1)\right)$

Step 1: Divide data into *T-1* pairs: $\left(\hat{\mathbf{C}}(t-1), \hat{\mathbf{C}}(t)\right) \qquad \forall t = 1, ..., T-1$

Step 2: Randomly initialize a population of *p* chromosomes, such that each chromosome is represented as $\hat{\mathbf{E}} = \left[e_{11}, e_{12}, ..., e_{1N}, e_{21}, ..e_{2N}, ..., e_{NN}\right]^T$ where e_{ij} is a random number on the [-1, 1] interval

Step 3: Calculate the fitness function for entire population, which is carried out by decoding the candidate FCM from each chromosome, and applying Formula 3.4

Step 4: Check the stopping condition. If it is satisfied, return the chromosome with the highest fitness function value

Step 5: Apply crossover operation on the population

Step 6: Apply mutation operation on the population

Step 7: Calculate the fitness function values for the entire population

Step 8: Check stopping condition

Step 9: Generate a new population using selection operations

Step 10: Go to *Step 5*

3.3.4 RCGA parameterization experiments

Experimental setup

We used both synthetic and real-life data to perform experiments with parameterization of the proposed method. The former group of experiments uses synthetic *input data*, which are generated from random *input FCM models* and random *initial state vectors*, whereas the latter uses real-life, i.e., previously reported in a scientific literature, FCM models to generate input data. The *synthetic* setup allows for performing large number of experiments with fully customizable parameters, such as: number of concepts or maps structure. The *real-life* setup allows for the verification of the quality of the solutions against the applications of FCMs in a real world.

Evaluation criteria

The RCGA learning method performs optimization based on input data. The quality of learning with respect to the input (training) data is measured as an average difference between the input data and the data from the simulation of the learned FCM starting from the same initial state vector:

$$in-sample = \frac{1}{(T-1)N}\sum\sum\left|C_n(t)-\hat{C}(t)\right| \tag{3.5}$$

where $C_n(t)$ is the activation value of concept C_n at iteration t obtained from simulating the candidate FCM, $\hat{C}_n(t)$ is the activation value of concept C_n at iteration t obtained from the input data, and T is the input data length

To avoid overfitting ,we complete simulations that start from different initial state vectors when compared with the training simulations. These experiments were repeated ten times using different initial vectors. Hence, the evaluation is carried out on previously unseen data, and thus it measures generalization capabilities of the candidate FCM. The formula to calculate this measure, *out-of-sample error*, is shown below:

$$out-of-sample = \frac{1}{P(T-1)N}\sum_{p=1}^{P}\sum_{t=1}^{T-1}\sum_{n=1}^{N}\left|C_n^p(t) - \hat{C}_n^p(t)\right| \quad (3.6)$$

where $C_n^p(t)$ is the activation value of concept C_n at iteration t obtained from simulating the candidate FCM started from p^{th} initial condition, $\hat{C}_n^p(t)$ is the activation value of concept C_n at iteration t obtained from simulating the input model started from p^{th} initial condition, T is the input data length, P is the number of different initial state vectors, and N is the number of concepts.

Results

The first group of experiments involved tuning the RCGA parameters. Randomly generated input FCMs with a different number of concepts (5, 10, 15) and densities (20% and 40%) were used to generate the input data. The goal was to find RCGA parameters that most consistently led to the best solutions using different setups and in a reasonable running time. In total, fifty experimental learning results were visually inspected to establish the learning parameters that 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: 0.5;
- probability of mutation: 0.5;
- population size: 100 chromosomes;
- maximum number of generations: 30,000;
- maximum value of fitness function: 0.999;

A relatively high value of mutation probability is motivated by a large number of sub-optimal solutions. A low mutation value leads to slow exploration of the search space,

and, consequently, the algorithm often gets stuck in sub-optimal solutions. We noted that in such cases the FCM performance on unseen data deteriorates significantly when compared to its performance on data used for learning.

The second group of experiments focused on comparing different fitness functions, see Section 3.3.3. Twenty experiments using each fitness function type were carried out to tune up the parameter *a*. Next, experiments given the same setup, i.e., number of concepts and densities as in the first group, were carried out. Each experiment was repeated 10 times and average out-of-sample error values are reported in Table 3.1.

Table 3.1: Fitness function parameterization results for RCGA learning method

Setup		Out-of-sample error		
Number of concepts	Input map density	Fitness function L1	Fitness function L2	Fitness function L∞
5	20%	0.019	0.014	0.015
	40%	0.018	0.012	0.011
10	20%	0.143	0.130	0.140
	40%	0.141	0.132	0.139
15	20%	0.147	0.141	0.153
	40%	0.148	0.140	0.151

The results show that the fitness function based on *L2* norm outperforms others. Additional comparisons between fitness functions are reported in our previous work (Stach et al., 2005c). We note that when the *L1* norm is used, the convergence to the final solution is slower, whereas $L\infty$ norm provides reasonably good results for smaller maps but is ineffective when the search space increases.

3.3.5 Evaluation

We used both synthetic and real-world data. In the first scenario, the data for each experiment were obtained by simulating randomly generated input FCM models starting from a random initial vector. We grouped our experiments based on FCM's size that includes 4, 5, 10, 15, and 20 concepts. For each group we used input maps with density 20% and 40%. In addition, an e-business company FCM model (Tsadiras, 2003) was used to evaluate the proposed method. Table 3.2 shows experimental results.

Table 3.2: Experimental results with RCGA method. Last three columns report out-of-sample error value for the corresponding learning approach followed by the standard deviations

Setup		Learning Method		
Number of concepts	Input map density	RCGA	NHL	DD-NHL
4	20%	0.001 (±0.001)	0.212 (±0.200)	0.211 (±0.201)
	40%	0.000 (±0.001)	0.213 (±0.201)	0.206 (±0.203)
5	20%	0.011 (±0.007)	0.201 (±0.188)	0.202 (±0.192)
	40%	0.012 (±0.009)	0.205 (±0.198)	0.204 (±0.197)
7	40%	0.092 (±0.063)	0.221 (±0.195)	0.215 (±0.199)
10	20%	0.130 (±0.125)	0.218 (±0.211)	0.211 (±0.205)
	40%	0.132 (±0.128)	0.212 (±0.201)	0.212 (±0.197)
15	20%	0.141 (±0.119)	0.211 (±0.198)	0.207 (±0.187)
	40%	0.140 (±0.132)	0.215 (±0.197)	0.202 (±0.187)
20	20%	0.152 (±0.135)	0.219 (±0.204)	0.211 (±0.199)
	40%	0.155 (±0.144)	0.214 (±0.211)	0.209 (±0.205)

In order to put the out-of-sample error values into perspective, a baseline which corresponds to the out-of-sample error value of a random map was calculated, and it equals 0.37. We observe that the learning quality measured by the out-of-sample error decreases as the number of the concepts increases. The errors produced by maps developed using the RCGA-based approach are significantly smaller than the errors of the baseline and the two existing learning approaches used for comparisons, i.e., NHL and DD-NHL. We verified the statistical significance of the differences using paired t-test and it revealed that the RCGA learning outperformed both NHL and DD-NHL with confidence level at 98%. Results included in Table 3.2 also demonstrates that the learning quality does not depend on the density of the input model.

Figures 3.1 and 3.2 show sample plots obtained from experiments performed for a map with 10 concepts and 40% density. These figures correspond to the best and worst case of experiments with the RCGA method, i.e., experiments with the lowest (Figure 3.1) and highest (Figure 3.2) out-of-sample error value, respectively.

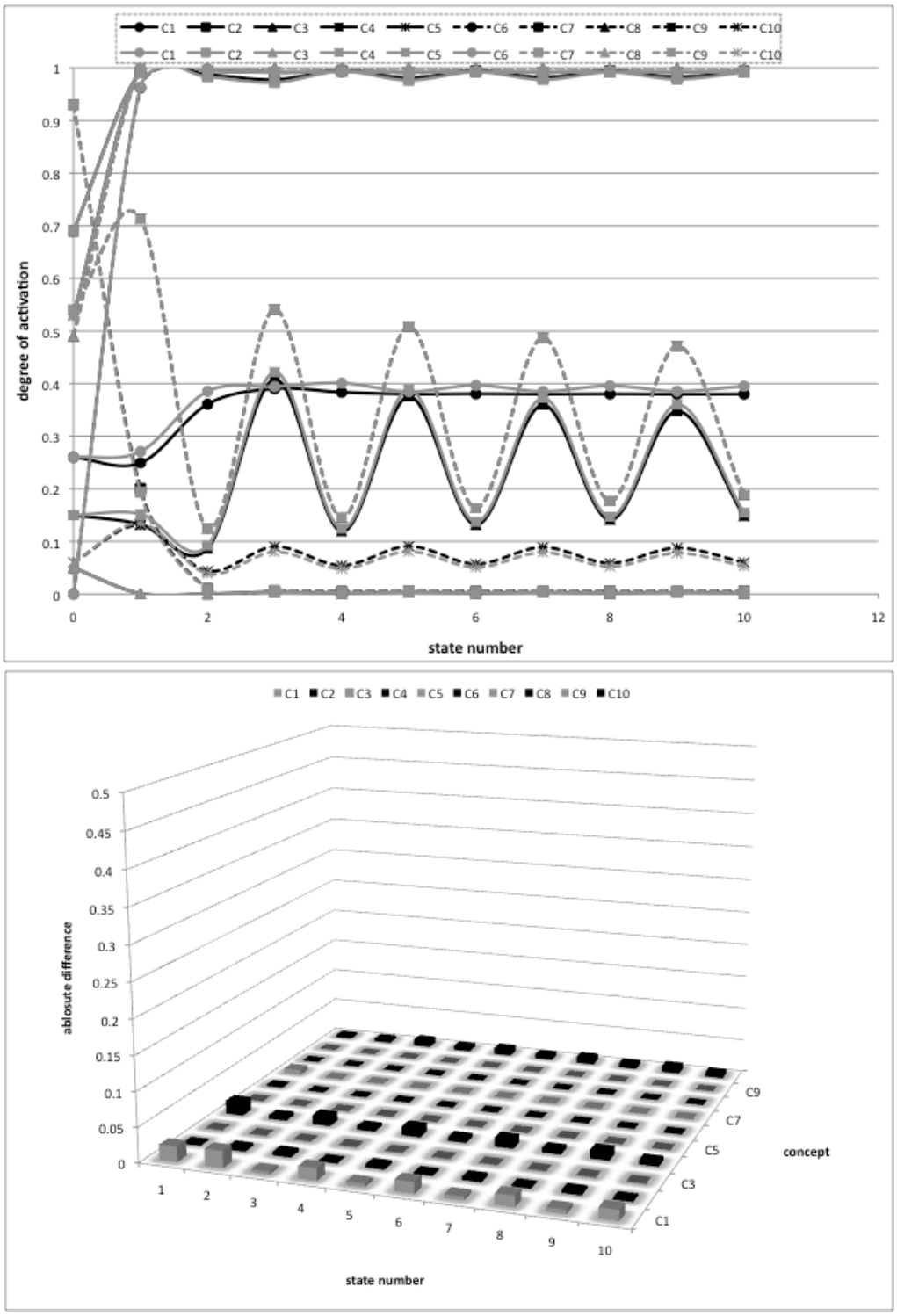


Figure 3.1: Experimental results with the RCGA method: the "best" experiment. The upper plot shows simulation result of both learned (black) and input (grey) FCMs. The bottom plot shows differences between activation degrees of corresponding concepts over successive iterations.

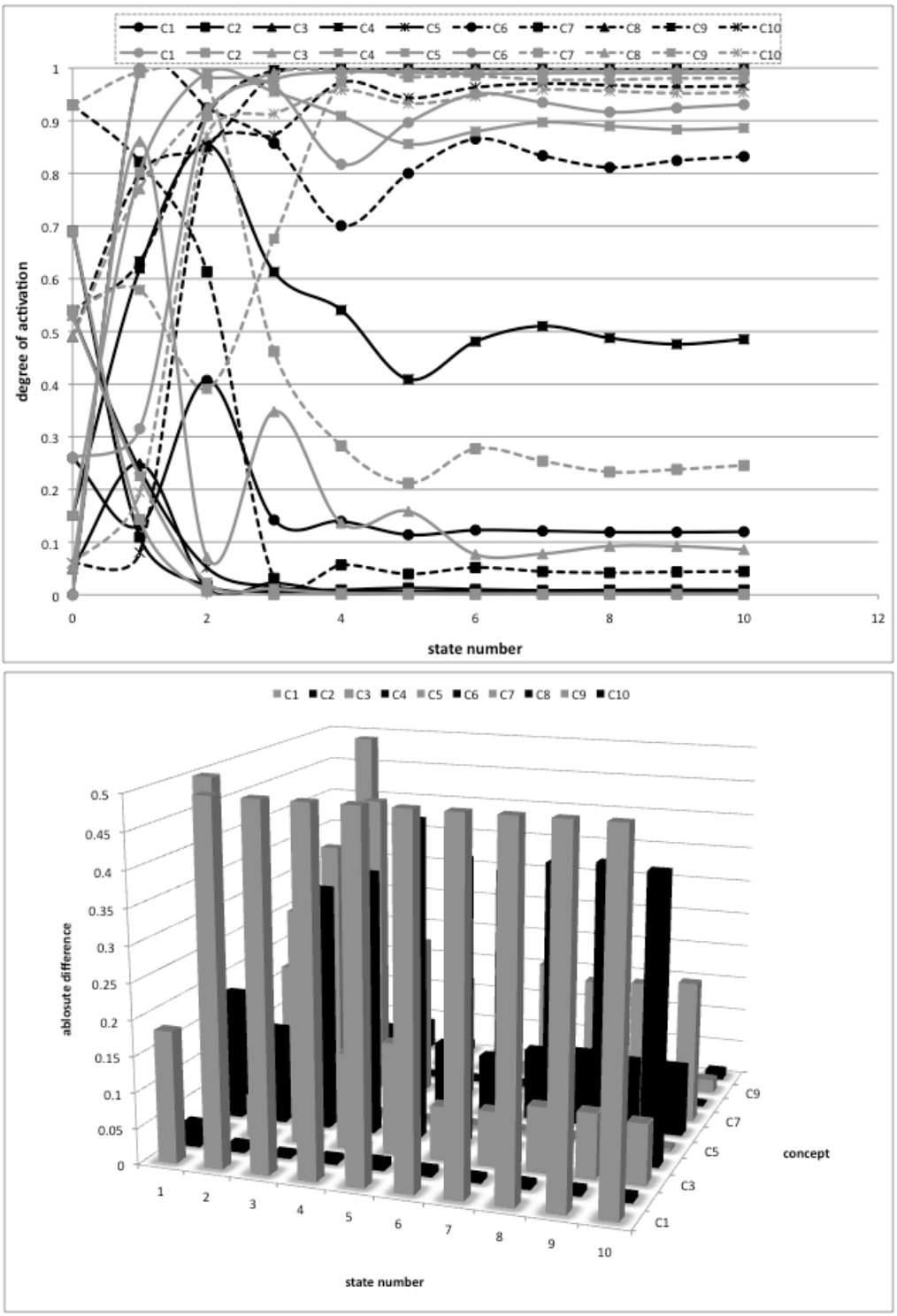


Figure 3.2: Experimental results with the RCGA method: the "worst" experiment. The upper plot shows simulation result of both learned (black) and input (grey) FCMs. The bottom plot shows differences between activation degrees of corresponding concepts over successive iterations.

The results from Figure 3.1 show that the proposed method can generate almost perfect solutions for some systems. Although the true plot and the simulation of the learned FCM which are shown in Figure 3.2 differ, we note that for more than half of concepts the differences between the activation levels of the corresponding concepts are negligible. Additional experimental results are presented in Section 3.5.5. They include a more complete comparison of the performance of the RCGA method against other FCM learning methods, as well as against other approaches proposed in the following subsections of this dissertation.

An analysis of the influence of input data length on the RCGA learning quality was presented in our previous work (Stach et al., 2004b). The conclusion drawn from this study is that increasing the size of input data length improves the accuracy of learning, whereas if the input data length is insufficient, the quality of learning deteriorates significantly. This is due to the existence of multiple different models that can be generated from input data of small size. Experiments performed with real-life models show that for small maps that consist of five concepts the input data length should be more than 10 iterations, whereas for the maps consisting of ten concepts more than 20 data points are required to allow for quality RCGA learning.

3.4 Application of RCGA learning to time series prediction

3.4.1 Outline

An evaluation of FCMs learned with the proposed RCGA method against real data is performed based on an application to time series predictions (Stach et al., 2008a). In our setup, Fuzzy Cognitive Maps along with the learning method are used to provide: 1) a description of a given time series at a certain abstraction level, and 2) numerical and linguistic predictions. Experimental results are compared with other state-of-the-art prediction methods based on fuzzy sets.

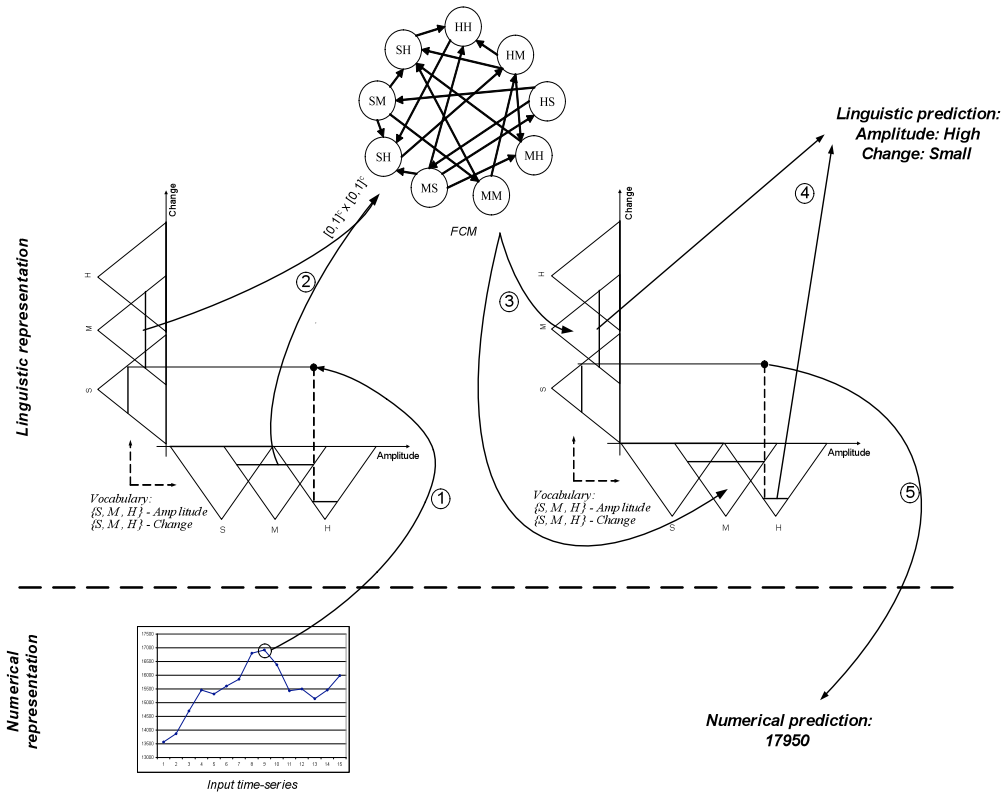


Figure 3.3: Overview of the proposed FCM and RCGA-based prediction system.

Figure 3.3 illustrates the design of the proposed system. The proposed architecture consists of three well-delineated and functionally distinct modules that include (a) an input interface, (b) a processing core formed by an FCM, and (c) the output interface. The modeling and prediction activities supported by the FCM are realized at the linguistic level as opposed to the numeric one at which the experimental data become available. Therefore, in contrast to classical time series prediction systems, that predict only numerical values, the proposed system can also perform predictions at the linguistic level.

The dynamics of a given numeric time series is captured through its amplitude and change in amplitude, say ($x(k)$, $\Delta x(k)$). These values are transformed through a collection of predefined linguistic descriptors, see Figure 3.3 step 1, and become available in the form of their activation levels. The encoding (fuzzification) process entails the determination of the membership values of the respective fuzzy sets. The computations at this stage are straightforward as the values of the membership functions for the current numeric value of the time series, $x(k)$ and its difference, $\Delta x(k)$ are taken. Next, the result of the encoding is processed by the FCM, see Figure 3.3 step 2. Each Cartesian product

of the linguistic terms in the space of amplitude, and its changes, corresponds to a certain node of the map. Here we can consider two alternatives. The results obtained by the FCM can be presented at two levels:

- linguistic level, see Figure 3.3 step 4. Here the node of the FCM with the highest degree of activation is selected. The result of the prediction comes in the format *(Amplitude is A* × *change of Amplitude is B) is* μ where *A, B* are the labels (fuzzy sets) forming the node of the FCM while μ is the level of activation of this node.

