A computational tool for simulation and learning of Fuzzy Cognitive Maps

Gonzalo Nápoles, Isel Grau, Rafael Bello
Department of Computer Sciences
Central University of Las Villas
Santa Clara, Cuba

Maikel León, Koen Vahoof
Department of Business Informatics
Hasselt University
Diepenbeek, Belgium

Elpiniki Papageorgiou
Department of Computer Engineering
Technological Education Institute of Central Greece
Lamia, Greece

***Abstract*—During the last decade Fuzzy Cognitive Maps (FCM) have become a useful tool for solving unstructured problems. In a few words they could be defined as Recurrent Neural Networks for simulating complex systems, where neurons denote concepts, objects or entities of the investigated system. Normally FCM are entirely designed using the best knowledge of a group of experts in a given domain, so frequently learning algorithms for tuning the model parameters are required. Despite the theoretical advances in such fields, the lack of a suitable computational framework for handling FCM-based systems is still an open problem. This paper introduces a novel tool for designing and simulating FCM which gathers several learning algorithms for adjusting the introduced parameters. More specifically, the framework includes supervised and unsupervised learning algorithms for computing the causal weights, algorithms for optimizing the network topology in large FCM (without losing significant information) and also methods for improving the global convergence on continuous FCM. It should be stated that these algorithms are oriented to prediction tasks, but they could be easily extended to other fields.**

Index Terms — ***FCM, prediction problems, modeling, learning algorithms, computational framework.***

I. Introduction

Fuzzy Cognitive Maps (FCM) were introduced by B. Kosko as a knowledge-based methodology for modeling and simulating dynamic systems [1]. They are in fact recurrent neural networks that incorporate elements of fuzzy logic, which are considered during the knowledge-engineering phase [2]. However, there are some differences regarding neural networks (e.g. usually hidden neurons are not allowed). Using this methodology, a system can be modeled in terms of concepts (e.g. variables, objects or states which are mapped as neurons in connectionist approaches) and causal relations among such entities.

From the structural point of view, a FCM may be represented by directed graphs allowing feedback, consisting on neurons and signed weighted arcs [3]. The sign associated to each connection involves a relevant meaning for the system. For example, if the sign is positive, then an increase or decrease on the first concept causes the effect neuron to change in the same direction, whereas in the case of negative causal relations, the change on the effect variable will be in the opposite direction.

Recently the FCM theory has gained a lot of attention among researchers. For instance, León and collaborators [4] proposed a FCM for studying travel behavior in modern societies allowing policy-makers a better understanding about such difficult issues. Similarly, Nápoles et al. [5] introduced a FCM-based model for analyzing the resistance mechanism of HIV proteins. This model comprises relevant biological knowledge that could be used by drug-designers when designing new therapies. As well, the FCM theory has been widely used in other fields including: decision making, risk analysis, engineering, system control, game theory, management, telecommunications, medicine, business requests, among other domains. For further information the reader could consult the survey published by Papageorgiou and Salmerón [6] about relevant applications of this methodology.

On the other hand, the pioneer research work of Papakostas at al. [7] opened a new research direction: the solution of pattern classification problems. Despite these notable advances, the lack of a suitable computational framework for handling FCM-based systems is still an open problem. The scientific literature shows some software products developed with the intention of drawing FCM by users with no expertise in mathematics or computer science, as the FCM Modeler [8] and the FCM Designer [9]. The first one is a simple development to support group decision making on a qualitative static model, while the second one is a better implementation, but still hard to interact with and it does not have experimental facilities. More recently, León et al. [10] proposed the FCM TOOL. It has a nice Graphical User Interface and also includes a learning algorithm for estimating the causal weights. However, this software was conceived for facing the public transportation problem mentioned before.

In this paper we propose a novel computational framework, for designing and learning and simulating FCM-based systems. This software is an extension of the FCM TOOL in the sense that we preserved the main options for designing new systems, however, simulation options and learning methods were entirely modified. Actually, we implemented an internal design oriented to pattern classification problems where the 80% of the source was modified. In next sections we describe the software options and the mathematical formulation of learning algorithms, which will be evaluated using a real case study.