- numeric level, see Figure 3.3 step 5. Here we consider all nodes of the FCM along with their activation levels and return a single numeric value by carrying out decoding (defuzzification).

3.4.2 Proposed prediction system

The heart of the prediction system is an FCM along with the RCGA learning algorithm. The RCGA method is used to establish a model of the given time series signal, which is then used to predict its future values. Figure 3.4 shows a high-level architecture of the proposed prediction system.

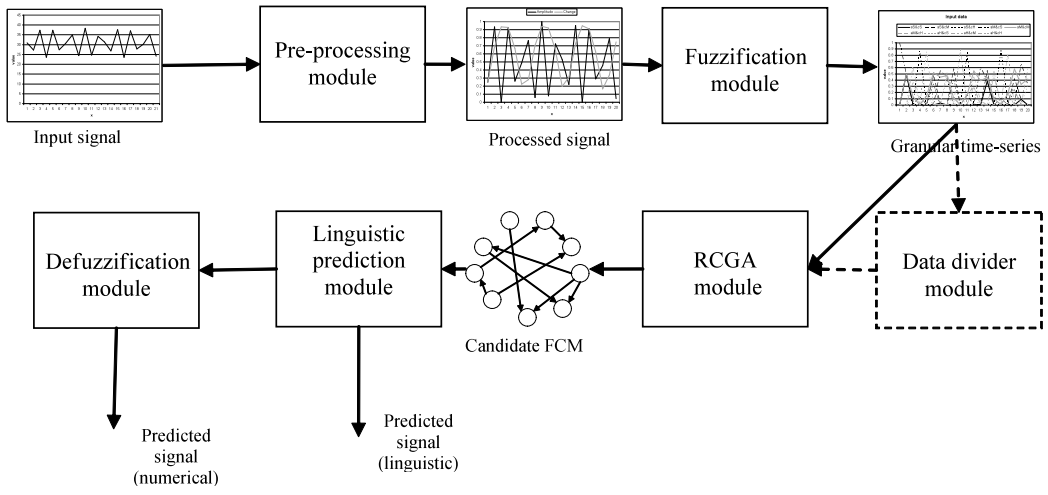


Figure 3.4: High-level diagram of the proposed prediction method.

The FCM prediction system realizes a series of well-delineated steps as shown in Figure 3.4. The input signal is pre-processed in a pre-processing module, which has a dual purpose. Firstly, it extracts feature(s) of interest for the linguistic prediction. They include

signal *change*, which is defined as the difference between two consecutive values of a given input signal, and the signal's *amplitude*. The change constitutes an additional time series. Secondly, both signals are normalized linearly to the unit interval. In order to avoid artificial enlargements of small signal changes, the normalization of the change signal is carried out based on the range of the original time series signal. More specifically, the maximum possible change value is determined and the normalization is performed with respect to this value. As a result of the pre-processing module, two normalized signals, i.e., amplitude and change, are obtained. The first value of the amplitude signal is dismissed to have an equal length of both signals.

After pre-processing, information granules of the signal determining its current status are extracted and aggregated in the fuzzification module. This process involves linguistic descriptors (labels), which are given as a set of fuzzy sets. Based on their definitions, membership values are calculated for each value of both signals. The linguistic descriptors can be defined uniformly or independently for each signal. Let us consider K time series as an input to this module and a number of corresponding linguistic descriptors denoted by the $N_1, N_2, \ldots, N_K$. In the first phase, these signals are represented in terms of membership values of given fuzzy sets, which results in having a $N_1+N_2+\ldots+N_K$ fuzzy time series. Next, the granularization process takes place, which links the fuzzy time series with the use of fuzzy operators. As a result, representation of each data point (observation), which is a unit hypercube $[0,1]^K$ at the entry to this module, extends to $[0,1]^{N_1*N_2*\ldots*N_K}$. Therefore, the total number of granular time series that form the output from this module is $N_1*N_2*\ldots*N_K$. Each of these time series expresses the level at which the given signal can be characterized by corresponding linguistic descriptors. We provide unique linguistic labels over the entire time series by choosing the descriptors with the highest values at each time point.

The next, data divider, module is motivated by the organization of our experimental setup and thus it does not belong to the proposed prediction method per se. In particular, it serves to provide an experimental evaluation of the prediction method dividing the input dataset into training and test subsets. The former subset is used to develop appropriate FCMs, whereas the latter is separate and is used to test prediction accuracy on unseen data.

The actual learning of FCM is performed in the RCGA module, which establishes FCMs based on training data, see Section 3.3.3. The number of concepts of the candidate FCM corresponds to number of granular time-series from the output of the fuzzification module. The concepts depict a complete signal description within the assumed fuzzy domain, i.e., each node corresponds to a single combination of linguistic descriptors of the granular time series.

The candidate FCM is used by the linguistic prediction module to carry out the signal prediction in a fuzzy domain (linguistic prediction) on the test data. This process involves a model simulation according to the scenario defined in the data divider module. Linguistic prediction uses fuzzy operations on a granular time series obtained from the simulation.

Numerical predictions require fuzzy values to be defuzzified. The defuzzification module performs this process according to a predefined defuzzification method on a granular time-series obtained from simulation and is carried out on the test data. The numerical prediction is performed based on the defuzzified values. In addition to the defuzzified signal value, other signal features defined in the pre-processing module may be also used as a supplement, or correction coefficient, during prediction.

3.4.3 Experimental results

This section summarizes the evaluation of the proposed prediction system against other existing predictors based on fuzzy sets, i.e., Song-Chissom (Song and Chissom, 1994), Chen (Chen, 1996), Markov (Sullivan and Woodall, 1994), and Hwang (Hwang et al., 1998) methods. In order to keep this section concise, we present only comparative results obtained from experiments on a single, most commonly used dataset. For other results, please refer to our recent paper (Stach et al., 2008a).

To achieve consistency with the original experimental setup (Song and Chissom, 1994) we used a data set that concerned enrolment at the University of Alabama during 1972-1992 and performed two types of experiments: one that used a single model for prediction (time-invariant), and another that used a moving window for prediction (time-variant). A numerical prediction error was measured to evaluate the quality of each method. Table 3.3 summarizes the experimental results.

Table 3.3: Comparative evaluation of the proposed prediction method on enrolment dataset

Method	Numerical prediction error [%]
Time-invariant experiments	
Song-Chissom	3.20
Chen's	3.22
Markov	2.60
FCM with RCGA (3 labels)	2.13
FCM with RCGA (4 labels)	2.58
Time-variant experiments (window = 4)	
Song-Chissom	4.37
Hwang	3.12
FCM with RCGA (3 labels)	3.23
FCM with RCGA (4 labels)	2.66

The results show that the proposed method gives better results for an enrolment dataset when compared to other methods. The proposed system using 3 linguistic labels (9 nodes FCM) achieved 2.13 of error value for the time-invariant test, while the second best Song-Chissom method scored 3.20. Similarly for the time-variant test, the proposed method was best and scored 2.66.

3.4.4 Conclusions

The proposed prediction system based on FCMs and RCGA learning outperforms other existing fuzzy sets-based predictors. Importantly, the FCM models that are used to perform time-series predictions are generated and validatated against real-life data, in contrast to the evaluation performed in Section 3.3.5 that uses expert-generated or synthetic FCM models as the reference point.

In addition, the method offers linguistic predictions. Comprehensive tests reported in our recent paper (Stach et al., 2008a) show that the linguistic accuracy of the proposed method decreases as the number of considered linguistic labels becomes higher, while at the same time, the numerical prediction accuracy increases. This shows that a trade-off between the quality of the numerical and linguistic prediction exists. By selecting a proper number of labels, users can control the quality and scope of the prediction in terms of granularity of the linguistic description. Both linguistic and numerical prediction accuracy are shown to improve with the increasing number of input data points that are used to develop the prediction model. The proposed method provides numerical

predictions with accuracy higher or comparable to other state-of-the-art prediction methods, which are based on fuzzy sets.

3.5 Scalable RCGA learning approaches to Fuzzy Cognitive Maps

3.5.1 Motivation

Although the RCGA learning method introduced in the previous paragraph is capable of learning high quality FCMs, the experiments showed that it struggles when applied to large systems (Stach et al., 2007). As a result, the method could be applied to learn models of up to two dozen or so concepts in about over an hour using a desktop PC, but the computational time grows exponentially with the number of concepts. Modeling larger systems, such as these from systems biology that include several dozens of concepts, can not be completed within a few hours. This is due to the poor scalability of genetic algorithms. The number of variables that need to be established during FCMs learning grows quadratically with the number of concepts. These issues call for a scalable approach that is capable of learning large maps of high quality within a reasonable time frame.

In this section we propose two improvements to the RCGA learning method that aim to improve its scalability. One of them includes the *parallelization* of genetic algorithms, whereas the other one uses a *divide and conquer* strategy.

3.5.2 Methods used

Parallel computing is one of the more popular techniques that speeds up the process of solving complex computational problems (Grama et al., 2003). It assumes simultaneous execution of the same task on multiple processors to obtain results more quickly. The underlying assumption is that a problem being solved can be divided into smaller tasks, which can be executed simultaneously with a certain level of coordination. In parallel computing, it is crucial to use parallel algorithms to take advantage of hardware systems. *Parallel algorithms*, in contrary to sequential algorithms, can be divided into parts performed in parallel (Xavier and Iyengar, 1998). Subsequently, the partial results are put back together to obtain the final result. The task of finding an efficient parallel algorithm

to solve a given problem can be very challenging as we often deal with sequential constrains. For instance, recursive solutions that require results from a previous iteration to calculate result in next iteration are very difficult to parallelize.

Divide and conquer (D&C) is an important algorithm design paradigm based on multi-branched recursion (Cormen et al., 2000). A divide and conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same (or related) type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem. The divide and conquer design is a powerful tool for solving conceptually difficult problems, such as the classic Tower of Hanoi puzzle. Divide and conquer algorithms are relatively easy to implement on multi-processor machines.

3.5.3 Parallel RCGA method for learning FCMs

As genetic algorithms (GAs) become increasingly popular, they are applied to difficult problems that require considerable computations. In such cases, parallel implementations of GAs become necessary to reach high-quality solutions in reasonable time frames. Generally speaking, these parallel approaches can be divided into methods that use a single population or a number of subpopulations. The following four main paradigms are utilized to parallelize GAs (Konfrst, 2004; Cantu-Paz, 2007):

- global single-population master slave: a single population is maintained but the evaluation of fitness is performed by multiple processors;
- single-population fine-grained: a single population is maintained, but the selection and crossover are restricted to a small neighbourhood. Nevertheless, some interaction among all the individuals is allowed;
- multiple-population coarse-grained: several subpopulations exist and they occasionally exchange individuals;
- hybrid models: some elements from the above three paradigms are combined.

Among the above four choices, the two most popular are the (a) global single-population master slave and (b) multiple-population coarse-grained (Konfrst, 2004). The former despite being very simple, has been shown to be very efficient (Cantu-Paz, 2007). The

implementation usually follows the master-slave framework, where the master stores the population and the slaves evaluate the fitness of chromosomes. This is performed by assigning a fraction of the population to each of the available processors, see Figure 3.5. The master stores the population, executes the GA operations, and distributes individuals to the slaves. The slaves then evaluate the fitness of the individuals. Communication between processes occurs only when slaves receive a subset of individuals to evaluate and when they return the fitness values to the master process. In this approach, the underlying computer architecture is not constrained by any special requirements. In addition, it does not affect the behaviour of the genetic algorithm since no additional restrictions are imposed on the genetic operators, such as crossover or selection. The second approach, which is based on multiple populations, uses a few relatively large subpopulations that exchange chromosomes, which is called migration, see Figure 3.6. Each subpopulation, denoted here by a circle, is executed as a standard GA and there is (infrequent) communication between populations. In this example, the populations are arranged in a ring, but a few other communication topologies have been introduced. (Konfrst, 2004; Cantu-Paz, 2007)

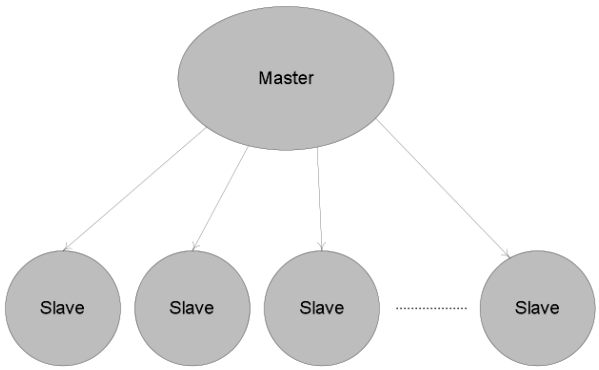


Figure 3.5: Master-slave parallel GA architecture. Arrows describe assignment of chromosomes to slave processes for fitness function calculation

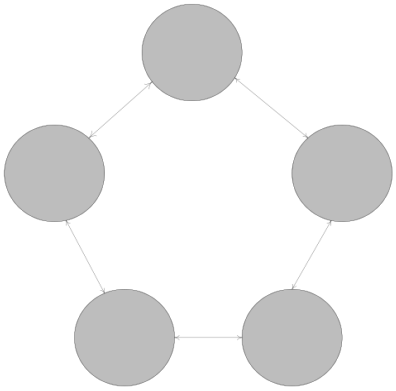


Figure 3.6: Multiple-population parallel GA topology. Nodes correspond to subpopulations, whereas arrows show how the subpopulations exchange chromosomes

3.5.4 Divide and conquer RCGA method for learning FCMs

In this section, we introduce a new scalable learning method for FCMs that divides the input data into subsets and performs simultaneous learning on each subset. The motivation for implementing this method is derived from the linear relation between the number of calculations required to evaluate the fitness function, which is the most time-consuming piece of the RCGA optimization, and the length of the input data. Therefore, a reduction in the length of the input data could provide a linear decrease in the execution time when using a multiprocessor architecture (assuming each sub-problem is solved on a different processor). This may potentially provide a better speed-up than the improvement observed when parallelizing the RCGA algorithm. Figure 3.7 presents a high level diagram of the proposed method. The two core modules are the Data Divider and FCM Fusion.

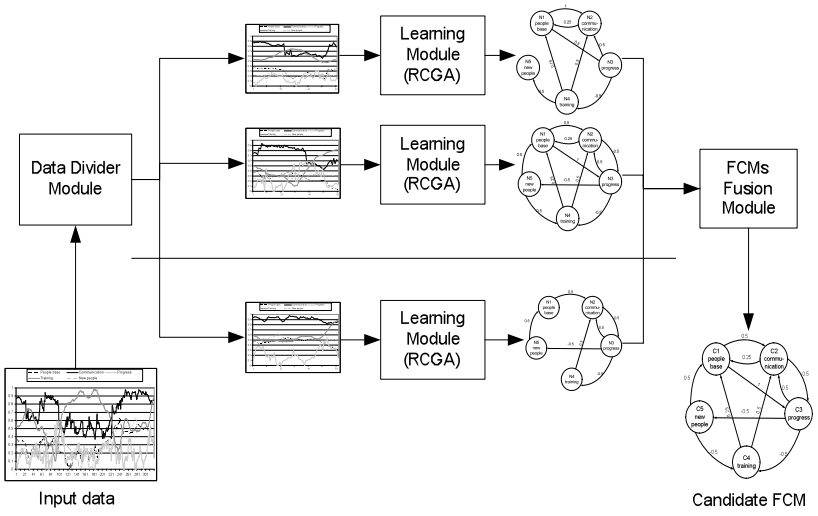


Figure 3.7: High level diagram of the proposed divide and conquer method to learn FCMs

A *Data Divider* takes the input data, given as input data matrix (equivalent to a discrete time-series), and splits it into subsets that are used to learn the submodels. Various strategies could be implemented by taking advantage of the inherent properties of FCMs. Firstly, the data may be divided into K non-overlapping contiguous subsets (intervals) where S is the number of available processors. Secondly, the split may be done as above except that the subsets would overlap to assure better similarity (and potentially better quality) between the submodels. Thirdly, as the RCGA method can use any two subsequent input data vectors to learn FCMs, the subsets can be obtained by random sampling of the input data pairs. Finally, random sampling with replacement could be used. The motivation for the latter case is the same as that of the second strategy. While these strategies are computationally equivalent (strategies 1 and 3, and, 2 and 4 would use the same number of input data pairs), the first two strategies may result in generating submodels that overfit to their corresponding input intervals. On the contrary, the latter two random sampling-based strategies include input data pairs that cover the entire input time-series. Their submodels are better generalized to describe the entire input data. This is the reason that focus on these two strategies. We empirically investigate the impact of the amount of overlap between the subsets (when allowing for the replacement) of the trade-off between the amount of computations and the quality of the learned FCM model. Each of the data subsets is used to generate a separate submodel. This process involves running, in parallel, K independent experiments using the RCGA learning method.

The *Fusion Module* aggregates the submodels. The subject of aggregating multiple FCMs into a single model is outlined in Section 2.2.2. The simplest, *average,* method aggregates FCMs averaging the corresponding relationship values (weights) across all the submodels. A modification of this method, *weighted average*, adds credibility (confidence) weights to each submodel. Consequently, weighted average across all submodels is calculated instead of simple average, see Formula 2.7. We decided to use an in-sample error of each submodel, see Formula 3.5, to quantify its quality. Next, we calculated the credibility weight of each model as (*1 – in-sample error*), and used Formula 2.7 to aggregate the submodels. The motivation to use an using in-sample error instead of splitting the input data for each submodel into the training and validation parts, was driven by the deteriorating learning quality of RCGA for insufficient input data length reported in our previous work (Stach et al., 2004b).

3.5.5 Evaluation criteria

The proposed methods have been evaluated based on the time needed to complete the learning and the learned FCMs quality. In particular, the following evaluation measures were applied:

- Execution time – measures the time needed to complete learning, which is expressed in seconds. This measure does not include the time to load the input data and to perform model evaluation.

- Out-of-sample error – measures the average difference between each concept value in the corresponding simulations generated by the candidate FCM and the input map. The value is calculated as an average over *P* experiments with different initial state vector, see Formula 3.6

3.5.6 Experimental setup

We used both synthetic and real-world data. In the first scenario, the data for each experiment were obtained by simulating randomly generated FCM models starting from a random initial vector. We grouped our experiments based on the FCM's size that includes 10, 20, and 40 concepts. Each test was executed in a sequential fashion, as well as in parallel using 2, 4, and 8 processors. The experiments were performed on an IBM p5 server. Additionally, we generated 5 independent input data for every setup where each

input data length was 40. The in-sample results were averaged over the 5 inputs and for each of the 4 FCM sizes, 4 parallelization architectures, and each learning method under consideration. For each setup, the out-of-sample data have been generated by simulating the models from 10 random initial vectors.

In the second scenario, two real-world maps reported in literature were used. We concentrated on relatively large maps selecting a model of factors affecting slurry rheology (13 concepts) (Banini and Bearman, 1998), and factors in the adoption of educational software in schools (24 concepts) (Hossain and Brooks, 2008). Similar to experiments using synthetic data, the out-of-sample error was computed by simulating the original and the learned models from 10 randomly chosen initial vectors.

The following methods have been used for comparisons:

- *Single population* – parallelized version of the RCGA learning method, in which the learning module has been implemented using global single-population master slave GAs. Each slave process executes on a separate processor and evaluates the fitness function of a given subset of individuals, see Section 3.4.3.

- *Divide and conquer* – divide and conquer FCM learning using RCGA, see Section 3.4.4. In this approach, the Data Divider module divides the input data without a replacement. Therefore, if the input data length is T and the number of available processors (the same as the number of submodels) is K, then each experiment uses T/K input data pairs to learn a given submodel.

- *Divide and conquer with oversampling* – divide and conquer method for RCGA FCMs learning, see Section 3.4.4. In this case, the Data Divider module selects the input data pairs with a replacement. We allow $O\%$ (oversampling coefficient) data points to be used twice to learn different submodels. Therefore, if the input data length is T, the number of submodels is K, and the oversampling coefficient is O, then each experiment uses $(T+O\%*T)/K$ input data pairs to learn a corresponding submodel. In our experiments we used different values of O, which include 25, 50, and 75, to analyze its influence on the evaluation measures.

- *Multiple population* - another parallelized version of the RCGA learning method, in which the learning module has been implemented using multiple-population

coarse-grained GAs. The number of populations is equal to the number of available processors, and each population is maintained on a designated processor. We used the design from Figure 3.2 to perform a migration among subpopulations: top 10% of chromosomes after each 1000 iterations. These values were selected experimentally taking into account the convergence to a final solution and learning time.

- *NHL* – a Hebbian-based method for FCM learning (Papageorgiou et al., 2003b), see Section 2.2.1

- *DD-NHL* – a modified Hebbian-based method for FCM learning (Stach et al., 2007), see Section 2.2.1

For the proposed "divide and conquer" and "divide and conquer with oversampling" methods we report results for experiments with random divisions of input data in the Data Divider module, as they gave better results than the results obtained when dividing the input data into contiguous intervals. Also, slightly better results were obtained when the model merging in the FCM Fusion module was carried out using a weighted average based on in-sample error for each submodel (when compared to using the average), see Section 3.4.4 for details. The remaining four approaches do not split the input data into subsets during the learning.

3.5.7 Results

Table 3.4 summarizes the experimental results and includes the three quality criteria for the six learning methods run with varying number of processors. Columns with 10, 20, and 40 nodes correspond to experiments with synthetic data, whereas columns with 13 and 24 nodes, to experiments with the real-life model. The columns with execution time and in-sample error are averaged over 5 independent experiments performed for each setup, whereas the out-of-sample error was additionally averaged over 10 independent experiments performed with different initial vectors. Each cell includes two values, the average and the corresponding standard deviation.

Table 3.4: Summary of the experimental results with the proposed scalable improvements to the RCGA approach. Rows show different learning methods, whereas columns correspond to different evaluation criteria. #p column indicates the number of processors. Rows (AV) shows the average value with the corresponding standard deviations shown below (SD).

	# p		Execution time [s]					Out-of-sample error				
			10 nodes	13 nodes	20 nodes	24 nodes	40 nodes	10 nodes	13 nodes	20 nodes	24 nodes	40 nodes
Single population	1	AV	1218	1904	3476	4901	11242	0.135	0.151	0.154	0.159	0.163
		SD	±35	±40	±47	±59	±69	±0.144	±0.148	±0.149	±0.165	±0.187
	2	AV	634	1042	1970	2719	5902	0.137	0.147	0.149	0.154	0.159
		SD	±29	±32	±38	±52	±51	±0.156	±0.154	±0.138	±0.181	±0.188
	4	AV	474	710	1316	1895	5008	0.143	0.154	0.152	0.161	0.163
		SD	±28	±28	±38	±53	±49	±0.133	±0.128	±0.172	±0.167	±0.156
	8	AV	386	476	916	1246	3784	0.145	0.152	0.155	0.161	0.166
		SD	±23	±29	±31	±43	±49	±0.136	±0.149	±0.166	±0.177	±0.148
Divide and conquer	2	AV	618	950	1770	2336	5634	0.141	0.154	0.160	0.179	0.185
		SD	±27	±32	±40	±54	±51	±0.139	±0.121	±0.145	±0.189	±0.180
	4	AV	324	482	902	1290	2836	0.144	0.160	0.167	0.173	0.176
		SD	±24	±26	±29	±40	±42	±0.162	±0.133	±0.185	±0.165	±0.188
	8	AV	170	250	454	667	1424	0.162	0.160	0.181	0.194	0.200
		SD	±8	±13	±23	±33	±34	±0.158	±0.144	±0.179	±0.185	±0.202
Divide and conquer with oversampling 25%	2	AV	680	1068	1986	2880	6591	0.140	0.153	0.158	0.165	0.174
		SD	±28	±29	±34	±50	±52	0.111	±0.150	±0.163	±0.152	±0.163
	4	AV	354	550	1041	1478	3185	0.147	0.157	0.165	0.163	0.176
		SD	±18	±19	±26	±38	±41	±0.172	±0.161	±0.183	±0.155	±0.148
	8	AV	196	295	547	733	1605	0.155	0.158	0.181	0.183	0.184
		SD	±21	±17	±23	±31	±35	±0.150	±0.141	±0.163	±0.198	±0.195
Divide and conquer with oversampling 50%	2	AV	906	1424	2648	3654	8450	0.139	0.154	0.154	0.162	0.164
		SD	±34	±37	±46	±55	±62	0.114	±0.155	±0.150	±0.145	±0.148
	4	AV	466	724	1352	2014	4246	0.149	0.154	0.161	0.165	0.167
		SD	±23	±25	±33	±47	±49	±0.156	±0.160	±0.149	±0.142	0.158
	8	AV	258	374	684	964	2140	0.155	0.157	0.169	0.174	0.182
		SD	±27	±22	±29	±33	±43	0.148	0.144	0.157	0.184	0.198

Divide and conquer with oversampling 75%	2	AV	1151	1780	3522	4754	11239	0.138	0.153	0.155	0.159	0.164
		SD	±44	±48	±60	±73	±81	±0.112	±0.161	±0.156	±0.148	±0.155
	4	AV	624	934	1812	2391	5308	0.144	0.153	0.157	0.163	0.167
		SD	±30	±32	±42	±53	±61	±0.161	±0.146	±0.157	±0.159	±0.160
	8	AV	328	490	889	1289	2782	0.150	0.154	0.162	0.171	0.181
		SD	±35	±29	±36	±52	±58	±0.154	±0.152	±0.148	±0.169	±0.172
Multiple population	2	AV	618	1022	1888	2605	5830	0.138	0.148	0.143	0.150	0.159
		SD	±29	±35	±40	±54	±57	±0.134	±0.129	±0.163	±0.159	±0.168
	4	AV	480	682	1176	1752	4998	0.141	0.149	0.151	0.161	0.164
		SD	±22	±26	±40	±49	±53	±0.156	±0.167	±0.149	±0.172	±0.180
	8	AV	370	460	892	1276	3676	0.137	0.148	0.152	0.157	0.160
		SD	±25	±22	±31	±44	±48	±0.133	±0.139	±0.144	±0.153	±0.155
DD-NHL	1	AV	42	46	96	130	216	0.212	0.222	0.211	0.215	0.218
		SD	±2	±3	±3	±5	±4	±0.180	±0.211	±0.225	±0.210	±0.203
NHL	1	AV	41	45	94	128	214	0.220	0.226	0.216	0.219	0.221
		SD	±2	±3	±3	±4	±4	±0.226	±0.198	±0.208	±0.190	±0.201

The fastest methods are those based on the Hebbian learning, i.e., NHL and DD-NHL. They learn the connection matrix weights based on a simple formula that performs local adjustments, which quickly converge into the final solution. From among the four other methods that use genetic optimization the divide and conquer RCGA approach outperforms both single and multiple population GAs. When compared to the sequential learning with a single processor, the gain in computational time is, on average, 79% and 85% for the divide and conquer RCGA with and without oversampling, respectively, when using 8 processors. In the case of the two other methods, i.e., the single and the multiple population, the gain between the sequential implementation and when 8 processors are used is approximately 70%. We also observe that doubling the number of processors results in almost a 50% (between 46% and 50%) decrease of the execution time for the divide and conquer methods. On the other hand, we observe decreasing returns when we increase the number of processors for both the single and multiple population GAs. For instance, the decrease in execution time for the single population GA when doubling the number of processors for a 40-nodes FCM is 43% for 2 processors, 33% for 4 processors, and 30% for 8 processors. This is due to the fact that

the execution time which corresponds to the sequential constraints of performing genetic operations on a population of chromosomes becomes a more significant part of the total execution time. The above overhead is caused because the fitness function evaluations performed in parallel are completed more quickly on a larger number of processors. For the multiple population GA, these numbers are 45%, 37%, and 24%, respectively. The decreasing return in this case stems from the fact that there is a nonlinear relationship between the population size and the execution time, as well as the migration among subpopulations.

Overall, excluding the Hebbian-based methods, the two methods based on the divide and conquer approach to learn FCMs are better than the methods that parallelize genetic algorithms using either single or multiple population. A comparison of the execution time of the divide and conquer method without oversampling with the single population method on 8 processors reveals that we can almost double the size of FCMs that are learned using the former method within the same amount of time. For instance, using the former method, the learning of 40-nodes FCM takes 1424sec., whereas the time needed to learn 20-nodes FCM using the latter method is 1316sec. The execution time increases by *O%* on average for *O%* oversampling when compared to the divide and conquer approach; this observation is consistent over all considered setups.

In order to facilitate an analysis of the out-of-sample error, the statistical paired t-test analysis for all tested methods has been reported in Table 3.5. The goal was to compare all other methods to the one that outperformed the rest, i.e., multiple population, and analyze whether the differences are statistically significant.

Table 3.5: Statistical comparison of the learning quality amongst the proposed methods. Results of the T-test at 95% confidence level are reported. The test compares the out-of-sample errors of all methods with the errors of the best performing multiple population method. Equality (=) represents statistically insignificant differences, whereas plus (+) denotes that multiple population method has led to statistically significantly better results.

Learning method	# processors	FCM size				
		10	13	20	24	40
Single population	2	=	=	=	=	=
	4	=	=	=	=	=
	8	=	=	=	=	=
Divide and conquer	2	=	=	=	+	+
	4	+	+	+	+	+
	8	+	+	+	+	+
Divide and conquer with oversampling 25%	2	=	=	=	+	=
	4	+	+	+	+	+
	8	+	+	+	+	+
Divide and conquer with oversampling 50%	2	=	=	=	+	=
	4	=	=	+	+	+
	8	+	+	+	+	+
Divide and conquer with oversampling 75%	2	=	=	=	=	=
	4	=	=	=	+	=
	8	+	+	+	+	+
DD-NHL	1	+	+	+	+	+
NHL	1	+	+	+	+	+

The differences between single and multiple population methods are shown to be insignificant across all experiments that include various numbers of processors and sizes of maps. The divide and conquer approach without oversampling performs similarly to the best method for small maps and only in the case when 2 submodels are used, i.e., when 2 processors are utilized. Increasing the amount of oversampling leads to improved results. When using the 50% oversampling, the quality improves and the results are similar to the best results for all map sizes when using 2 processors and for smaller maps when using 4 processors. By increasing the oversampling to 75%, the differences are statistically insignificant for all setups except when using up to 4 processors. We observe

that the RCGA method provides higher quality models when more data points are available, which results in similar performance to methods based on the parallelization of GA and the divide and conquer strategies. For instance, when comparing the divide and conquer strategy with 50% oversampling to the multiple population method for 4 processors and 10 nodes, the quality is comparable and the execution times equal 466 and 480 seconds, respectively. A comparison between the divide and conquer strategy without oversampling and the multiple population method for 2 processors and 20 nodes shows that the quality is comparable and the learning time is 1770 and 1888 seconds, respectively. Splitting the input data into more processors results in fewer input data pairs used to learn each submodel. This is particularly important in the context of learning maps with a larger size, where additional data pairs are needed for learning in order to obtain a high quality solution. At the same time, the lower quality of the learned model is traded for the faster learning time. For example, the execution times equal 3784, 3676, 2140, and 1424 seconds when learning maps with 40 concepts on 8 processors using the single population, multiple population, divide and conquer with 50% oversampling, and divide and conquer without oversampling methods, are used respectively. Both Hebbian-based methods perform statistically significantly worse than the multiple population method (see Table 3.4) for all setups. For the divide and conquer approach, the out-of-sample error value decreases when the oversampling is used. On average across all setups, adding extra 25% oversampling leads to a 2-6% reduction in the out-of-sample error. The error reduction is within this range for all considered oversampling levels, i.e., 25%, 50%, and 75%. Also, it does not depend on the number of processors and, on average, it varies by as little as 2% when comparing configurations with 2 and 8 processors.

We also investigated differences with respect to the convergence to a final solution as a function of time for each method. We analyzed how fast the fitness value increases before reaching a plateau. Figure 3.8 shows a sample plot that illustrates how the fitness value of the best chromosome changes over the time during learning. Four methods were compared: single population, multiple population, divide and conquer without oversampling, and divide and conquer with 50% oversampling. This plot shows the results for the experiments with 4 processors and a map consisting of 40 concepts. The simulations were stopped by reaching the maximum number of iterations, which was equal to 30,000.

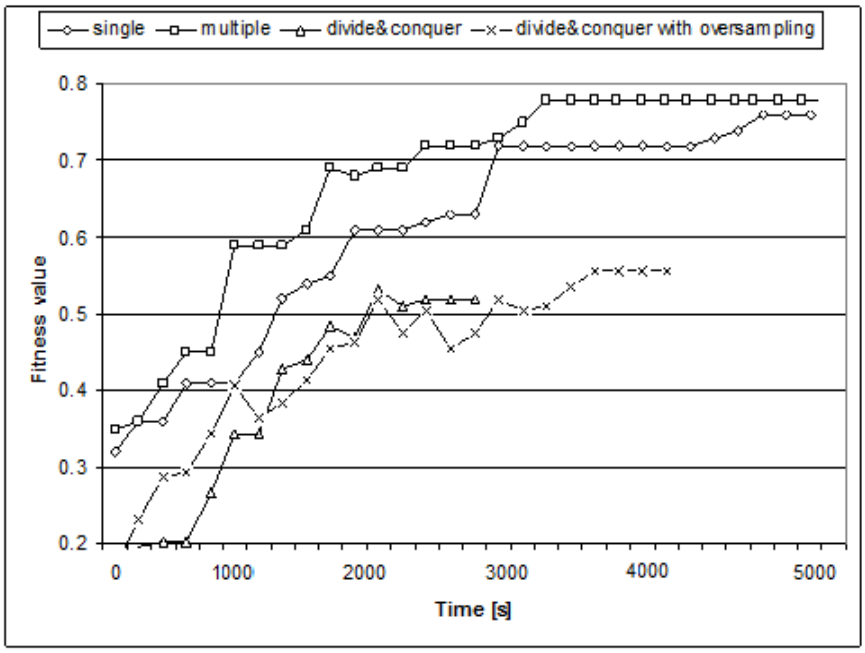


Figure 3.8: Fitness value vs. time for the four learning methods that use genetic optimization. The experiments concern learning 40-nodes maps on 4 processors with the maximum number of iterations limited to 30000.

When compared to the single population method, multi population method converges faster and provides a better final solution as well as better solutions during the simulation, i.e., at the corresponding time points. Since these two methods always forward the best solution to the next iteration in the genetic algorithm evolution, the fitness function does not decrease over time. Although the proposed methods based on the divide and conquer approach obtain lower value of the fitness function calculated for the entire input data set, they provide models of similar quality in terms of the out-of-sample error, see Tables 3.4 and 3.5. This suggests that these models are capable of generalizing the solution.

We observe that even though the fitness function value may fluctuate (decrease and increase) over the time when using the divide and conquer approaches, the overall convergence trend is clearly visible. The main reason for the fluctuations is the fact that the solution generated by the divide and conquer method is based on averaging several submodels. Figure 3.9 shows how the convergence of individual submodels and the solution FCM change over time. The fitness value of each submodel is higher than the fitness of the merged model because it is calculated only for a given subset of data that is used to learn this submodel. As the simulation proceeds and the fitness values of

individual submodels become higher, the fitness value of the merged solution increases. While the fitness value of the submodels does not decrease over time, its value for the aggregated model may decrease as can be observed after approximately 3100 seconds.

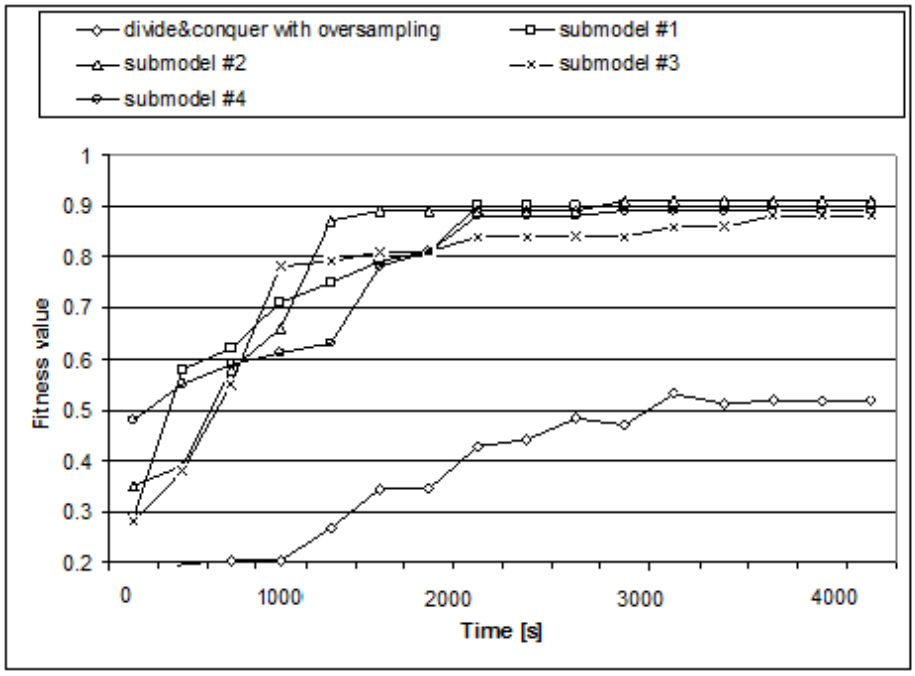


Figure 3.9: Fitness value vs. time for the divide and conquer learning method. The plot includes the fitness values of the solution FCM model and the individual submodels for the experiment that was shown in Figure 3.8.

3.5.8 Conclusions

In this section we have proposed and tested two approaches to improve the scalability of the RCGA method. Our goal was to propose a method that would substantially speed up the learning process for large systems while maintaining a high quality of solutions.

The first proposed improvement utilizes parallelization methods of genetic algorithms. We chose, implemented, and evaluated two popular parallelization strategies. One of them maintains a single population, whereas the other maintains multiple subpopulations that occasionally interact amongst each other. The results of FCM learning on 8 processors show that this method is up to four times faster than sequential learning.

The second proposed improvement is based on a divide and conquer strategy and involves dividing input data into subsets, performing independent, parallel learning of

submodels using the RCGA method, and aggregating the submodels into a final model. The two different scenarios based on this approach have been investigated. In the first one, the entire dataset is divided without oversampling, whereas in the second one, oversampling is used to increase the quality of the generated models as a trade-off of the increased execution time. The amount of the introduced oversampling could be used to control the trade-off between the learning time and quality of the generated model. Overall, the quality of the maps generated by the divide and conquer method decreases with the increasing size of the maps and number of processors used. The time to compute the maps increases with the increased size and it decreases when using more processors. Increasing the oversampling lowers the rate with which the error grows, but it also lowers the savings in computational time. When compared to the single and the multiple population-based approaches, the divide and conquer method provides larger speed ups when using more processors and when considering increasing map sizes as a trade-off for higher error rates.

The quality of the FCM model, which is measured using out-of-sample error, learned using the proposed method decreases along with increasing the number of processors. Statistical significance tests that compare quality of the FCM models obtained with the divide and conquer methods and the single and multiple population parallelization methods reveal that the method without oversampling provides comparable solutions for maps of up to 20-nodes and using 2 processors when compared with the best performing multiple populations based method. The oversampling at 50% results in improving the quality of the generated maps, i.e., their quality is comparable to the quality of the models generated by the best performing method for maps including up to 40-nodes when using 2 processors and up to 10-nodes for 4 processors. The oversampling at 75% results in comparable results across virtually all setups for up to 4 processors. The tests also demonstrate that the proposed method generates FCM models of quality that is significantly better that the quality of models computed using Hebbian-based methods.

3.6 Improved RCGA learning method using density estimate

3.6.1 Motivation

The motivation for this method comes from a structural comparison of models generated by existing fully-automated learning approaches against real-life models developed by

human experts. Table 3.6 shows a few examples of models reported in scientific literature along with the number of concepts (#concepts column) and density of connections (density column).

Table 3.6: Examples of FCMs reported in literature.

Reference	Application area	# concepts	Density
(Stylios and Groumpos, 2000)	process control	5	32%
(Aguilar, 2003)	model of a country	5	35%
(Stach and Kurgan, 2004)	software development	5	44%
(Stylios and Groumpos, 2004)	heat exchanger	5	50%
(Stylios and Groumpos, 2004)	heat exchanger performance	5	75%
(Siraj et al., 2001)	intrusion detection system	6	20%
(Espinosa-Paredes et al., 2008)	Nuclear power plant	6	50%
(Tsadiras, 2003)	e-business company	7	40%
(Papageorgiou et al., 2008a)	brain tumor characterization	9	25%
(Kosko, 1997)	virtual squad of soldiers	10	31%
(Yaman and Polat, 2009)	military planning	12	38%
(Banini and Bearman, 1998)	slurry rheology	13	39%
(Hossain and Brooks, 2008)	educational software adoption	24	8%

The majority of real-life FCM models reported in Table 3.6 are characterized by a relatively low density, which is defined as the fraction of non-zero entries in the connection matrix, in the range of 30-40%. The average model density is approximately 37% with a standard deviation of 16%. In order to examine average density of maps generated by different learning methods, we used our case study, i.e., the slurry rheology model (13 nodes, 39% density). We carried out experiments of FCM learning using different methods from data generated by simulating the model. Each experiment was repeated five times and the following values of density averages were obtained: 95% (using RCGA method) (Stach et al., 2005c), 93% (NHL) (Papageorgiou et al., 2003b), and 92% (DD-NHL) (Stach et al., 2008b).