II. Fuzzy Cognitive Maps

Mathematically speaking, a FCM can be defined using a 4-tuple (C, W, A, f) where $C = \{C_1, C_2, \dots, C_M\}$ is a set of M graph nodes, $W: (C_i, C_j) \rightarrow w_{ij}$ is a function which associates a causal value $w_{ij} \in [-1,1]$ to each pair of nodes (C_i, C_j). It denotes the weight of the directed edge from C_i to C_j. Notice that the matrix $W_{M\times M}$ gathers the system causality which could be estimated by experts or automatically computed from historical data. Equally, $A: (C_i) \rightarrow A_i$ associates an activation value $A_i \in \mathbb{R}$ to each node C_i at each time t $(t = 1,2, \dots, T)$. To conclude, a transformation function $f: \mathbb{R} \rightarrow [0,1]$ or $f: \mathbb{R} \rightarrow [-1,1]$ is used to preserve the activation value of neurons in the desired range.

Next equation formalizes the rule for updating the activation vector (i.e. the activation value of neurons) using the state vector A^0 as the initial configuration. In the same way to other recurrent models (e.g. the well-known Hopfield network) this information propagation rule is iteratively repeated until a hidden pattern is observed (which is an ideal outcome), or a maximal number of cycles T is reached [11] which means that the map was unable to converge towards a fixed-point attractor.

$$A_i^{t+1} = f\left(\sum_{j=1}^{M} w_{ji}\, A_j^t + w_{ii} A_i^t\right), i \neq j \qquad (1)$$

In this rule the most commonly used threshold functions are: the bivalent function, the trivalent function, and also the sigmoid variants [12]. The effect on selecting a function over the stability and inference of FCM was explored by Tsadiras [13]. From this work the following remarks are summarized:

- Discrete FCM never show chaotic behavior. It means that always a fixed-point attractor or a limit cycle will be observed. More explicitly:
  - Fixed-point attractor: the system will produce the same output after the time t_k.
  - Limit cycle: the same output or state vector will be regularly observed with period P.
- Continuous FCM may exhibit chaotic states, where the model continues producing different state vectors for successive iterations. In such situations the map cannot stabilize, leading to confusing outcomes.

Usually, a FCM represents the knowledge extracted from an expert in a given domain. However, it is possible to have better consistency in the drawing of a FCM if more than one expert are used in the system modeling [14]. The aggregation of multiple experts into a single structure allows improving the reliability of the final model by merging incomplete opinions from different knowledge sources. This process makes the map less susceptible to the effects of ignorance and erroneous beliefs, during the knowledge representation and conceptualization.

The possibility of aggregating numerous FCM into a single knowledge structure is an important advantage over other well-known knowledge-based models such as Bayesian Networks or Petri Nets. Being more explicit, it is well known that the drawing process of Petri Nets is extremely hard by non-expert users since it requires a mathematical background.

III. Features of the proposed framework

In this section we introduce a computational framework for handling FCM-based systems. This software comprises 20.000 source code lines (completely written in Java language) which are distributed in 115 source files. These archives are organized in 4 global packages (*map*, *algorithms*, *interface* and *resources*) and several sub-packages. Here the most relevant packages are *algorithms* and *interface*: the first one contains several learning methods for adjusting the map parameters, whereas the second one includes the main graphical components.

Before providing a deeper description of the central options and functionalities of the proposed framework, we must remark that the software was designed for solving prediction tasks with a single decision concept (i.e. a partition of the activation space defines various decision classes). These scenarios are common when addressing pattern classification problems. The reader can find illustrative examples in the literature [15] [16] [17]. It does not imply that the FCM TOOL cannot be used for facing other scenarios (e.g. modeling problems where only a single example is available and therefore supervised learning algorithms do not fit very well). However, most options of the software will remain inactive when solving such kinds of problems.