Therefore, the newly proposed modification of the RCGA method is aimed at learning models that are structurally more similar to real-life models when compared to the FCMs obtained from current fully-automated learning approaches. The method incorporates a parameter (*density estimate*), which is used to guide the learning process towards a solution of predefined density. As mentioned in the previous paragraph, based on a number of published real-life models, the average density 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.

3.6.2 Sparse RCGA learning method

Proposed method, *Sparse RCGA*, is based on modified RCGA approach, which, similar to generic RCGA method, has a number of parameters that need to be fixed before running the simulations. In our experiments they have been set up consistently with the parameters reported for parameterization of the RCGA method (see Section 3.3.4). In contrast, however, to the generic RCGA learning, the Sparse RCGA method takes an additional parameter – the *density estimate*. This parameter is used to guide the learning towards solutions with a predefined structure defined by their density.

Detailed procedure for the proposed Sparse RCGA learning method

The proposed algorithm works as follows:

Given

A set of concepts $C = \{C_1, C_2, ..., C_N\}$ and a sequence of their activation degrees $\left(\hat{\mathbf{C}}(0), \hat{\mathbf{C}}(1), ..., \hat{\mathbf{C}}(T-1)\right)$

Step 1: Divide data into T-1 $\left(\hat{\mathbf{C}}(t-1), \hat{\mathbf{C}}(t)\right) \qquad \forall t = 1, ..., T-1$

Step 2: Initialize a population of p chromosomes, such that each chromosome is represented as $\hat{\mathbf{E}} = \left[e_{11}, e_{12}, ..., e_{1N}, e_{21}, ..e_{2N}, ..., e_{NN}\right]^T$ where e_{ij} is a random number on the [-1, 1] interval. Each gene is reset to the value of zero with probability (1 – *density estimate*). All remaining genes are initialized with the value chosen randomly from an uniform distribution on the [-1,1] interval, excluding the value of zero.

Step 3: Calculate the fitness function for the entire population. This is carried out by decoding the candidate FCM from each chromosome, and applying Formula 3.4

Step 4: Check the stopping condition. If it is satisfied, return the chromosome with the highest fitness function value

Step 5: Apply a crossover operation on the population

Step 6: Apply a mutation operation on the population. Firstly, for a given gene, the density of the chromosome that includes the gene is calculated. It is defined as the number of non-zero genes over the total number of genes. If this value is lower than or equal to the density estimate, the mutation is carried out, otherwise the gene is reset to the value of zero.

Step 7: Calculate the fitness function for the entire population

Step 8: Check the stopping condition.

Step 9: Generate new population using selection operations

Step 10: Go to *Step 5*

Hence, the Sparse RCGA method, using the modified population initialization and mutation operators (*Step 2* and *Step 6*), guides the learning towards the solutions of a given density.

3.6.3 Evaluation criteria

The proposed method has been evaluated based on both the structural and dynamic quality of the developed models. In particular, the following evaluation measures were applied:

- Out-of-sample error – measures the quality of the solution based on its dynamic properties performed on unseen data, see Formula 3.6
- Matrix error – evaluates the candidate FCM structure against the input model. It is defined as a normalized average of absolute errors between corresponding weights in the connection matrices:

$$matrix\ \ error = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|e_{ij} - \hat{e}_{ij}\right| \tag{3.7}$$

where $\hat{e}_{ij}$ is the relationship strength from concept C_i to concept C_j in the input model, e_{ij} is the relationship strength from concept C_i to concept C_j in the candidate FCM, and N is the number of concepts.

The structural evaluation was extended and transformed into a binary classification problem with two classes, zeros and non-zeros, which are defined for all map weights. Each weight from both the original FCM and the candidate FCM is assigned to one of two classes. Each comparison of the corresponding weights from the input FCM with these from the candidate FCM results in one of four outcomes, *TP* – correctly identified zero, *TN* – correctly identified non-zero, *FP* – non-zero (in the input model) incorrectly identified as zero (in the candidate FCM), and *FN* – zero (in the input model) incorrectly identified as non-zero (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:

$$\begin{aligned} Sensitivity &= \frac{TP}{TP + FN} \\ Specif icity &= \frac{TN}{TN + FP} \end{aligned} \tag{3.8}$$

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

$$SS\ \ mean = \frac{2 \cdot Sensitivity \cdot Specif icity}{Sensitivity + Specif icity} \tag{3.9}$$

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

3.6.4 Experimental setup

We used both synthetic and real-life data. In the first scenario, the data for each experiment (the input data matrix) were obtained by simulating randomly generated

FCMs (input models) by starting from a certain random initial vector. Next, 10 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 5 times with different input models and state vectors. We report on the average and the standard deviations from these 5 repeats. We also performed experiments with the real-life map, i.e., the case study defined in Section 2.1.5.

3.6.5 Results

Table 3.7 summarizes the experimental results with the synthetic data. For these experiments, the density estimate parameter was set to be the same as the actual density of the input FCM. The columns reporting the matrix error, specificity, sensitivity, and SS mean are averaged over five independent experiments performed with each setup, whereas the out-of-sample error was additionally averaged over 10 experiments performed with different initial vectors. The results were compared to three other learning methods, i.e., RCGA, NHL, and DD-NHL. In order to perform an unbiased comparison, the same initial population was used for the RCGA and Sparse RCGA methods (in the latter case, the genes reset was carried out after the initialization, see *Step 2* of the Sparse RCGA method). Since both NHL and DD-NHL methods use just a single initial connection matrix as their starting point (in contrary to the RCGA and Sparse RCGA 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 Sparse RCGA methods. The best (based on the in-sample error), among the 100 experiments, results were reported in Table 3.7 for both the NHL and the DD-NHL methods.

Table 3.7: Experimental results with the Sparse RCGA method on synthetic data. The #concepts and density columns define the input model. The subsequent columns correspond to the evaluation measures described in Section 3.5.3. Matrix error (Matrix), Specificity (Spec), Sensitivity (Sens), and SS mean (SS mean) values are averaged across 5 experiments for each setup (with corresponding standard deviations), while out-of-sample values are additionally averaged across 10 experiments performed from different initial vectors.

	#concepts	Density	Out-of-sample		Matrix		Spec	Sens	SS mean
			AV	SD	AV	SD			
Sparse RCGA	5	20%	0.010	±0.012	0.002	±0.002	0.85	0.93	0.89
	5	40%	0.008	±0.011	0.003	±0.003	0.87	0.91	0.89
	10	20%	0.123	±0.141	0.105	±0.198	0.58	0.90	0.71
	10	40%	0.124	±0.151	0.168	±0.200	0.69	0.85	0.76
	20	20%	0.142	±0.143	0.122	±0.219	0.49	0.86	0.62
	20	40%	0.146	±0.148	0.203	±0.295	0.59	0.79	0.68
	40	20%	0.166	±0.183	0.135	±0.235	0.36	0.84	0.50
	40	40%	0.164	±0.196	0.245	±0.288	0.48	0.72	0.58
RCGA	5	20%	0.017	±0.017	0.321	±0.381	0.93	0.12	0.20
	5	40%	0.012	±0.018	0.361	±0.372	0.96	0.09	0.16
	10	20%	0.135	±0.137	0.398	±0.322	0.96	0.11	0.19
	10	40%	0.136	±0.140	0.385	±0.316	0.94	0.11	0.20
	20	20%	0.151	±0.149	0.426	±0.346	0.94	0.09	0.16
	20	40%	0.152	±0.149	0.413	±0.376	0.94	0.08	0.14
	40	20%	0.171	±0.187	0.453	±0.385	0.94	0.08	0.15
	40	40%	0.167	±0.189	0.436	±0.368	0.96	0.08	0.15
DD-NHL	5	20%	0.199	±0.209	0.317	±0.345	0.96	0.06	0.11
	5	40%	0.197	±0.203	0.381	±0.333	0.96	0.07	0.12
	10	20%	0.201	±0.181	0.412	±0.356	0.96	0.07	0.13
	10	40%	0.192	±0.189	0.423	±0.348	0.93	0.05	0.10
	20	20%	0.201	±0.222	0.464	±0.316	0.93	0.08	0.14
	20	40%	0.203	±0.224	0.436	±0.348	0.94	0.07	0.13
	40	20%	0.198	±0.199	0.468	±0.388	0.96	0.07	0.12
	40	40%	0.199	±0.209	0.465	±0.384	0.93	0.06	0.12
NHL	5	20%	0.201	±0.195	0.345	±0.349	0.94	0.07	0.13
	5	40%	0.209	±0.197	0.346	±0.306	0.93	0.05	0.10
	10	20%	0.200	±0.215	0.420	±0.348	0.93	0.07	0.12
	10	40%	0.206	±0.217	0.435	±0.301	0.93	0.06	0.11
	20	20%	0.199	±0.222	0.461	±0.348	0.93	0.07	0.13
	20	40%	0.201	±0.222	0.468	±0.322	0.93	0.05	0.10
	40	20%	0.203	±0.190	0.489	±0.388	0.95	0.08	0.15
	40	40%	0.203	±0.193	0.498	±0.389	0.94	0.06	0.12

Results from Table 3.7 show that the Sparse RCGA method outperforms the others in terms of both the out-of-sample and the matrix errors. In order to facilitate a comparative analysis, we used paired t-test to investigate statistically significant differences between the proposed and other methods. The obtained results are summarized in Table 3.8.

Table 3.8: Tests of statistically significant differences between the Sparse RCGA and other methods. The density estimate parameter was set to be the same as the actual density of the input FCM. The number of plus signs in cells reflect different confidence levels, i.e., 98% (+++), 95% (++), and 90% (+), for the significance tests.

# concepts	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%	+	+++	+++	+++	+++	+++

The analysis of the out-of-sample error shows that the Sparse RCGA is significantly better at a 98% confidence level than the three other methods under consideration for the small and medium size maps (containing 5 and 10 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 Sparse RCGA is compared with the RCGA).

The matrix errors generated by the Sparse RCGA method, see Table 3.8, are statistically significantly better at a 98% confidence level than the errors produced by the three other methods considered for all setups. Results from Table 3.7 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 vs. 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 10 randomly generated maps of a given density and size 40 with their non-zero weights set at random. The average matrix error for these maps equals

0.18 for density of 20% and 0.34 for a density of 40%. This means that the Sparse RCGA improves over the baseline by 25% for the sparser and by 28% for the denser maps, respectively.

Additional insights concerning the structural quality of the candidate FCM can be obtained from an analysis of the three last columns in Table 3.6. Specificity determines the ratio of correctly assigned "non-zero" values whereas sensitivity – the ratio of correctly assigned "zero" values in the connection matrix of the candidate FCM comparing to the input model. Using an approach described in the previous paragraph, we calculated the baselines for these measures. The baseline specificities equal 0.18 (20%) and 0.37 (40%), the baseline sensitivities equal 0.82 (20%) and 0.63 (40%), and the baseline SS means are 0.30 (20%) and 0.47 (40%), respectively. In addition, 10 random maps without any density restrictions were generated for each setup and the following baselines were obtained: specificity 0.95 (20%) and 0.94 (40%), sensitivity 0.05 (20%) and 0.06 (40%), and SS mean 0.10 (20%) and 0.11 (40%). The analysis of SS mean values, which combine specificity and sensitivity, shows that the Sparse RCGA method outperforms the other considered approaches. Moreover, the proposed method is approximately 5 times better (for large 40-nodes maps) and about 9 times better (for small 5-nodes maps) than the baseline that does not consider 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 Sparse RCGA provides substantially higher sensitivity and lower levels of specificity. We note that both of these measures are relatively balanced in the case of the RCGA method, while the other approaches are characterized by high specificity (due to the fact that they predict virtually all weight with non-zero values), and very low specificity between 5% and 12%. Although the RCGA and Sparse RSGA methods obtain the out-of-sample errors that differ "only" at 90% significance for the 40 nodes maps (see Table 3.7), 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.

Table 3.9 shows experimental results for the real-life model. In addition to experiments that are analogous to the experiments performed with synthetic data, we also carried out an analysis of the sensitivity of the proposed method to the *density estimate* parameter. The parameter varied between 10% and 100% with 5% increments, and we investigated its influence on the quality of the candidate FCM.

Table 3.9: Experimental results with the case study. The density column reports the density estimate parameter used with the Sparse RCGA learning.

	Density	Out-of-sample		Matrix		Spec	Sens	SS mean
		AV	SD	AV	SD			
Sparse RCGA	10%	0.141	±0.138	0.190	±0.317	0.10	0.96	0.17
	15%	0.139	±0.139	0.187	±0.322	0.20	0.93	0.32
	20%	0.137	±0.141	0.169	±0.316	0.39	0.92	0.55
	25%	0.136	±0.132	0.150	±0.307	0.47	0.89	0.61
	30%	0.134	±0.133	0.138	±0.269	0.55	0.86	0.67
	35%	0.132	±0.131	0.120	±0.245	0.63	0.84	0.72
	40%	0.132	±0.130	0.117	±0.247	0.65	0.84	0.73
	45%	0.136	±0.134	0.123	±0.265	0.70	0.72	0.71
	50%	0.138	±0.135	0.129	±0.265	0.72	0.62	0.67
	55%	0.139	±0.132	0.135	±0.311	0.76	0.56	0.65
	60%	0.139	±0.134	0.166	±0.319	0.80	0.46	0.58
	65%	0.140	±0.131	0.179	±0.318	0.82	0.41	0.55
	70%	0.143	±0.138	0.219	±0.349	0.84	0.33	0.47
	75%	0.144	±0.139	0.239	±0.329	0.87	0.30	0.44
	80%	0.149	±0.141	0.265	±0.346	0.92	0.19	0.31
	85%	0.149	±0.144	0.305	±0.359	0.94	0.15	0.26
	90%	0.149	±0.149	0.329	±0.376	0.95	0.10	0.18
	95%	0.151	±0.137	0.346	±0.388	0.95	0.09	0.17
	100%	0.151	±0.141	0.392	±0.386	0.95	0.08	0.15
RCGA	N/A	0.151	±0.141	0.392	±0.386	0.95	0.08	0.15
DD-NHL	N/A	0.197	±0.165	0.426	±0.382	0.94	0.07	0.13
NHL	N/A	0.199	±0.184	0.436	±0.379	0.94	0.06	0.11

Both out-of-sample and matrix errors have the lowest values when the density estimate equals 40%, which is close to the actual density of 39%. These results are similar to the results for the synthetic maps with a size of 10, which is comparable to the size of the slurry rheology map. Figure 3.10 illustrates the influence of the density estimate on these two criteria for the learning of the slurry rheology map.

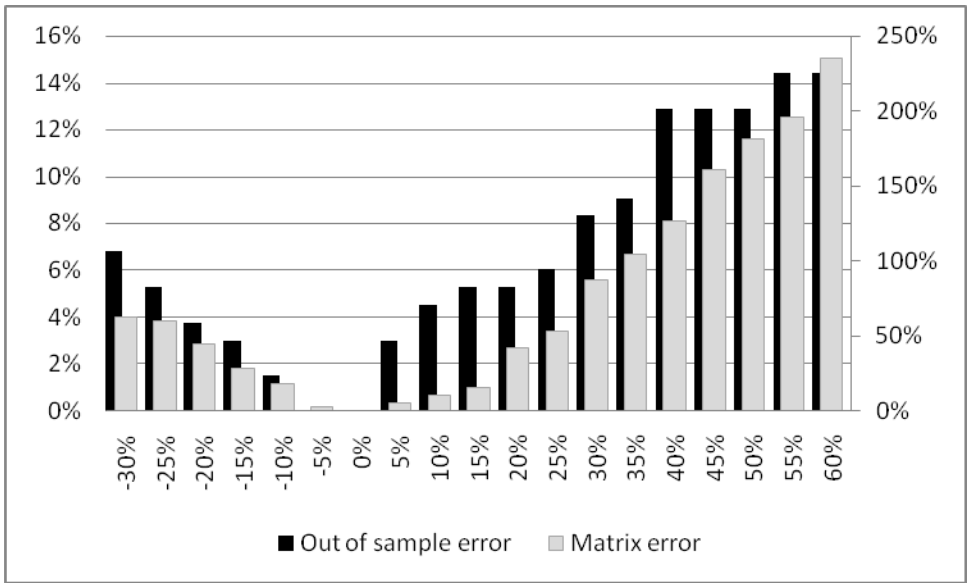


Figure 3.10: Sensitivity of the Sparse RCGA method to the setting of the density estimate parameter. Bars show a relative increase of the out-of-sample error (scale to the left) and the matrix error (scale to the right) with respect to the lowest error obtained for the density estimate that equals 40%. 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%.

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 Sparse RCGA will be still better or at least equivalent, in terms of both 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 Sparse RCGA with a density estimate of 100%, which is confirmed by the results in Table 3.8. Relatively small increases 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 behaviour. Classification results reported in Table 3.8 show that the Sparse RCGA outperforms other methods in terms of the SS mean values. In order to put these numbers into perspective, we compute the baseline by averaging 10 randomly generated maps with a density of 39%. The baseline specificity, sensitivity and SS mean equal 0.35, 0.63, and 0.45, respectively. Results from Table 3.8 demonstrates that when considering the SS mean values, the results obtained with the Sparse RCGA 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.

3.6.6 Conclusions

The proposed improved version of the RCGA approach aims at learning FCM models that are structurally more similar to real-life models when compared to 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 guide the learning process towards finding maps of a predefined density, determined by the density estimate parameter.

Experimental analysis of the Sparse RCGA 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 RCGA method, where the proposed method is statistically significantly better at the 95% and 90% level of confidence, respectively. An analysis of the structural quality expressed by the matrix error reveals that the Sparse RCGA approach performs statistically significantly better (at the 98% level of confidence) than all other considered methods for all setups. Experiments with a real-life model demonstrate that when the correct density estimate is unknown, the Sparse RCGA method is still able to develop models of quality, which is equivalent to or better than the quality offered by the other methods.

3.7 Summary

In Chapter 3, a number of new methods for learning Fuzzy Cognitive Maps from data based on genetic optimization has been introduced and thoroughly tested. Our research in this area was motivated by the shortcomings of the FCMs design practices. Disadvantages of expert-based methods and existing computational approaches call for a systematic solution for the automated development of FCMs.

The first proposed method is based on real-coded genetic algorithms (RCGA). It is fully-automated and generates FCMs from input data consisting of a single sequence of state

vectors. We investigated different setups and came up with a set of recommended guidelines for setting parameters of the employed genetic algorithm. Experiments performed with the RCGA-based method, involving both synthetic and real-life data, demonstrated that this method generates FCM models of high quality in terms of the dynamic properties (similarity of simulations between the generated and the true FCM).

Two methods for the parallelization of genetic algorithms have been experimentally evaluated to improve the scalability of the RCGA-based learner, as well as the single and the multiple population approaches. The reported performance results of the parallel RCGA methods showed that the learning of FCMs on 8 processors was four times faster than the sequential learning, i.e., it demonstrated a 2:1 ratio between the speed up and the number of processors used.

The second proposed method aims at introducing a scalable approach for FCM learning that uses the divide and conquer strategy to split the learning task into subtasks performed simultaneously. This method is tested against the two methods that parallelize the genetic algorithm. A number of configuration setups have been examined, including the usage of oversampling. The experimental results performed on 8 processors show that our divide and conquer-based solution without oversampling is up to seven times faster than the sequential learning. With 50% oversampling, the proposed method is up to five times faster than the sequential learning. We compare the proposed solution with the single and multiple population parallelizations of the RCGA method. The divide and conquer strategy without oversampling is shown to be up to three times faster than both parallelization methods.

The third method discussed in this chapter entails a modification of the RCGA method, called *Sparse RCGA*, that accommodates additional a priori information on the model's density. It was motivated by the fact that real-life FCM models are much sparser than the models generated using modern computational methods. Advantages of this improved approach are twofold: (1) it improves the model quality in terms of its dynamics (it generates simulations that are more similar to the simulations generated by the true FCM when compared to the other existing methods), and (2) it improves its structural properties. The Sparse RCGA method requires an estimate of the density as input, while the other methods do not need this input. At the same time, our empirical experiments

show that even if the correct density estimate is unknown, the Sparse RCGA method performs better or equal to the other methods.

Chapter 4

Aggregation of Fuzzy Cognitive Maps

4.1 Introduction

Designing an FCM model for a given system often includes the aggregation of knowledge from a variety of sources (Aguilar, 2005). This operation, which is the final development stage, aims to improve the reliability of the FCM model. Typically, aggregated *input models* are developed by multiple experts from the application domain. Collaboratively, experts may select a set of related concepts or each expert may be given an opportunity to define an FCM with his or her own set. The former approach assures that the connection matrices of the aggregated models are of the same size. The latter approach may result in models with different sets of concepts. In order to aggregate the maps, a pre-processing step must be carried out, which includes augmentation of matrices, see Section 2.2.2.

Table 4.1 summarizes an example of FCM models where models from different experts were aggregated to obtain the final model.

Table 4.1: Examples of aggregated FCMs reported in scientific literature. "Unavailable" values in the credibility weight columns means that the source paper does not provide this information.

Application area	Reference	Number of experts	Map size	Credibility weights	Availability of individual experts' maps
Slurry rheology	(Banini an d Bearman, 1998)	3	13	No	Yes
Control process	(Stylios and Groumpos, 2000)	3	5	No	No
Supervisory control system	(Stylios and Groumpos, 2000)	3	9	Unavailable	No
Brain tumor characterization	(Papageorgiou et al., 2008a)	3	9	Unavailable	No
Conflict management	(Noori et al., 2009)	5	17	No	No
E-learning	(Salmeron, 2009)	6	10	Unavailable	No
Military planning	(Yaman and Polat, 2009)	6	12	Yes	No

The above table lists only those examples that were supported with information on the number of experts and map size in the original reference. Many other publications include only a brief statement that a corresponding final model was obtained by aggregating models developed by a number of experts, but no additional information was provided.

4.2 Detailed goals

In this chapter, a new method for aggregating FCMs is introduced. The method aggregates individual models into a single, final FCM. The goals of the proposed research are to:

- introduce a new method to aggregate FCMs based on simulations of individual models;
- carry out well-organized, thorough empirical tests, integrating large number of configurations in terms of map sizes and number of experts;
- evaluate the new method against other approaches for the aggregation of FCMs;

- examine whether the FCM models generated by the new method preserve behavioural properties of the individual input model (whether they provide similar simulations); and

- investigate whether the proposed method can find behaviourally good solutions when assuming constraints on their structures

4.3 Motivation

The existing approaches to aggregate FCMs rely solely on the static models, i.e., connection matrices, to derive the aggregated map, see Table 2.3 in Section 2.2.2. This is adequate when the derived map is used to perform a static analysis since the aggregated connection matrix is similar to the input matrices. On the other hand, structural similarity does not imply behavioural (dynamic) similarity as we show with the following example. Let us consider the case study, i.e., FCM model of slurry rheology (Banini and Bearman, 1998). This is the only example that we found where the authors provide the individual maps developed by the experts and the aggregated map established using Formula 2.6. Although the aggregated map is structurally similar to the individual input maps and therefore the static analysis is consistent for the input maps and the derived map, the simulation results are not consistent, see Figure 4.1.

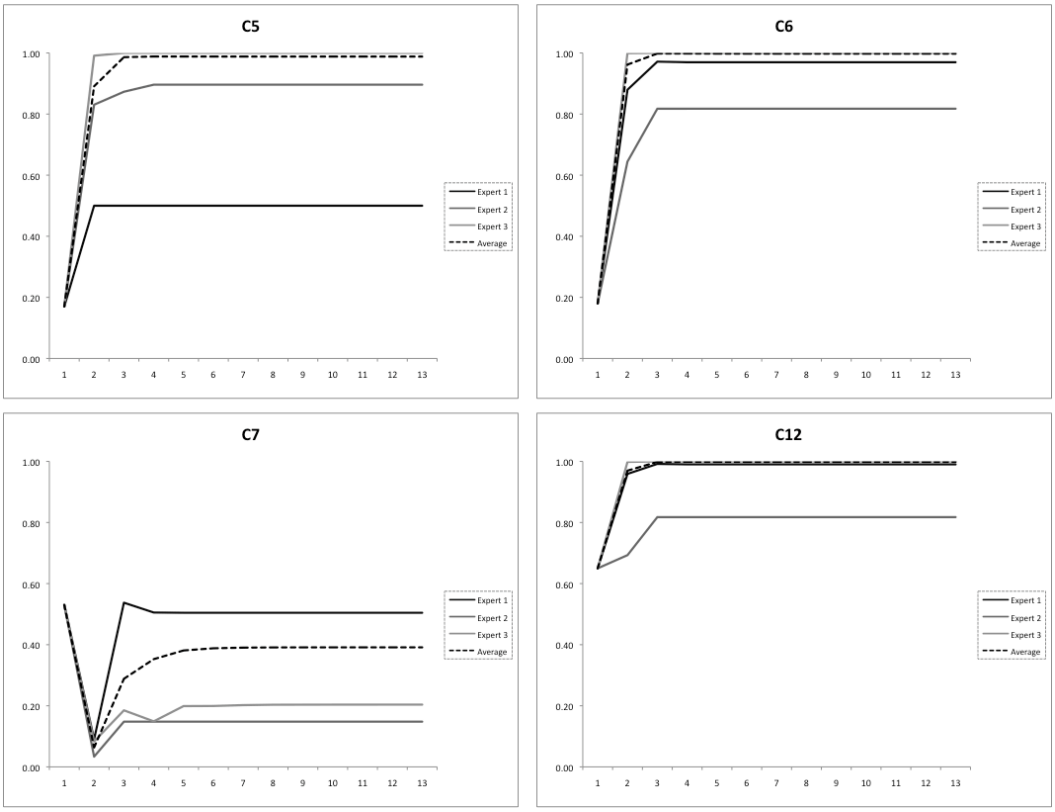


Figure 4.1: Simulation results for the four most important concepts in the case study FCM. Solid lines illustrate simulation results of maps from individual experts whereas the dotted lines show the simulation of the aggregated FCM.

Due to the large map size we focus on the simulations for the most important four concepts. The importance was calculated using a static analysis based on guidelines from (Tsadiras et al., 2001), where it is quantified as the sum of absolute values of the weights for all incoming and outgoing connections for a given concept. The four concepts with the highest summed values are C5, C6, C7 and C12. We observe that the activation levels of these four concepts converge to a certain value (stable state of the system). Assuming the same credibility factors for the three experts, the aggregated map should generate simulations that correspond to a consensus of outcomes from the individual input maps. However, in Figure 4.1 only the simulations for concept C7 are acceptable; in this case, the plot of the aggregated FCM (in dashed line) falls roughly in-between the plots based on the maps from the three experts. The simulations of the aggregated map for the other three concepts are far from being a consensus of the simulations from the three expert's maps. For instance, simulations for concept C6 show that the aggregated map follows just one of the experts and its final (stable state) value for this concept is close to 1, while the

simulations from the other two experts result in the stable state values at about 0.95 and 0.8, respectively. In essence, the combined map picks one of the experts instead of aggregating the maps from all three experts who are assumed to be equally credible. Therefore, the conclusions drawn from the dynamic analysis of the aggregated FCM could be misleading, i.e., one would conclude that the stable state for concept C6 should equal 1 and this way would disregard opinions of 2 out of 3 equally knowledgeable experts.

Methods for the aggregation of FCMs based on the dynamic properties of the individual maps have not been yet investigated, see Table 2.3. In order to fill this gap we introduce a new approach to combine FCMs which takes into account simulations of the input models.

4.4 Proposed method

The proposed approach performs a search using the real-coded genetic algorithms for an optimal aggregated connection matrix that generates simulations similar to the simulations from the input connection matrices (input FCMs). The fitness function is defined as the difference between the simulation from the aggregated matrix and the simulations from the input FCMs, and we search for the combined matrix that minimizes this difference. in fact, we search for the matrix that corresponds to the maximal values of the inverse of the difference. The RCGA is used because it is capable of finding a global maximum, which is critical since there are usually many suboptimal solutions, and because this method can search through a highly-dimensional space, which again is important since we need to find *NxN* values that make up the connection matrix. This approach is motivated by the successful prior application of the RCGA algorithm to learn connections matrices (FCMs) from a given simulation data, see Chapter 3. The main difference here is that we extend this prior work to accommodate for simulations from multiple input FCMs, instead of using a simulation from a single FCM. The following description explains the parameters used to configure real-coded genetic algorithms, which is used to perform the search.

Chromosome structure

The chromosome structure stores values of all *NxN* variables, and therefore it is defined as follows:

$$\hat{\mathbf{E}} = \left[e_{11}, e_{12}, ..., e_{1N}, e_{21}, ..e_{2N}, ..., e_{NN}\right]^T \tag{4.1}$$

where e_{ij} corresponds to the relationship strength from the concept C_i to the concept C_j and is normalized on the [-1,1] interval.

Fitness function

The fitness function utilizes state sequences generated from input FCMs by simulating them from the initial state vector **C**(0) and is defined as follows:

$$fitness = h\left(\sum_{k=1}^{K}\sum_{t=1}^{T-1}\sum_{n=1}^{N}\left(C_n(t) - \hat{C}_n^k(t)\right)^2\right) \tag{4.2}$$

where $C_n(t)$ is the value of a node *n* at iteration *t* from simulation of the map encoded by a given chromosome, $\hat{C}_n(t)$ is the value of a node *n* at iteration *t* from simulation of a model from expert *k*, *T* is the length of the simulation, *N* is the number of concepts, *K* is the number of experts, and *h* is an auxiliary function,

The auxiliary function is defined as $h(x) = \frac{1}{ax+1}$ and is explained in Section 3.3.3.

In other words, the fitness function is defined as using the sum of the least squared errors between the state sequence (simulation) generated by the map encoded by a given chromosome and the individual input maps simulated from the same initial state.

Genetic operators and stopping conditions

They have been defined and described in Section 3.3.3.

Detailed procedure for the proposed aggregated learning method for FCMs

The proposed method for aggregating FCMs is implemented using the three steps listed below.

Given

K input FCMs and an initial state vector ***C****(0)*

The individual input maps describe the same system, i.e., they were developed by different experts or learning algorithms, and we assume that they consider the same sets of concepts $C = \{C_1, C_2, ..., C_N\}$. Otherwise, the pre-processing step that unifies the input maps, see Section 2.2.2, is necessary. The initial state vector is usually given for a particular "what-if" type of dynamic analysis that a user wants to carry out with the model. Otherwise, a random input state vector could be used.

Step 1: Simulate each individual input FCM from the initial state vector **C**(0)

Each simulation is represented by *TxN* matrix (*T* is the length of the simulation, i.e., the number of iterations), which stores the values of the concepts over successive iterations calculated using Formula 2.1. These data are used in the RCGA-based optimization, instead of the individual connection matrices which are used by the existing methods for the aggregation of FCMs.

Step 2: Perform genetic optimization using RCGA

Step 2.1: Randomly initialize a population of *p* chromosomes. We also tried different initial populations based on the individual input maps, but this did not improve the results.

Step 2.2: Calculate the fitness function for the entire population. This is carried out by decoding a candidate FCM from each chromosome, and applying Formula 4.2

Step 2.3: Check the stopping condition. If it is satisfied, return the population of chromosomes

Step 2.4: Apply the crossover operation on the population

Step 2.5: Apply the mutation operation on the population

Step 2.6: Calculate the fitness function for the entire population.

Step 2.7: Check the stopping condition.

Step 2.8: Generate a new population using selection operation.

Step 2.9: Go to *Step 2.4*

Step 3: Select the solution FCM

The outcome of *Step 2* is the population of chromosomes. Given that there are no additional constraints, the aggregated map is decoded from the chromosome with the highest value of the fitness function among the entire population. However, if some constraints are imposed on the aggregated map then the chromosomes that do not fulfill these criteria are filtered out, and next the chromosome with the highest fitness value is selected from the remaining chromosomes. We use a constraint that requires the output map to be structurally similar to the maps from the input experts, i.e., to have a similar connection matrix, in our experimental analysis.

4.5 Evaluation criteria

We evaluate the proposed method on both structural (measured with matrix error) and behavioural (measured with out-of-sample error and stable state error) characteristics. A structural evaluation was performed by comparing matrices of the individual input maps and the aggregated map. A behavioural evaluation was performed using the simulation of all (input and aggregated) maps with ten different randomly chosen initial state vectors. These simulations are different than the simulations used to learn the FCMs using the RCGA-based methods to ensure that out-of-sample testing is performed. These simulation results were used to calculate the two corresponding measures, i.e., out-of-sample error and stable state error.

- Matrix error – measures the average difference between corresponding relationships in the aggregated map and the individual input maps

$$matrix = \frac{1}{K \cdot N \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \left| e_{ij} - \hat{e}_{ij}^{k} \right| \tag{4.3}$$

where e_{ij} is weight between nodes C_i and C_j in the aggregated map, e^k_{ij} is weight between nodes C_i and C_j in the map from k^{th} expert, N is the number of concepts, and K is the number of experts

- Out-of-sample error – measures the average difference between each concept value in the corresponding simulations generated by the aggregated map and the by input individual maps. The value is calculated as an average over P experiments with different initial state vector

$$out - of - sample = \frac{1}{P \cdot K \cdot T \cdot N} \sum_{p=1}^{P} \sum_{k=1}^{K} \sum_{t=1}^{T} \sum_{n=1}^{N} \left| C_n(t,p) - \hat{C}_n^k(t,p) \right| \tag{4.4}$$

where $C_n(t,p)$ is the concept C_n value at iteration t in p^{th} experiment in the aggregated map, $C_n^k(t,p)$ is the concept C_n value at iteration t in p^{th} experiment in the map from k^{th} expert, P is the number of simulations from different initial state vector, T is the maximum number of iterations, N is the number of concepts, and K is the number of experts

- Stable state error – measures the average difference between the final concepts values (at iteration T) in simulations generated by the aggregated map, and the final values of corresponding concepts generated with the individual input maps. In cases where FCMs did not stabilize but kept cycling between several states, one full cycle was taken and average concept values were calculated and used instead of the stable state values. The stable state error value is calculated as an average over P experiments with different initial state vectors.

$$stable \;\; state = \frac{1}{P \cdot K \cdot N} \sum_{p=1}^{P} \sum_{k=1}^{K} \sum_{n=1}^{N} \left| C_n(T,p) - \hat{C}_n^k(T,p) \right| \tag{4.5}$$

where $C_n(T,p)$ is the concept C_n value at iteration T (if the system stabilizes or otherwise, an average value of the concept C_n across one full cycle) in p^{th} experiment in the aggregated map, $C_n^k(T,p)$ is the concept C_n value at iteration T (if the system stabilizes or otherwise, an average value of the concept C_n across one full cycle) in p^{th} experiment in the map from k^{th} expert, P is the number of simulations from different initial state vectors, T is the maximum number of iterations, N is the number of concepts, and K is the number of experts

4.6 Experimental setup

We used both synthetic and real-life maps. The real life model concerns slurry rheology (Banini and Bearman, 1998) since this was the only aggregated real map with the individual input maps that were fully defined, see Table 4.1. We generated the input data (simulations from the input FCMs) from the initial vector suggested in the original reference.

Due to the lack of additional real-life maps, we also prepared synthetic data. We performed experiments with twelve different setups defined by a different number of experts (2, 3, 4, and 5) and different map sizes (5, 10 and 15). These are typical numbers encountered in prior works involving the aggregation of FCMs, see Table 4.1. For each setup, the map from the first expert was generated randomly. However, the remaining maps should not be random as they should describe the same system as the map from the first expert. We used our real-life slurry rheology map to guide the generation of the other maps. We calculated the set of 169 (for each of the 13x13 = 169 concepts) standard deviations (across the three maps) and these values were used to define 169 corresponding normal distributions with a mean equal to 0. Next, a given relationship (a given entry in the FCM) in the maps from the other experts are computed by taking the corresponding entry from the map from the first expert and adding to it a value δ drawn from one of the 169 distributions, which is chosen at random. The corresponding entry for each additional expert uses a different δ drawn from the same distribution. If the sum does not fall into the [-1, 1] range, then we draw another δ. This procedure enables similarity between a given set of the input synthetic maps and the maps from our real-life FCM.

4.7 Methods used for comparison

We compare against the *average* aggregation method since the other two existing methods require additional information (see Table 4.1) which is usually not available, as in the case of our real-life slurry rheology map. However, we include two additional approaches which are derived from the average-based aggregation. The first one is based solely on structural properties of the individual maps whereas the second one uses both structural and behavioural properties.

- *Median* - each value of the aggregated matrix is calculated as a median (instead of the average) across all individual maps, which is described by the following formula:

$$\forall i,j \in \{1,...,N\} \qquad e_{ij} = median\left(e_{ij}^{1}, e_{ij}^{2}, ..., e_{ij}^{K}\right) \tag{4.6}$$

  where e^{k}_{ij} is the weight (relationship strength) between nodes C_i and C_j in the map from k^{th} expert

  A median is also a central point which minimizes the average of the absolute deviations among the weights. This method does not introduce new weight values, instead the weights in the aggregated map are drawn from among the weight in the input maps, which could be considered advantageous when compared to the average.

- *Weights optimization* (weightOpt) – the aggregated matrix is calculated using Formula 2.7. The credibility weights w_k are established using the RCGA with the fitness function (Formula 4.2), and normalized such that sum of all weights equals 1. Each chromosome consists of K floating point numbers, which are optimized through a real-coded genetic algorithm to maximize the fitness function. This method uses the RCGA to find "optimal" weights for a weighted average-based aggregation, and it takes into account both structural and behavioural aspects of aggregating individual FCMs into a single map.

4.8 Results

Experimental results are summarized in Table 4.2. The first column, which includes the number of concepts and the number of experts, defines the experimental setup. The last setup (the last four rows) corresponds to the real-life model, whereas all other rows correspond to synthetic models. The second column lists methods used which are denoted as follows: *genetic* – the method proposed in this Chapter, *average* – the most commonly used map averaging-based aggregation method (see Table 2.3), *median* and *weightOpt* – two other methods based on the median weights and genetically optimized weighted average of weights, as described in Section 4.7. The next column shows the evaluation measures (see Section 4.5) values obtained from the empirical experiments. We also performed paired statistical tests for the behavioural measures (the out-of-sample and stable state errors) where we compare our method against the other three methods. For each setup, a paired t-test was carried out on the corresponding measures values calculated for each concept and for each expert. In other words, *N*K*P* values (the number of concepts multiplied by the number of experts and by the number of experiments with different initial conditions) were paired to perform the tests for each setup. The results are shown in the "Best solution" column. In addition, since the matrix error of the proposed method is substantially higher than that of the other methods, we investigated the quality of the entire population of solutions generated by our method, see Section 4.4. Instead of selecting the best chromosome (the one with the highest fitness value) from the entire population, we filtered out these chromosomes that are structurally different (measured by matrix error) by more than *X%* when compared to the median method; this is because the median-based method by definition minimizes the matrix error value for a given setup. Next, the chromosome with the highest fitness value was selected from the remaining subset of chromosomes. The same statistical tests were performed with three different thresholds of X=10%, 20%, and 30%. The results are reported in the last three sub-columns. The underlined values show cardinality of corresponding subsets of populations for each threshold that had a lower out-of-sample error than the solution obtained using the *median* method. For instance, the results for the setup with 5 concepts and 3 experts in the "Matrix error +20%" column are as outlined below. The final population returned by the proposed method includes 146 solutions that were structurally different by no more than 20% of the optimal solution measured by matrix error. From among these remaining solutions, the one with the highest fitness was

chosen to be compared with the other methods. This solution was found to be statistically significantly better in terms of both measures when compared to the average as well as to the median method ("+/+" signs), whereas the differences were not statistically significantly different when compared to the weights optimization method ("=/=" signs).

Table 4.2: Experimental results with different aggregation methods. The first two columns define the experimental setup. The last four rows correspond to the results for the real-life model. The statistical significance results are shown as α/β where α is the result of the statistical test for the stable state error, and β is the result of the statistical test for the out-of-sample error. "+" means that our Genetic method was statistically significantly better when compared to a method in a corresponding row, "=" means that the methods were not statistically significantly different.