As a rough picture, our framework involves three groups of functionalities that are distributed in 6 menus (*File*, *Edit*, *Build*, *Run*, *Reset* and *Help*). The first group is oriented to the design of new FCM-based systems, where an expert in a given domain could completely design and simulate a system, without having previous knowledge in mathematics or computer science; while the second group involves learning algorithms for adjusting the introduced parameters (e.g. the direction and intensity of causal relations). To conclude, the third group includes procedures for evaluating the system behavior (e.g. stability and convergence) which generally requires more expertise.

Next we describe basic commands that allow creating and handling new components such as concepts and relations. They belong to the first group of functionalities:

- ***New concept***. It creates a concept with a fixed name that can be renamed later (by clicking the right button over the concept and next changing the field "*name*").
- ***New relation***. It creates a new causal relation with a fixed value. This action is done by clicking the left button over the first concept and next draw a straight line from the cause neuron to the effect node. If there is a connection between such nodes, the line will be changed by a curve. The expert could also modify the initial causal value by clicking the right button over the causal relation and next changing the field "*causal value*".
- ***Select component***. It allows selecting (or moving) a map component (e.g. this action is mandatory when changing concepts' properties such as the initial value).
- ***Delete component***. This action deletes a component (i.e. causal relations and concepts). If the selected component is a concept, then the connected causal relations will be automatically deleted as well. Besides, when removing nodes be aware of deleting the decision concept! If this happens, learning procedures cannot be used.

As mentioned before, the second group of options includes learning algorithms for improving the initial modeling provided by experts. They are oriented to three directions:

- Supervised and unsupervised methods for estimating the causal weights. The unsupervised procedures are based on the Hebbian rule, whereas the supervised algorithms are focused on prediction problems.
- Learning procedures for optimizing the system topology without losing significant information. They are entirely focused on solving prediction problems.
- Learning algorithms for improving the convergence on sigmoid FCM without affecting the prediction capability of the original FCM-based system.

These procedures require a low user intervention, since they involve a few parameters which were extensively studied in the original papers. In the next section we review the mathematical formulation behind such learning algorithms.

Finally, the third group of functionalities is oriented to the system exploitation and interpretation. For example:

- ***Classify new patterns***. This function allows classifying a new instance or simply simulating a state vector. It could be performed by directly modifying the activation initial values of input neurons. Moreover, this option provides a diagram with the activation values of map neurons during each step of the inference process.
- ***Prediction report***. This function is only available if the map has a decision concept. It computes several statistics (e.g. accuracy, sensitivity, F-measure). In order to trigger this command the expert must specify the path of the base with the input instances to be evaluated.
- ***Convergence plotter***. This option provides an interface with the activation values of the decision concept during the execution method, for each input instance. In order to trigger this command the user must specify the path of the base with the input instances to be evaluated.
- ***Execution parameters***. By using this function the expert can manage the map parameters such as the activation (or normalization) function, which is used to keep the values of concepts in the desired range, or the maximal number of iterations allowed during the inference.

Similarly, the FCM TOOL provides more functionalities that could not be detailed in this paper. For example, the expert could also modify further graphical aspects when designing a new map such as the color of concepts or the position of labels. Even it is possible to execute the inference step in a mode where the size of each concept is proportional to its activation. Of course, *undo* and *redo* buttons were also implemented.

The framework also incorporates methods for automatically building large FCM-based systems, and a standard procedure for combining multiple FCM into a single one. However, we believe that further work in this direction is required since regularly the prediction capability of the final system decreases regarding the global performance of individual sub-systems.

IV. Learning algorithms

In this section we briefly summarize the mathematical basis of the learning methods implemented in the proposed tool. Such algorithms are essentially optimization procedures that could be addressed using exact or heuristic approaches. We prefer to use approximate methods since population-based metaheuristics are able of finding near-optimal solutions in a reasonable execution time, ignoring analytical properties of the target function (e.g. continuity, differentiability, convexity or gradient information) which are regularly unknown in advanced. Moreover, learning methods are quite complex, so the time required for computing a reasonable solution could be excessive, even for modern CPU. However, exact algorithms such as mathematical programming techniques could be adopted as well.