Setup		Method	Measures			Statistical significance analysis			
Number of concepts	Number of experts		Matrix error	Out-of-sample error	Stable state error	Best solution	Matrix error +10%	Matrix error +20%	Matrix error +30%
5	2	Genetic	0.240	0.099	0.092		51	103	135
		Average	0.084	0.134	0.139	+/+	=/+	+/+	+/+
		Median	0.084	0.134	0.139	+/+	=/+	+/+	+/+
		WeightOpt	0.115	0.104	0.103	+/+	=/=	+/=	+/+
10	2	Genetic	0.335	0.126	0.145		82	127	145
		Average	0.091	0.174	0.182	+/+	+/+	+/+	+/+
		Median	0.091	0.174	0.182	+/+	+/+	+/+	+/+
		WeightOpt	0.118	0.148	0.157	+/+	+/=	=/+	+/+
15	2	Genetic	0.272	0.148	0.152		62	153	189
		Average	0.085	0.173	0.181	+/+	=/=	+/+	+/+
		Median	0.085	0.173	0.181	+/+	=/=	+/+	+/+
		WeightOpt	0.122	0.159	0.164	+/+	=/=	+/=	+/+
5	3	Genetic	0.379	0.165	0.177		83	146	192
		Average	0.092	0.203	0.212	+/+	=/=	+/+	+/+
		Median	0.082	0.217	0.226	+/+	=/=	+/+	+/+
		WeightOpt	0.101	0.18	0.185	+/+	=/=	=/=	=/=
10	3	Genetic	0.356	0.148	0.165		3	29	168
		Average	0.089	0.181	0.199	+/+	=/=	=/=	+/+
		Median	0.073	0.206	0.216	+/+	=/=	=/=	+/+
		WeightOpt	0.094	0.162	0.186	+/+	=/=	=/=	+/+
15	3	Genetic	0.380	0.201	0.147		0	28	195
		Average	0.091	0.22	0.164	+/+	N/A	=/=	=/=
		Median	0.072	0.213	0.200	+/+	N/A	=/=	=/=
		WeightOpt	0.098	0.209	0.154	+/+	N/A	=/=	=/=

5	4	Genetic	0.364	0.141	0.151		72	129	219
		Average	0.082	0.158	0.164	+/+	=/=	+/+	+/+
		Median	0.070	0.166	0.171	+/+	=/=	+/+	+/+
		WeightOpt	0.103	0.151	0.156	+/+	=/=	=/=	=/=
10	4	Genetic	0.266	0.182	0.178		21	49	138
		Average	0.094	0.219	0.187	+/+	=/=	=/=	+/+
		Median	0.079	0.214	0.188	+/+	=/=	=/=	+/+
		WeightOpt	0.119	0.191	0.183	+/+	=/=	=/=	=/=
15	4	Genetic	0.363	0.176	0.16		39	98	147
		Average	0.088	0.188	0.17	=/+	=/=	=/=	=/=
		Median	0.082	0.198	0.192	+/+	=/=	=/=	=/=
		WeightOpt	0.102	0.182	0.166	=/=	=/=	=/=	=/=
5	5	Genetic	0.288	0.141	0.153		41	73	108
		Average	0.091	0.169	0.174	=/+	=/=	=/=	+/+
		Median	0.089	0.174	0.168	+/+	=/=	=/=	+/+
		WeightOpt	0.099	0.161	0.153	=/+	=/=	=/=	=/+
10	5	Genetic	0.238	0.197	0.187		0	43	98
		Average	0.095	0.209	0.192	=/+	N/A	=/=	=/=
		Median	0.082	0.201	0.201	+/+	N/A	=/=	=/=
		WeightOpt	0.105	0.198	0.186	=/=	N/A	=/=	=/=
15	5	Genetic	0.251	0.152	0.204		2	42	108
		Average	0.097	0.188	0.201	=/=	=/=	=/=	=/=
		Median	0.089	0.203	0.211	+/+	=/=	=/=	=/=
		WeightOpt	0.108	0.174	0.208	=/=	=/=	=/=	=/=
13	3	Genetic	0.28	0.148	0.142		29	72	183
		Average	0.094	0.195	0.164	+/+	=/=	+/+	+/+
		Median	0.073	0.198	0.214	+/+	=/=	+/+	+/+
		WeightOpt	0.102	0.189	0.154	+/+	=/=	=/+	+/+

The results show that for each setup with up to 4 experts and 10 concepts (this setup is the most similar to the average real-life model reported in Table 4.1), the proposed method performs statistically significantly better than the other three methods. For setups with more experts and/or concepts, the results are not statistically significantly better, although our method still scores better than or similarly well as the other approaches.

When compared to the most commonly used *average* method, the proposed method is statistically significantly better in 12 out of 13 setups (including the real-life model). On average, the out-of-sample error is smaller by 16% and this improvement decreases when the number of experts increases (23% for two experts and 13% for five experts across all experiments with different map sizes). This might be caused by the increased complexity

of the optimization task. On average, the absolute value of this error increases from 0.12 (two experts) to 0.16 (five experts). The out-of-sample error increases slightly when comparing the average values across different map sizes for a given number of experts from 0.13 (for 5 concepts) through 0.16 (for 10 concepts) to 0.17 (for 15 concepts). These results are better by 0.3 to 0.4 when compared to the *weight optimization* method, which scored second best.

Similar to the out-of-sample error, the stable state error for the proposed method is statistically significantly lower when compared to the error for all other methods; this is consistent for each setup of up to 4 experts and 10 concepts. For setups with more experts, our method outperforms or is similar to the other solutions, but these improvements are not statistically significant.

When compared to the *average* method, our solution is statistically significantly better for 19 out of 23 setups. The differences are not statistically significantly different for the setups with five experts and for one setup with four experts, and the largest considered map with 15 concepts. On average the stable state error is smaller by 12%. Due to the same reason as for the out-of-sample error, the average improvement offered by the proposed method decreases when the number of experts increases: from 23% for two experts to 4% for five experts. On average, the absolute value of this error increases from 0.13 for two experts to 0.17 for five experts. The average stable state error across all number of experts increases by 16% (from 0.14 to 0.17) when the number of concepts increases from 5 to 15.

The proposed aggregation method does not use structural information, i.e., the connection matrices from the input maps. Consequently, the matrix error that measures structural similarity is considerably higher for our method when compared to the other three methods. Since the median value by definition minimizes the average of the absolute deviations among a set of given numbers, the connection matrices computed using the *median* method correspond to the optimal solution in terms of the matrix error criterion. Any other method can only perform equally well or worse than the *median* approach. However, since the proposed method returns a population of results (each chromosome in the final population is a solution) instead of selecting the chromosome with the lowest quality index value from the entire population, we filtered out the chromosomes with the matrix error value higher by *X%* when compared to the *median* method. From among the

remaining chromosomes, we selected only those with a lower out-of-sample error value than the result of the *median* method. Cardinality of this set is reported in Table 4.2 (using the underlined numbers) and it shows how many solutions of good structural quality are still better in terms of simulation error than the median-based solutions. For X=20%, the proposed method finds more than 25 of such solutions for each setup. Therefore, our method is capable of finding high quality maps in terms of structural properties, which are still better than the solutions generated by the other methods in terms of dynamic properties.

We use simulation results for the slurry rheology model to put the measures reported in Table 4.2 into perspective. Figure 4.2 gives simulations for the four most important concepts from this model (the same concepts as in Figure 4.1). Each figure includes plots obtained using simulations of three individual models proposed by experts as well as plots obtained using simulations of aggregated maps generated by the four considered methods.

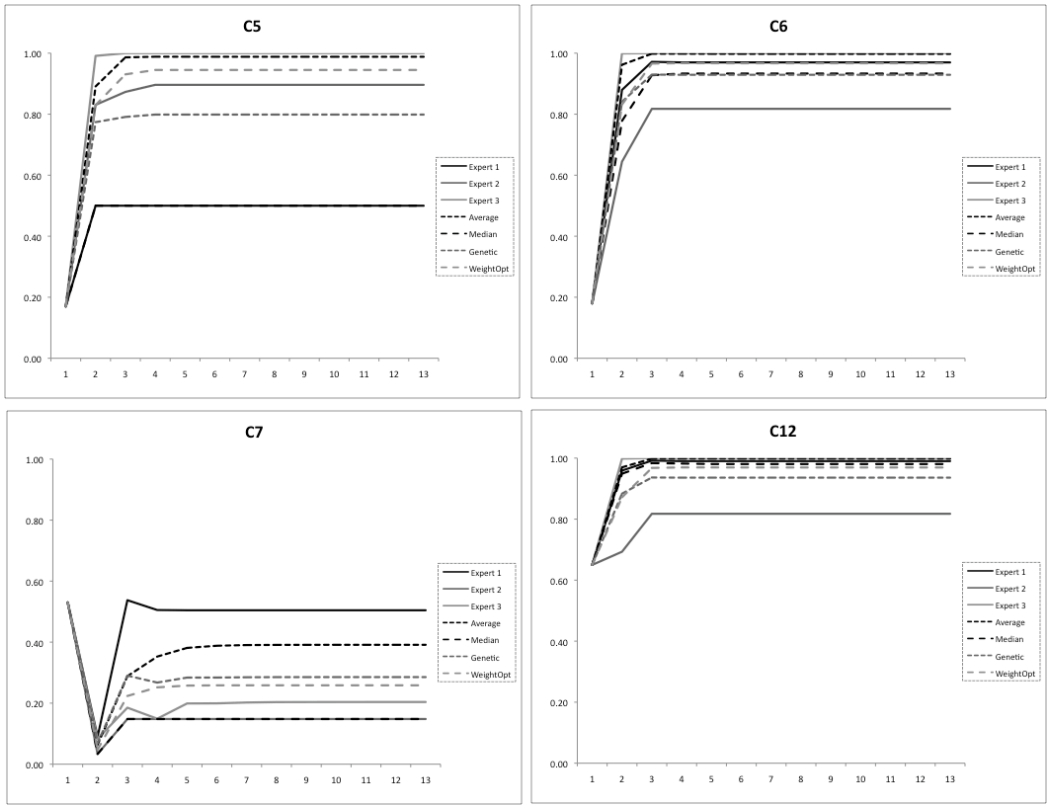


Figure 4.2: Case study: simulation outcomes of the four most important concepts for different aggregation approaches. Solid lines illustrate simulation results of maps from individual experts whereas dotted lines illustrate simulations of the four aggregated FCMs.

When compared to the other three methods, the proposed approach generates simulations that better reflect a consensus of the dynamic properties of the simulations from maps produced by the individual models. Given that the experts are equally knowledgeable, which has been usually assumed in real-life examples including our case study (see Table 4.1), the simulation for a given concept should be based on a consensus of the individual plots for the same concept generated from maps proposed by different experts. This is shown in Figure 4.2 where our method generates simulations that average the simulations from all three experts. This is particularly transparent for concepts C5, C6, and C12, where our solution generates a "consensus" simulation, while the other methods essentially try to closely mimic simulations from one (or two) of the experts while ignoring the other expert(s). While the above simulations offer just one snapshot for one real-life model, and thus could not be categorized as "typical", they offer an explanation why the proposed approach is characterized by a favourable quality with respect to its simulations.

4.9 Conclusions

Developing a reliable FCM model for a given system is a challenging task. Models established by a single expert are vulnerable to bias and inaccuracy. Credibility of modeling with FCMs can be improved by aggregating maps from multiple experts. However, surprisingly only a few simple methods have been proposed to aggregate FCMs and all of them operate at the structural level, which may not be suitable when dynamic analysis is performed using an aggregated model.

We propose a new approach for the aggregation of FCMs, which operates on the behavioural (simulation) level and utilizes data from the simulation of individual input models instead of their structures. Experimental results demonstrate that aggregated models obtained using the new method better preserves the dynamic properties of the input models, as defined by the out-of-sample error and the stable state error, when compared to existing aggregation methods. These two measures have been selected since they are often used in practical applications of FCMs; they play a crucial role when dynamic modeling with Fuzzy Cognitive Maps is considered (Aguilar, 2005; Tsadiras et al., 2008; Papageorgiou et al., 2009).

Experimental studies show that the new method is statistically significantly better than other considered methods with respect to the abovementioned criteria when up to four experts and ten concepts are involved. Importantly, these setups cover many of the typical FCM models that involve aggregation, as shown in Table 4.1. When applied to larger systems and problems that involve more experts, our method still outperforms, or at worst works similarly well, when compared to the modern approaches. An evaluation of the proposed method, in terms of the structural quality of its solutions, was performed using the matrix error, which measures how much the aggregated map is different from the individual input maps. Despite the fact that our method provides a map that may be structurally significantly different from the input maps proposed by the experts, the final population of solutions generated by our genetic algorithm-based algorithm virtually always includes solutions that are worse only by a maximum of 20% from the structurally optimal solution (the median map) but better with respect to dynamic simulations. Therefore, our method could be used when the aggregated maps need to have high structural and dynamic quality.

Chapter 5

Conclusions

5.1 Empirical results for the case study

Chapters 3 and 4 discuss methods for the development and aggregation of FCMs and include empirical evaluations that were performed on somewhat inconsistent setups defined by the map sizes and their densities. This resulted from the fact that some methods needed to be evaluated using specific setups, that these results were generated over a long period of times, and that they were influenced by the peer-review review process. This section aims to provide a consistent set of empirical results across all proposed and existing methods on the same real-world FCM. It summarizes the experimental results for the case study, i.e., the slurry rheology model (Banini and Bearman, 1998), and is divided into two subsections: results for learning FCMs from data, and a summary of experiments for FCMs aggregation.

5.1.1 Learning FCMs from data

Table 5.1 presents a summary of the results of the FCM learning task. The two reported criteria include the execution time and the out-of-sample error. The former quality index is used to perform a side-by-side comparison between the proposed generic RCGA-based learning methods and the divide and conquer / genetic parallelization – based methods that aim to improve the scalability of the genetic-based FCM learning. The latter criterion evaluates a given method based on out-of-sample (never used in the training of the model) simulations of the corresponding model. For the *Parallel RCGA* and *Divide and conquer* (D&C) *RCGA* methods, the experiments utilizing 8 processors were selected.

Table 5.1: Case study: summary of the experimental results for the learning of FCMs. Parallel RCGAs include results for both parallelization types: (S) single poplulation, and (M) multiple-population. The Divide and conquer method results without oversampling (0%) as well as with 75% oversampling (75%) are also reported. The Sparse RCGA method was executed with a default density estimate parameter of 37%. In addition, Hebbian-based methods were executed from a random initial condition, or from a 100 random initial conditions, rows NHL(100) and DD-NHL(100), and the best solution was reported. The bold font shows the lowest value across all the genetic-based methods for a given measure.

Method	Time [s]	Out-of-sample error
RCGA	1904	0.151 ±0.148
Parallel RCGA (S)	386	0.152 ±0.149
Parallel RCGA (M)	370	0.148 ±0.139
D&C RCGA (0%)	**250**	0.160 ±0.144
D&C RCGA (75%)	490	0.154 ±0.150
Sparse RCGA	2102	**0.132** ±0.130
NHL	45	0.226 ±0.198
NHL (100)	4458	0.199 ±0.178
DD-NHL	46	0.222 ±0.211
DD-NHL (100)	4623	0.195 ±0.178

The Sparse RCGA is the method that provides the best solution in terms of the out-of-sample error. The results were obtained assuming the default value of the density estimate parameter, which was coincidently, relatively close to the real density of the case study (37% vs. 39%). However, as shown in Table 3.8 in Section 3.5.5, the Sparse RCGA performs better or equally well when compared to the RCGA for any density estimate value. The divide and conquer method was the fastest of all genetic optimization-based approaches. When compared to the sequential RCGA, this method was approximately 7 times faster. The quality of its solution is worse by approximately 6% (based on the out-of-sample error) when compared to the best performing Sparse RCGA. The time-quality ratio could however, be controlled by using oversampling. Parallelization approaches provide substantial speedups (about 5 times) over the sequential RCGA for a relatively small decrease in the out-of-sample error.

The learning quality of Hebbian-based methods can be improved by starting the learning from multiple initial conditions, but even when executing the Hebbian-based methods from the same number of initial conditions as the population size in genetic-based algorithms (see the NHL(100) and DD-NHL(100) in Table 5.1) , the quality is still lower while the total execution time is longer.

5.1.2 Aggregation of FCMs

Table 5.2 summarizes the results obtained for the case study when considering the aggregation of the FCMs. The three criteria reported measure both structural (matrix error), and behavioural (out-of-sample error and stable state error) quality.

Table 5.2: Case study: summary of experimental results for the aggregation of FCMs. "Genetic" refers to the proposed method. The other three methods are described in Section 4.7. The last column includes statistical significance results, which are shown as α/β, where α is the result of the statistical test for the stable state error, and β is the result of the statistical test for the out-of-sample error. "+" means that our Genetic method was statistically significantly better when compared to a method in a corresponding row, and "=" means that the methods were not statistically significantly different.

Method	Matrix error	Out-of-sample error	Stable state error	Matrix error +20%
Genetic	0.280	**0.148**	**0.142**	72
Average	0.094	0.195	0.164	+/+
Median	**0.073**	0.198	0.214	+/+
WeightOpt	0.102	0.189	0.154	=/+

The new aggregation method provides the best solution based on both the out-of-sample and the stable-state errors that measure similarity between the simulations of the aggregated map and the simulations of the individual models proposed by the experts. A structural analysis of the entire population of the solutions returned by the proposed method shows that it includes solutions that are worse by no more than 20% of the structurally optimal solution (the map generated by the *median* method) and better with respect to the dynamic simulations, when compared to methods based on the structural aggregation, and the *weightOpt* method based on the out-of-sample error.

Although both the *average* and *median* methods perform the aggregation based solely on the structures of the input maps and they provide the solution with lower *matrix error*

values, their behavioural quality is statistically significantly worse when compared to the proposed approach (based on the t-test performed at a 98% confidence level with both out-of-sample and stable state measures). The method that optimizes the experts' credibility weights (*weightOpt*) offers a trade-off solution between the structural- and the dynamic-oriented properties when compared to median- and average-based aggregations.

5.2 Summary

This dissertation tackles two challenging tasks that concern the design of Fuzzy Cognitive Maps, namely learning FCMs from data and the aggregation of multiple FCM models. The two investigations resulted in a fully-automated genetic algorithm-based method for learning FCMs and a new method to aggregate FCMs that utilizes simulations of individual input models.

The proposed fully-automated learning method delivers high quality models when dynamic properties are considered. This method can potentially replace a domain expert when input data are available. It optimizes internal connections among concepts in the FCM using real-coded genetic algorithms (RCGAs). Our solution was empirically shown to outperform existing methods that are based on Hebbian learning. The RCGA-based learner has already been successfully applied in real-life applications to predict secondary structure of proteins (Kurgan et al., 2007) and to perform time-series predictions (Stach et al., 2008a).

Two avenues of improvements for the RCGA method were also explored. The first one addresses scalability, since genetic optimization is complex and time consuming. The poor scalability of the generic RCGA approach makes it ineffective in its applications to large systems that consist of a few dozen concepts. We considered two strategies for the parallelization of the genetic algorithm as the most natural and problem-independent way to solve this problem. We also utilized a divide and conquer strategy, which involves dividing input data, learning submodels, and aggregating them into a final model. The latter approach is problem-specific as it draws from the fact that the FCM submodels can be relatively easily combined into a final solution. We analyzed and discussed trade-offs between the above approaches and empirically compared them to the generic FCM learner. The motivation behind exploring the second avenue of the RCGA improvement was due to the following observation: learned FCMs are much denser than the real-life

models. This a priori knowledge was used in the proposed Sparse RCGA method to improve the learning quality by guiding the learning process towards models of a given density.

Another major topic addressed in this dissertation concerns the aggregation of Fuzzy Cognitive Maps. A new method that aims at preserving dynamic properties of the individual models and which uses the real-coded algorithms-based optimization, was proposed.

The main findings from the research reported in this dissertation are summarized in Section 5.3.

5.3 Major contributions

The major contributions of this dissertation include:

- the development of a novel, genetic optimization-based method (RCGA) for learning Fuzzy Cognitive Maps from data
- a thorough, empirical evaluation of the method, which includes its parameterization, tests with different map sizes and densities, and a comparison against other existing approaches for the automated learning of FCMs
- an empirical analysis of the relationship between the quality of the learned FCM model and the size of the input data
- an evaluation of the proposed learning method on real data, with application to time series prediction, and comparative analysis with other state-of-the-art predictors based on fuzzy sets
- the development and assessment of two types of solutions: genetic algorithm parallelization and divide and conquer approach, to improve the scalability of the RCGA method
- a first-of-its kind evaluation of learning of large size FCM model, as existing methods were evaluated on map sizes below 10 nodes, while our analysis is comprised of maps with up to 40 nodes

- a first-of-its kind assessment of the running time necessary for automated learning of FCMs from data
- an analysis demonstrating that current automated methods for learning FCMs generate maps that are too dense when compared to the underlying true maps
- the development and evaluation of the Sparse RCGA method that uses a priori knowledge on the map density to improve the quality of the learned FCM models
- an analysis showing that current structure-based methods for the aggregation of FCMs do not preserve the dynamic (simulation-based) properties in some cases
- the development of a new method for the aggregation of FCMs based on simulations of individual input FCM models
- a thorough empirical evaluation of the new aggregation method, which includes a comparison against other existing structure-based aggregation methods

Below we outline the most important findings for the aforementioned studies.