A. Estimating the causal weights

The first learning problem is maybe the most important since it defines the system behavior. It consists on estimating a set of weights minimizing the differences between the expected output and the predicted outcome. From the optimization point of view each dimension of the solution comprises a causal value, that is, the value between two concepts. As a result, the total number of dimensions will be in correspondence with the number of causal connections. The next equation describes the objective function codified into the FCM TOOL for facing this problem, which was proposed by Nápoles et al. [5]. It should be stated that we assume a FCM-based system with single decision concept.

$$F(x,\phi) = \sum_{i=1}^{|\phi|} |I(x,\phi_i) - R(\phi_i)| \qquad (2)$$

In this formulation ϕ represents the input set (training cases), while $I(.)$ is a binary function that calculates the map inference for the ith instance, using the solution x as causal weight matrix. Besides, another binary function $R(.)$ is used for computing the map response for the current train-case ϕ_i. Hence, the objective function will achieve its optimal value when the FCM inference process and historical data are identical.

For solving this continuous optimization task the framework FCM TOOL uses 12 potent population-based optimizers, which includes Particle Swarm Optimization [18], Real-Coded Genetic Algorithms [19] and Differential Evolution [20]. Likewise, the tool includes 4 Hebbian-type learning methods: Active Hebbian Learning [21], Differential Hebbian Learning [22], Non-linear Hebbian Learning [23] and the Balanced Differential Algorithm proposed by Huerga [24]. We suggest using heuristic algorithms when facing pattern classification problems, since in such cases Hebbian-type methods report poor performance.

B. Optimizing the network topology

As mentioned, during the construction of a new FCM-based system, experts determine map concepts and causal connections among such entities. In this scenario is unlikely to find a "weak concept", that is to say, a graph node that can be surely removed without affecting the system performance. However, sometimes FCM-based systems are automatically computed from data and therefore some attributes could be superfluous or contradictory, negatively affecting the system performance.

Here the optimization problem consist on finding a minimal subset of concepts (i.e. a minimal sub-system) having the same performance regarding the original system, without affecting the overall performance. Since we are mainly focused on maps with prediction capabilities, the map performance could be measured as the relative number of patterns recognized by the system (i.e. the system accuracy). The following equation (3) formalizes the objective function to be minimized during optimization phase, where ϕ is the training set, ψ_Y is the number of non-relevant concepts, regarding the total number of variables involved in the modeling, whereas $0 < \lambda < 1$ regulates the importance that the expert confers to the number of non-relevant map concepts (i.e. variables) with respect to the map quality.

$$G(y, \phi) = (1 - \lambda) \sum_{i=1}^{|\phi|} |I(x, \phi_i) - R(\phi_i)| + \lambda |\psi_Y| \qquad (3)$$

The authors suggests that $\lambda < 0.4$ since the main idea of this learning method is to estimate central concepts by reducing the number of non-relevant nodes, but always preserving the ability of predicting new patterns, and consequently the strength of the original system [25]. Besides, if the induced classification error is greater than a fixed threshold, then the objective function must be penalized using a positive constant.

For solving this complex combinatorial task the framework FCM TOOL uses 8 discrete methods, which includes a Binary-Coded Genetic Algorithm [19], Ant Colony Optimizers [26] and variants of Variable Mesh Optimization [27].

C. *Improving the map convergence*

Most learning methods for computing the causal weights do not accurately consider stability issues when estimating the final solution [3]. As a result, we obtain systems with high prediction rate, but unable to converge to a fixed-point. On the other hand, these methods assume that FCM are closed systems and they do not consider external influences. As far as known, there exist no learning algorithms for enhancing the system stability once the causality is established. Based on these considerations, Nápoles et al. [28] introduced a new learning algorithm which is oriented to estimate the most adequate sigmoid function for each concept. It simulates the outcome of an external stimulus over the neurons with the hope of improving the convergence.