Learning FCMs from data

Real-coded genetic algorithms proved to be an efficient approach to build a fully-automated approach for the learning of FCMs. The proposed RCGA method outperforms existing Hebbian learning-based approaches when considering behavioural properties, i.e., simulation results. The parameterization of the method provides firm design guidelines that include RCGA parameters and fitness function selections. A comparison of three fitness functions was performed and suggested the function is based on Euclidean distance. In addition, experimental results demonstrate that the optimization of the map to the input data is a challenging task with many sub-optimal solutions. Learning quality and the convergence of the RCGA method to the optimal solution improves as the size of input data increase. For small maps that consist of five concepts the input data length should be more than 10, whereas for maps consisting of ten concepts, more than 20 should be used to allow for quality RCGA learning. Comprehensive experiments performed with different setups, including maps with up to 40 concepts show that the genetic-based method is capable of providing solutions of statistically significantly better quality (based on t-tests with a 98% confidence level) when compared to Hebbian-based

methods, even for large models. When applied to time series prediction on real data, FCMs with the RCGA, learning outperforms or performs similarly to other state-of-the-art predictors based on fuzzy sets, and provides the unique feature of linguistic prediction. A running time analysis reveals an exponential relationship between the RCGA learning time and model size for the genetic-based methods. Two approaches to study the scalability enhancements of the proposed method were investigated. The former, which concerns the parallelization of the genetic optimization, resulted in a 2:1 ratio between the learning speed up and the number of processors (for 8 processors), without a significant deterioration of the quality of the resulting solution. The latter method, which utilizes a divide and conquer strategy, resulted in a 8:7 ratio (for 8 processors), though our study shows that the increase in the number of processors negatively affects the quality of the model. Data oversampling was used to compensate and control the loss of the quality. The oversampling rate defines the trade-off between learning speed up and the quality of results. A map with 40 concepts can be learned in about 20 minutes on a desktop computer using the divide and conquer approach, while RCGA learning takes more than 3 hours. Another proposed modification of the RCGA is aimed at improving the quality of the learned model by including a priori knowledge of structural properties (density of the map) of the generated map. The following observation was made: models obtained from existing learning methods are much sparser than real-life FCMs (over 90% vs. less than 40%). The results demonstrate that a statistically significant improvement of learning quality can be achieved when the correct density estimate is used. A study on the effects of the over- or under-estimation of the density estimate on the learning quality shows that in the worst case scenario, this method performs equally well as the generic RCGA method. Literature research suggests that the density estimate of around 37% should be used in the case where no other knowledge is available.

Aggregation of FCMs

A new method for FCM aggregation based on real-coded genetic algorithms was introduced. Experimental results show that the current methods which are based solely on the structures of individual input maps are not always reliable when dynamic analysis is the objective. For most of the setups considered in our empirical analysis, including the typical FCM setups reported in literature, the proposed method outperformed other approaches when evaluation based on preserving behavioural properties, such as the ability to reach a desired stable state of the individual models, was performed. For setups

using larger numbers of experts and/or concepts, the new method either outperforms or performs equally well as the other approaches. A structural analysis of the solutions returned by the proposed method reveals that they virtually always include solutions that are worse by no more than 20% from the structurally optimal solution and which are better with respect to dynamic simulations. Hence, the proposed method is applicable for aggregation tasks where both structural and dynamic quality of aggregated models are required.

5.4 Limitations and further directions

The studies undertaken contribute to the area of FCM design. The limitations and future directions that are listed below relate specifically to the two types of investigations performed in this dissertation.

Learning FCMs from data

Although the scalability of the RCGA method was improved, it still may be not sufficient for maps that consist of over a hundred concepts. In addition, when using the divide and conquer approach, the most time-efficient method, the quality of learning decreases when the number of processors used increases.

Further research direction may include the application and evaluation of other paradigms to FCM learning, such as gradient descent. This approach should be faster and could scale better than the genetic algorithms-based optimization, though a large number of suboptimal solutions may result in a generation of FCM models that are inferior to those generated by the proposed RCGA method.

Aggregation of FCMs

An experimental analysis shows that the proposed method, though effective for smaller problems, does not scale well for larger setups in terms of preserving dynamic properties of the individual models.

Future research direction may include experiments with the parameterization of the RCGA method and/or the fitness function used to tackle this optimization task for

different setups towards improving the method's quality in its application to larger problems.

Another interesting approach would be to develop a hybrid method which combines optimization based on (a) the structure (the input connection matrices) and (b) the behaviour (the simulations of the input connection metrics). Such a design could lead to a more balanced trade-off between the structural and behavioural quality of the aggregated maps.

Bibliography

Aguilar J., A Dynamic Fuzzy-Cognitive-Map Approach based on Random Neural Networks, *International Journal of Computational Cognition,* pp. 91-107, 2003

Aguilar J., A Survey about Fuzzy Cognitive Maps Papers, *International Journal of Computational Cognition,* vol. 3, no. 2, pp. 27-33, 2005

Akgun I., Kandakoglu A., Ozok A.F., Fuzzy Integrated Vulnerability Assessment Model for Critical Facilities in Combating the Terrorism, *Expert Systems with Applications*, vol. 37, no. 5, pp. 3561-3573, 2010

Ando N., Watanabe S., Yamaguchi T., Human Centered Architecture by Means of Q-learning Algorithm and the KEI (Knowledge, Emotion and Intention) Model, *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 189-193, 2001

Andreou A.S., Mateou N.H., Zombanakis G.A., Soft Computing for Crisis Management and Political Decision Making: The Use of Genetically Evolved Fuzzy Cognitive Maps, *Soft Computing*, vol. 9, no. 3, pp. 194-210, 2005

Axelrod R., Structure of Decision: The Cognitive Maps of Political Elites, *Princeton University Press*, 1976

Ayesh A., Emotionally Motivated Reinforcement Learning Based Controller, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 874-878, 2004

Banini G.A., Bearman R.A., Application of Fuzzy Cognitive Maps to Factors Affecting Slurry Rheology, *International Journal of Mineral Processing*, vol. 52, no. 4, pp. 233-244, 1998

Borrie D., Ozveren C.S., The Electric Power Market in the United Kingdom: Simulation with Adaptive Intelligent Agents and the use of Fuzzy Cognitive Maps as an Inference Engine, *Proceedings of the International Universities Power Engineering Conference*, vol. 2, pp. 1150-1154, 2004

Bueno S., Salmeron J.L., Fuzzy Modeling Enterprise Resource Planning Tool Selection, *Computer Standards and Interfaces*, vol. 30, no. 3, pp. 137-147, 2008

Buyukozkan G., Vardaloglu Z., An Application of Fuzzy Cognitive Map Based on Active Hebbian Learning Algorithm in Credit Risk Evaluation of Listed Companies, *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence*, pp. 89-93, 2009

Campello R.J.G.B., Amaral W.C., Towards True Linguistic Modelling through Optimal Numerical Solution, *International Journal of Systems Science,* vol. 34, no. 2, pp. 139-157, 2003

Cantu-Paz E., Parameter Setting in Parallel Genetic Algorithms, in F. Lobo, C. Lima, and Z. Michalewicz: Parameter Setting in Evolutionary Algorithms (Studies in Computational Intelligence), *Springer*, pp. 259-276, 2007

Carvalho J.P., Tome J.A.B, Qualitative Modelling of an Economic System using Rule-based Fuzzy Cognitive Maps, *Proceedings of the IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 659-664, 2004

Castelfranchi C., Falcone R., Pezzulo G., Cooperating Through a Belief-based Trust Computation, *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 263-268, 2003

Chen M.-E., Huang Y.-P., Dynamic Fuzzy Reasoning Model with Fuzzy Cognitive Map in Chinese Chess, *Proceedings of the IEEE International Conference on Neural Networks*, vol. 3, pp. 1353-1357, 1995a

Chen M.-E., Huang Y.-P., Guard Heuristic by Dynamic Fuzzy Reasoning Model for Chinese Chess, *Proceedings of International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society*, pp. 530-533,1995b

Chen S.M., Forecasting Enrollments based on Fuzzy Time Series, *Fuzzy Sets and Systems,* vol. 81, no. 3, pp.311-319, 1996

Christova N., Hadjiski M., Vachkov G., Stylios C., An Integrated Approach to Intelligent Modeling of Industrial Plants, *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, vol. 3, pp. 1527-1532, 2003

Cole J.R., Persichitte K.A., Fuzzy Cognitive Mapping: Applications in Education, *International Journal of Intelligent Systems*, vol. 15, no. 1, pp. 1-25, 2000

Cormen T.H., Leiserson C.E., and Rivest R.L., Introduction to Algorithms, *MIT Press*, 2000

D'Ambrosio B., Qualitative Process Theory Using Linguistic Variables, *Springer-Verlag,* 1989

De Kleer J., Brown J., A Qualitative Physics Based on Confluences, *Artificial Intelligence,* vol. 24, no. 1-3, pp. 7-83, 1984

De Korvin A., Simeonov P., Sirisaengtaksin O., Resource Allocation based on Imprecise Information, *Neural, Parallel and Scientific Computations,* vol. 15, no. 1, pp. 91-102, 2007

Dempster A.P., Upper and Lower Probabilities Induced by a Multivalued Mapping, *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325-339, 1967

Dickerson J.A., Kosko B., Virtual Worlds as Fuzzy Cognitive Maps, *Proceedings of the Annual International Symposium on Virtual Reality*, pp. 471-477, 1993

Dickerson J.A., Kosko B., Virtual Worlds as Fuzzy Cognitive Maps, *Presence*, vol. 3, no. 2, pp. 173-189, 1994

Espinosa-Paredes G., Nunez-Carrera A., Laureano-Cruces A.L., Vazquez-Rodriguez A., Espinosa-Martinez E.-G., Emergency Management for a Nuclear Power Plant using Fuzzy Cognitive Maps, *Annals of Nuclear Energy*, vol. 35, no. 12, pp. 2387-2396, 2008

Espinosa-Paredes G., Nunez-Carrera A., Vazquez-Rodriguez A., Espinosa-Martinez E.-G., Modeling of the High Pressure Core Spray Systems with Fuzzy Cognitive Maps for

Operational Transient Analysis in Nuclear Power Reactors, *Progress in Nuclear Energy*, vol. 51, no. 3, pp. 434-442, 2009

Feyzioglu O., Buyukozkan G., Ersoy M.S., Supply Chain Risk Analysis with Fuzzy Cognitive Maps, *Proceeding of the IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 1447-1451, 2007

Fons S., Achari G., Ross T.J., Analyses of the Environmental Impacts of an Eco-industrial Park using Fuzzy Cognitive Maps, *Proceedings of the IEEE International Conference on Industrial Informatics*, pp. 345–350, 2003-

Fons S., Achari G., Ross T., A Fuzzy Cognitive Mapping Analysis of the Impacts of an Eco-industrial Park, *Journal of Intelligent and Fuzzy Systems*, vol. 15, no. 2, pp. 75-88, 2004

Froelich W., Wakulicz-Deja A., Application of Fuzzy Cognitive Maps for Stock Market Modeling and Forecasting, *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp. 72-81, 2008

Furfaro R., Kargel J.S., Lunine J.I., Fink W., Bishop M.P., Identification of Cryovolcanism on Titan using Fuzzy Cognitive Maps, *Planetary and Space Science,* vol. 58, no. 5, pp. 761-779, 2010

Georgiou D.A., Makry D., , *Proceedings of the IEEE International Conference on Advanced Learning Technologies*, pp. 36-40, 2004

Georgopoulos V.C., Malandraki G.A., A Fuzzy Cognitive Map Hierarchical Model for Differential Diagnosis of Dysarthrias and Apraxia of Speech, *Proceedings of the IEEE Conference on Engineering in Medicine and Biology Society*, vol. 3, pp. 2409-2012, 2005.

Georgopoulos V.C., Malandraki G.A., Stylios C.D., Development of Intelligent Method for Differential Diagnosis of Specific Language Impairment, *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 4, pp. 3815-3818, 2001

Georgopoulos V.C., Malandraki G.A., Stylios C.D., A Fuzzy Cognitive Map Approach to Differential Diagnosis of Specific Language Impairment, *Artificial Intelligence in Medicine*, vol. 29, no. 3, pp. 261-278, 2003

Georgopoulos V.C., Stylios C.D., Complementary Case-based Reasoning and Competitive Fuzzy Cognitive Maps for Advanced Medical Decisions, *Soft Computing* vol. 12, no. 2, pp. 191-199, 2008

Giles B.G., Findlay C.S., Haas G., LaFrance B., Laughing W., Pembleton S, Integrating Conventional Science and Aboriginal Perspectives on Diabetes using Fuzzy Cognitive Maps, *Social Science and Medicine*, vol. 64, no. 3, pp. 562-576, 2007

Giles B.G., Haas G., Sajna M., Findlay C.S. Exploring Aboriginal views of Health using Fuzzy Cognitive Maps and Transitive Closure: A Case Study of the Determinants of Diabetes, *Canadian Journal of Public Health*, vol. 99, no. 5, pp. 411-417, 2008

Giordano R., Passarella G., Uricchio V., F., Vurro M., Fuzzy Cognitive Maps for Issue Identification in a Water Resources Conflict Resolution System, *Physics and Chemistry of the Earth*, vol. 30, no. 6-7, pp. 463-469, 2005

Glykas M., Xirogiannis G., A Soft Knowledge Modeling Approach for Geographically Dispersed Financial Organizations, *Soft Computing*, vol. 9, no. 8, pp. 579-593, 2005

Goldberg D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, *Addison-Wesley,* 1989

Golmohammadi S.K., Azadeh A., Gharehgozli A., Action Selection in Robots based on Learning Fuzzy Cognitive Map, *Proceedings of the IEEE International Conference on Industrial Informatics*, pp. 731-736, 2007

Grama A., Gupta A., Karypis G., Kumar V., Introduction to Parallel Computing, *Addison Wesley*, 2003

Grant D., Osei-Bryson K.-M., Using Fuzzy Cognitive Maps to Assess MIS Organizational Change Impact, *Proceedings of the International Conference on System Sciences*, pp. 263c-268c, 2005

Gras R., Devaurs D., Wozniak A., Aspinall A., An Individual-based Evolving Predator-prey Ecosystem Simulation using a Fuzzy Cognitive Map as the Behavior Model, *Artificial Life,* vol. 15, no. 4, pp. 423-463, 2009

Hashimoto T., Proposal of Emotion Model in Robot-assisted-activity, *Proceedings of the Annual Conference of the IEEE Industrial Electronics Society*, pp. 527–529, 2000

Hashimoto T., Yamaguchi T., Model of Knowledge, Emotion and Awareness, *Proceedings of the IEEE International Workshop on Robot and Human Communication*, pp. 326-331, 1997

Haykin S., Neural Networks: A Comprehensive Foundation, *Prentice-Hall,* 1999

Hebb D.O., The Organization of Behavior, *Wiley*, 1949

Hellendoorn H., Driankov D., Fuzzy Model Identification: Selected Approaches, *Springer-Verlag,* 1997

Herrera F., Lozano M., Verdegay J.L., Tackling Real-coded Genetic Algorithms: Operators and Tools for Behavioural Analysis, *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265-319, 1998

Hobbs B.F., Ludsin S.A., Knight R.L., Ryan P.A., Biberhofer J., Ciborowski J.J.H., Fuzzy Cognitive Mapping as a Tool to Define Management Objectives for Complex Ecosystems, *Ecological Applications*, vol. 12, no. 5, pp. 1548-1565, 2002

Holland J.H., Adaptation in Natural and Artificial Systems, *University of Michigan Press*, 1975

Hossain S., Brooks L., Fuzzy Cognitive Map Modelling Educational Software Adoption, *Computers & Education*, vol. 51, no. 4, pp. 1569–1588, 2008

Hu L., Gao J., Luo X., Research on Forming Line Features of Basic Shapes Based on Fuzzy Cognitive Map, *Proceedings of the International Conference on Information Acquisition*, pp. 392-397, 2004

Huerga A.V., A Balanced Differential Learning Algorithm in Fuzzy Cognitive Maps, *International Workshop on Qualitative Reasoning (poster)*, 2002

Hwang J.R., Chen S.M., Lee C.H., Handling Forecasting Problems using Fuzzy Time Series, *Fuzzy Sets and Systems,* vol. 100, no. 1-3, pp.217-228, 1998

Innocent P.R., John R.I., Computer Aided Fuzzy Medical Diagnosis, *Information Sciences*, vol. 162, no. 2, pp. 81-104, 2004

Jetter A.J.M., Educating the Guess: Strategies, Concepts and Tools for the Fuzzy Front End of Product Development, *Proceedings of the International Conference on Management of Engineering and Technology*, pp. 261–273, 2003

Jetter A.J., Fuzzy Cognitive Maps for Engineering and Technology Management: What Works in Practice?, *Proceedings of the Technology Management for the Global Future Conference*, vol. 2, pp. 498-512, 2006

John R.I., Innocent P.R., Modeling Uncertainty in Clinical Diagnosis Using Fuzzy Logic, *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 35, no. 6, pp. 1340-1350, 2005

Kardaras D., Karakostas B., Use of Fuzzy Cognitive Maps to Simulate the Information Systems Strategic Planning Process, *Information and Software Technology*, vol. 41, no. 4, pp. 197-210, 1999

Khan M., Quaddus M., Group Decision Support using Fuzzy Cognitive Maps for Causal Reasoning, *Group Decision and Negotiation Journal*, vol. 13, no. 5, pp. 463-480, 2004

Khan M.S., Chong A., Fuzzy Cognitive Map Analysis with Genetic Algorithm, *Proceedings of the Indian International Conference on Artificial Intelligence,* pp. 1196-1205, 2003

Kok K., The Potential of Fuzzy Cognitive Maps for Semi-quantitative Scenario Development, with an Example from Brazil, *Global Environmental Change,* vol. 19, no. 1, pp. 122-133, 2009

Konfrst Z., Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives, *Proceedings of the Parallel and Distributed Processing Symposium*, pp. 162-166, 2004

Kosko B., Fuzzy Cognitive Maps, *International Journal of Man-Machine Studies*, vol. 24, pp. 65-75, 1986

Kosko B., Hidden Patterns in Combined and Adaptive Knowledge Networks, *International Journal of Approximate Reasoning,* vol. 2, pp. 377-393, 1988

Kosko B., Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence, *Prentice-Hall,* 1992

Kosko B., Fuzzy Engineering, *Prentice- Hall*, 1997

Koulouriotis D.E., Diakoulakis I.E., Emiris D.M., A Fuzzy Cognitive Map-based Stock Market Model: Synthesis, Analysis and Experimental Results, *Proceedings of the IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 465-468, 2001a

Koulouriotis D.E., Diakoulakis I.E., Emiris D.M., Learning Fuzzy Cognitive Maps using Evolution Strategies: a Novel Schema for Modeling and Simulating High-Level Behaviour, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 364-371, 2001b

Kurgan L., Stach W., Ruan J., Using Genetic Algorithms and Fuzzy Cognitive Maps to Analyze Hydrophobicity Scales and Indices for Prediction of Protein Secondary Structure Content, *Journal of Theoretical Biology,* vol. 248, no. 2, pp. 354-366, 2007

Lee K., Kim S., Sakawa M., On-line Fault Diagnosis by Using Fuzzy Cognitive Map, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E79-A, no. 6, pp. 921-927, 1996

Lee K.C., Kim J.S., Chung N.H., Kwon S.J., Fuzzy Cognitive Map Approach to Web-mining Inference Amplification, *Expert Systems with Applications*, vol. 22, no. 3, pp. 197-211, 2002

Lee S., Kim B.G., Lee K., Fuzzy Cognitive Map-based Approach to Evaluate EDI Performance: A Test of Causal Model, *Expert Systems with Applications*, vol. 27, no. 2, pp. 287-299, 2004

Lee S., Han I., Fuzzy Cognitive Map for the Design of EDI Controls, *Information and Management*, vol. 37, no. 1, pp. 37-50, 2000

Lee K.C., Lee S., Causal Knowledge-based Design of EDI Controls: an Explorative Study, *Computers in Human Behavior,* vol. 23, no. 1, pp. 628-663, 2007

Li X., Ji H., Zheng R., Li Y., Yu F.R., A Novel Team-centric Peer Selection Scheme for Distributed Wireless P2P Networks, *Proceeding of the IEEE Wireless Communications and Networking Conference*, pp. 2938-2942, 2009

Lin C.-M., Combination Study of Fuzzy Cognitive Map, *International Journal of Energy and Environment*, vol. 1, no. 2, pp. 65-69, 2007

Lin C.-M., Combination Study of Multi Fuzzy Cognitive Map, *Proceedings of the International Conference on Knowledge-Based and Intelligent Information & Engineering Systems,* pp. 332–339, 2008

Liu Z.-Q., and Satur R., Contextual Fuzzy Cognitive Map for Decision Support in Geographic Information Systems, *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 5, pp. 495-507, 1999

Macedo J., Generative System for Re-engineering Manufacturing System Organization, *International Journal of Production Research*, vol. 37, no. 12, pp. 2639-2664, 1999

Mu C.-P., Huang H.-K., Tian S.-F., Fuzzy Cognitive Maps for Decision Support in an Automatic Intrusion Response Mechanism, *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1789-1794, 2004

Ndousse T.D., Okuda T, Computational Intelligence for Distributed Fault Management in Networks Using Fuzzy Cognitive Maps, Communications, *Proceedings of the IEEE International Conference on Conference Record, Converging Technologies for Tomorrow's Applications*, vol. 3, pp. 1558-1562, 1996

Nguyen C., Mannino M., Gardiner K., Cio K.J., ClusFCM: An Algorithm for Predicting Protein Function using Homologies and Protein Interactions, *Journal of Bioinformatics and Computational Biology*, vol. 6, no. 1, pp. 203-222, 2008

Niskanen V.A., Application of Fuzzy Cognitive Maps to Business Planning Models, *Advances in Soft Computing*, vol. 40, pp. 119-127, 2007

Noori S., Amiri R.H., Bourouni A., An FCM Approach to Better Understanding of Conflicts: a Case of New Technology Development, *International Journal of Business and Management*, vol. 4, no. 3, pp. 106-115, 2009

Oja E., Ogawa H., Wangviwattam J., Learning in Nonlinear Constrained Hebbian Networks. In: Kohonen T, Makisara K, Simula O, Kangas J, Artificial Neural Networks, *Elsevier*, 1991

Ozesmi U., Ozesmi S., A Participatory Approach to Ecosystem Conservation: Fuzzy Cognitive Maps and Stakeholder Group Analysis in Uluabat Lake, Turkey, *Environmental Management*, vol. 31, no. 4, pp. 518-31, 2003

Ozesmi U., Ozesmi S., Ecological Models Based on People's Knowledge: A Multi-step Fuzzy Cognitive Mapping Approach, *Ecological Modelling*, vol. 176, no. 1-2, pp. 43-64, 2004

Pajares G., De la Cruz J.M., Fuzzy Cognitive Maps for Stereovision Matching, *Pattern Recognition*, vol. 39, no. 11, pp. 2101-2114, 2006

Pan X., Duan G., Xiang D., Mou P., Intelligent Disassembly Sequence Planning for EOL Recycling Based on Hierarchical Fuzzy Cognitive Map, *Proceedings of the IEEE International Symposium on Electronics and the Environment*, pp. 255–259, 2005

Papageorgiou E.I., Groumpos P.P., A New Hybrid Method using Evolutionary Algorithms to Train Fuzzy Cognitive Maps, *Applied Soft Computing*, vol. 5, pp. 409-431, 2004

Papageorgiou E., Stylios C.D., Groumpos P.P., Decision Making in External Beam Radiation Therapy Based on Fuzzy Cognitive Maps, *Proceedings of the First International IEEE Symposium on Intelligent Systems*, vol. 1, pp. 320-325, 2002

Papageorgiou E.I., Stylios C.D., Groumpos P.P., An Integrated Two-level Hierarchical System for Decision Making in Radiation Therapy Based on Fuzzy Cognitive Maps, *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 12, pp. 1326-1339, 2003a

Papageorgiou E., Stylios C.D., Groumpos P.P., Fuzzy Cognitive Map Learning Based on Nonlinear Hebbian Rule, *Proceedings of the Australian Conference on Artificial Intelligence*, pp. 256-268, 2003b

Papageorgiou E., Stylios C.D., Groumpos P.P., Active Hebbian Learning Algorithm to Train Fuzzy Cognitive Maps, *International Journal of Approximate Reasoning*, vol. 37, no. 3, pp. 219-249, 2004

Papageorgiou E.I., Spyridonos P.P., Stylios C.D., Advanced Soft Computing Diagnosis Method for Tumour Grading, *Artificial Intelligence in Medicine*, vol. 36, no. 1, pp. 59-70, 2006

Papageorgiou E., Stylios C., Groumpos P., Novel Architecture for Supporting Medical Decision Making of Different Data Types based on Fuzzy Cognitive Map Framework, *Proceedings of the IEEE Annual International Conference in Medicine and Biology*, pp. 1192-1195, 2007

Papageorgiou E.I., Spyridonos P.P., Glotsos D.T., Stylios C.D., Ravazoula P., Nikiforidis G.N., Groumpos P.P., Brain Tumor Characterization using the Soft Computing Technique of Fuzzy Cognitive Maps, *Applied Soft Computing Journal,* vol. 8, pp. 1, pp. 820-828, 2008a

Papageorgiou E.I., Papandrianos N.I., Apostolopoulos D.J., Vassilakos P.J., Fuzzy Cognitive Map based Decision Support System for Thyroid Diagnosis Management, *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 1204-1211, 2008b

Papageorgiou E.I., Papandrianos N., Karagianni G., Kyriazopoulos G., Sfyras D.Fuzzy Cognitive Map Based Approach for Assessing Pulmonary Infections, *Proceedings of the 18th International Symposium on Foundations of Intelligent Systems*, pp. 109-118, 2009

Papakostas G.A., Boutalis Y.S., Koulouriotis D.E., Mertzios, Fuzzy Cognitive Maps for Pattern Recognition Applications, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 22, no. 8, pp. 1461-1486, 2008

Parsopoulos K.E., Papageorgiou E.I., Groumpos P.P., Vrahatis M.N., A First Study of Fuzzy Cognitive Maps Learning Using Particle Swarm Optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1440-1447, 2003

Patnaik L.M., Mandavilli S., Adaptation in Genetic Algorithms. In: Sankar K.P., Wang P.P., Genetic Algorithms for Pattern Recognition, *CRC Press,* pp. 45-64, 1996

Pedrycz W., Distributed Fuzzy System Modeling, *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 25, no. 5, pp. 769-780, 1995

Pedrycz W., Fuzzy Modeling: Paradigms and Practice, *Kluwer Academic Publishers,* 1996

Pelaez C.E., Bowles J.B., Applying Fuzzy Cognitive-Maps Knowledge-Representation to Failure Modes Effects Analysis, *Proceedings of the Symposium on Reliability and Maintainability*, pp. 450-456, 1995

Pelaez C.E., Bowles J.B., Using Fuzzy Cognitive Maps as a System Model for Failure Modes and Effects Analysis, *Information Sciences*, vol. 88, no.1-4, pp. 177-199, 1996

Peng Z., Yang B., Fang W., A Learning Algorithm of Fuzzy Cognitive Map in Document Classification, *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 1, pp. 501-504, 2008

Perusich K., Fuzzy Cognitive Maps for Policy Analysis, *Proceedings of the International Symposium on Technology and Society Technical Expertise and Public Decisions*, pp. 369-373, 1996

Perusich K., McNeese M.D., Using Fuzzy Cognitive Maps as an Intelligent Analyst, *Proceedings of the IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety*, pp. 9-15, 2005

Perusich K., Qalitatively Troubleshooting Electronic Circuits using Fuzzy Cognitive Maps, *Proceedings of the IEEE International Conference on Electro/Information Technology*, pp. 327-332, 2007

Perusich K., Using Fuzzy Cognitive Maps to Identify Multiple Causes in Troubleshooting Systems, *Integrated Computer-Aided Engineering,* vol. 15, no. 2, pp. 197-206, 2008

Petalas Y.G., Parsopoulos K.E. Vrahatis M.N., Improving Fuzzy Cognitive Maps Learning Through Memetic Particle Swarm Optimization, *Soft Computing*, vol. 13, no. 1, pp. 77-94, 2009

Pipe A.G., Jin Y., Fogarty T.C.; Winfield A., Abstracting Non-situated Behaviours from Situated Experiences: an Experiment in Mobile Robotics, *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 447-452, 1995

Rajaram T., Das A., Modeling of Interactions among Sustainability Components of an Agro-ecosystem using Local Knowledge through Cognitive Mapping and Fuzzy Inference System, *Expert Systems with Applications,* vol. 37, no. 2, pp. 1734-1744, 2010

Rodriguez-Repiso L., Setchi R., Salmeron J.L., Modelling IT Projects Success with Fuzzy Cognitive Maps, *Expert Systems with Applications*, vol. 32, no. 2, pp. 543-559, 2007

Saaty T.L., The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation, *McGraw-Hill*, 1980

Sadiq R., Kleiner Y., Rajani B., Tesfamariam S., A Novel Modelling Approach to Predict Risk of Water Quality Failures in Deteriorating Water Mains, *Proceedings of the Combined International Conference of Computing and Control for the Water Industry, and Sustainable Urban Water Management*, pp. 223-228, 2007

Salmeron J.L., Augmented Fuzzy Cognitive Maps for Modelling LMS Critical Success Factors, *Knowledge-Based Systems*, vol. 22, no. 4, pp. 275-278, 2009

Satish Jamadagni N.S., Dealing with Location Uncertainty in Mobile Networks Using Contextual Fuzzy Cognitive Maps as Spatial Decision Support Systems, *Proceedings of the Vehicular Technology Conference*, pp. 1489-1492, 2000

Satur R., Liu Z.-Q., Contextual Fuzzy Cognitive Maps for Geographic Information Systems, *Proceedings of the IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 1165-1169, 1999a

Satur R., Liu Z.-Q., Contextual Fuzzy Cognitive Map Framework for Geographic Information Systems, *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 5, pp.481-494, 1999b

Schlager C., Pernul G., Trust Modelling in e-commerce through Fuzzy Cognitive Maps, *Proceedings of the International Conference on Availability, Security, and Reliability*, pp. 344-351, 2008

Shafer G., A Mathematical Theory of Evidence, *Princeton University Press*, 1976

Siraj A., Bridges S.M., Vaughn R.B., Fuzzy Cognitive Maps for Decision Support in an Intelligent Intrusion Detection System, *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society*, vol. 4, pp. 2165-2170, 2001

Song Q., Chissom B.S., Forecasting Enrollments with Fuzzy Time Series – Part I, *Fuzzy Sets and Systems*, vol. 51, no. 1, pp.1-9, 1993

Stach W., Kurgan L., Modeling Software Development Projects Using Fuzzy Cognitive Maps, *Proceedings of the ASERC Workshop on Quantitative and Soft Software Engineering*, pp.55-60, 2004

Stach W., Kurgan L., Pedrycz W., Reformat M., Parallel Fuzzy Cognitive Maps as a Tool for Modeling Software Development Projects, *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society*, pp.28-33, 2004a

Stach W., Kurgan L., Pedrycz W., Reformat M., Learning Fuzzy Cognitive Maps with Required Precision Using Genetic Algorithm Approach, *Electronics Letters*, vol. 40, no. 24, pp. 1519-1520, 2004b

Stach W., Kurgan L.A., Pedrycz W., A Survey of Fuzzy Cognitive Map Learning Methods, In: Grzegorzewski, P., Krawczak, M., Zadrozny, S., (Eds.), Issues in Soft Computing: Theory and Applications, pp. 71-84, *Exit*, 2005a

Stach W., Kurgan L., Pedrycz W., Reformat M., Evolutionary Development of Fuzzy Cognitive Maps, *Proceedings of the 14th International Conference on Fuzzy Systems*, pp. 619-624, 2005b

Stach W., Kurgan L., Pedrycz W., Reformat M., Genetic Learning of Fuzzy Cognitive Maps, *Fuzzy Sets and Systems*, vol. 153, no. 3, pp. 371-401, 2005c

Stach W., Kurgan L.A., Pedrycz W., Parallel Learning of Large Fuzzy Cognitive Maps, *Proceedings of the International Joint Conference on Neural Networks*, pp.1584-1589, 2007

Stach W., Kurgan L., Pedrycz W., Numerical and Linguistic Prediction of Time-Series with the use of Fuzzy Cognitive Maps, *IEEE Transactions on Fuzzy Systems,* vol. 16, no. 1, pp. 61-72, 2008a

Stach W., Kurgan L.A., Pedrycz W., Data-Driven Nonlinear Hebbian Learning Method for Fuzzy Cognitive Maps, *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp.1975-1981, 2008b

Stach W., Kurgan L.A., Pedrycz W., Expert-based and Computational Methods for Developing Fuzzy Cognitive Maps, In: Glykas, M., Fuzzy Cognitive Maps: Advances in Theory, Methodologies, Tools and Applications, *Springer*, 2010 (to appear in July)

Styblinski M.A., Meyer B.D, Fuzzy Cognitive Maps, Signal Flow Graphs, and Qualitative Circuit Analysis, *Proceedings of the IEEE International Conference on Neural Networks*, vol. 2, pp. 549-556, 1988

Stylios C.D., Georgopoulos V.C., Fuzzy Cognitive Maps Structure for Medical Decision Support Systems, *Studies in Fuzziness and Soft Computing,* vol. 218, pp. 151-174, 2008

Stylios C.D., Groumpos P.P., Fuzzy Cognitive Maps: A Model for Intelligent Supervisory Control Systems, *Computers in Industry*, vol. 39, no. 3, pp. 229-238, 1999

Stylios C.D., Groumpos P.P., Fuzzy Cognitive Maps in Modeling Supervisory Control Systems, *Journal of Intelligent and Fuzzy Systems*, vol. 8, no. 2, pp. 83-98, 2000

Stylios C.D., Groumpos P.P., Modeling Complex Systems Using Fuzzy Cognitive Map*, IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 34, no. 1, pp. 155-162, 2004

Stylios C.D., Georgopoulos V.C., Groumpos P.P., Introducing the Theory of Fuzzy Cognitive Maps in Distributed Systems, *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 55-60, 1997

Stylios C.D., Georgoulas G., Groumpos P.P., The Challenge of Using Soft Computing for Decision Support During Labour *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 4, pp. 3835-3838, 2001

Subramanian H., Dagli C.H., Cooperative Cleaning for Distributed Autonomous Robot Systems Using Fuzzy Cognitive Maps, *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society*, pp. 287-290, 2003

Sullivan J., Woodall W.H., Comparison of Fuzzy Forecasting and Markov Modeling, *Fuzzy Sets and Systems,* vol. 64, no. 3, pp.279-293, 1994

Tan C.O., Ozesmi U.A., A Generic Shallow Lake Ecosystem Model Based on Collective Expert Knowledge, *Hydrobiologia*, vol. 563, no. 1, pp. 125-142, 2006

Tsadiras A.K., Kouskouvelis I., Margaritis K.G., Using Fuzzy Cognitive Maps as a Decision Support System for Political Decisions, *Proceedings of the Panhellenic Conference on Informatics,* pp. 172-182, 2001

Tsadiras A.K., Using Fuzzy Cognitive Maps for E-Commerce Strategic Planning, *Proceedings of the Panhellenic Conference on Informatics*, Greece, pp. 142-151, 2003

Tsadiras A.K., Comparing the Inference Capabilities of Binary, Trivalent and Sigmoid Fuzzy Cognitive Maps, *Information Sciences*, vol. 178, no. 20, pp. 3880-3894, 2008

Xavier C., Iyengar S.S., Introduction to Parallel Algorithms, *Wiley–Interscience*, 1998

Xin J., Dickerson J.E., Dickerson J.A., Fuzzy Feature Extraction and Visualization for Intrusion Detection, *Proceedings of the IEEE Conference on Fuzzy Systems*, vol. 2, pp. 1249-1254, 2003

Xirogiannis G., Glykas M., Fuzzy Cognitive Maps in Business Analysis and Performance-driven Change*, IEEE Transactions on Engineering Management*, vol. 51, no. 3, pp. 334-351, 2004

Xirogiannis G., Stylianidis O., Stefanou J., Intelligent Decision Support of Urban Design, *Journal of Architectural and Planning Research*, vol. 23, no. 2, pp. 113-133, 2006

Xirogiannis G., Glykas M., Intelligent Modeling of e-business Maturity, *Expert Systems with Applications*, vol. 32, no. 2, pp. 687-702, 2007

Yager R.R., Filev D.P., Essentials of Fuzzy Modeling and Control, *Wiley-Interscience*, 1994

Yaman D., Polat S., A Fuzzy Cognitive Map Approach for Effect-based Operations: An Illustrative Case, *Information Sciences*, vol. 179, no. 4, pp. 382-403, 2009

Yeap W.K., Wong C.K., Schmidt J., Using a Mobile Robot to Test a Theory of Cognitive Mapping, *Springer Tracts in Advanced Robotics,* vol. 38, pp. 281-295, 2008

Zhou X., Zhang H., An Algorithm of Text Categorization based on Similar Rough Set and Fuzzy Cognitive Map, *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 3, pp.127-131, 2008

Appendix A

Abbreviations

AHL – Active Hebbian Learning

BDA – Balanced Differential Hebbian Algorithm

D&C – Divide and Conquer

DD-NHL – Data-driven Nonlinear Hebbian Learning

DHL – Differential Hebbian Learning

FCM – Fuzzy Cognitive Map

GA – Genetic Algorithm

GS – Genetic Strategy

MPSO – Memetic Particle Swarm Optimization

NHL – Nonlinear Hebbian Learning

PSO – Particle Swarm Optimization

RCGA – Real-Coded Genetic Algorithm

Appendix B

List of publications

This appendix provides chapter-wide lists of relevant contributions, including published in-print and submitted works, that were used to compile this dissertation.

Chapter 2

- **Overview of methods for the development of FCMs**

Stach W., Kurgan L.A., Pedrycz W., Expert-based and Computational Methods for Developing Fuzzy Cognitive Maps, In: Glykas, M., Fuzzy Cognitive Maps: Advances in Theory, Methodologies, Tools and Applications, *Springer*, 2010 (to appear in July)

Stach W., Kurgan L.A., Pedrycz W., A Survey of Fuzzy Cognitive Map Learning Methods, In: Grzegorzewski, P., Krawczak, M., Zadrozny, S., (Eds.), Issues in Soft Computing: Theory and Applications, pp. 71-84, *Exit*, 2005

Chapter 3

- **RCGA-based method for learning FCMs**

Stach W., Kurgan L., Pedrycz W., and Reformat M., Genetic Learning of Fuzzy Cognitive Maps, *Fuzzy Sets and Systems*, vol. 153, no.3, pp.371-401, 2005

Stach W., Kurgan L., Pedrycz W., Reformat M, Evolutionary Development of Fuzzy Cognitive Maps, *Proceedings of the International Conference on Fuzzy Systems (FUZZ-IEEE 2005)*, pp.619-624, 2005

Stach W., Kurgan L., Pedrycz W., and Reformat M., Learning Fuzzy Cognitive Maps with Required Precision Using Genetic Algorithm Approach, *Electronics Letters*, vol.40, no.24, pp.1519-1520, 2004

- **Parallel RCGA methods for learning FCMs**

Stach W., Kurgan L.A., Pedrycz W., Parallel Learning of Large Fuzzy Cognitive Maps, *Proceedings of the International Joint Conference on Neural Networks*, pp.1584-1589, 2007

- **Divide and conquer RCGA method for learning FCMs**

Stach W., Kurgan L.A., and Pedrycz W., Divide and Conquer Method for Learning Large Fuzzy Cognitive Maps, *Fuzzy Sets and Systems*, 2010, (accepted)

Stach W., Kurgan L., Pedrycz W., A Framework for a Novel Scalable FCM Learning Method, *Proceedings of the Symposium on Human-Centric Computing and Data Processing*, pp.13-14, 2007

- **Sparse RCGA method for learning FCMs**

Stach W., Pedrycz W., and Kurgan L.A., Learning of Sparse Cognitive Maps using Density Estimate, 2010 (submitted)

Chapter 4

- **Improved method for aggregation of FCMs**

Stach W., Kurgan L.A., Pedrycz W., Improved Method for Aggregating Fuzzy Cognitive Maps, 2010 (submitted)