The following equation displays the objective function to be minimized during the search progress. It attempts reducing the system response variability, when activating the causal inference rule. In the objective function, K represents the available number of training instances, M denotes the number of neurons, and T is the maximal number of iterations. In this scheme a solution is considered as no feasible if the accuracy is negatively affected, which allows preserving the global accuracy.

$$H(f_1, f_2, f_3, \ldots, f_M) = \sum_{k=1}^{K} \sum_{i=1}^{M} \sum_{t=2}^{T} \left| A_{i(k)}^{t} - A_{i(k)}^{t-1} \right| \qquad (4)$$

For solving this problem FCM TOOL uses 8 Particle Swarm Optimizers which were conceived to deal with stagnation and premature convergence states [29]. These metaheuristics allow improving the search, hence computing more stable FCM-based systems with high prediction capability.

V. Experiments and Simulations

In this section we illustrate some functionalities of the FCM TOOL (mainly those related to learning algorithms) by using six FCM taken from Nápoles et al. [30]. These systems describe the resistance mechanism of the HIV-1 *protease* protein regarding existing antiviral drugs. More precisely, the authors modeled the protein as a FCM where each sequence position was taken as a map neuron, whereas a decision concept for the resistance target was also defined. The *protease* sequence is defined by 99 amino acids, so it leads to a FCM with 99+1 concepts. But with the goal of reducing the model complexity sequence sites associated with drug resistance are only used [31]. In this topology neurons are fully connected, where a causal relation between each sequence position and the resistance neuron is established.

It is important to mention that each map denotes the protein behavior for a specific drug: Indinavir (IDV), Lopinavir (LPV), Nelfinavir (NFV), Ritonavir (RTV), Saquinavir (SQV) and also Atazanavir (ATV). Moreover, each drug has associated a high-quality filtered datasets consisting on reported mutations and their resistance value [32]. The following figure displays the configuration of the decision neuron (once the map design phase is done) in the software where "0" represents susceptible classes, whereas "1" denotes the resistant ones.

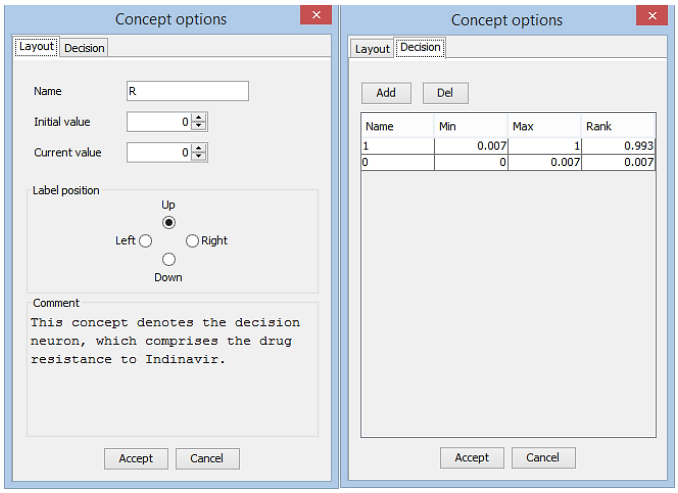

Fig. 7. Configuration of the decision space.

The following workflow shows the semantic of experiments performed next. First, the maps are automatically created from historical data, and afterward we use these data for computing the causal weigh matrices. As a further learning phase, we adopt a second learning method for optimizing the network topology, and lastly the convergence is improved. Figures 9-11 summarize the parameters used for each algorithm and how the FCM TOOL should be configured at each case.

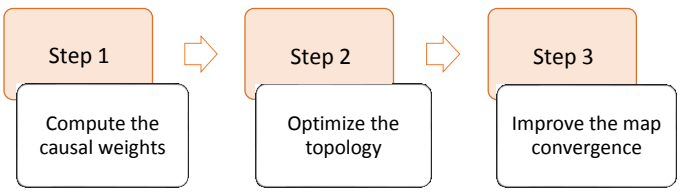

Fig. 8. Description of the methodology used during simulations.

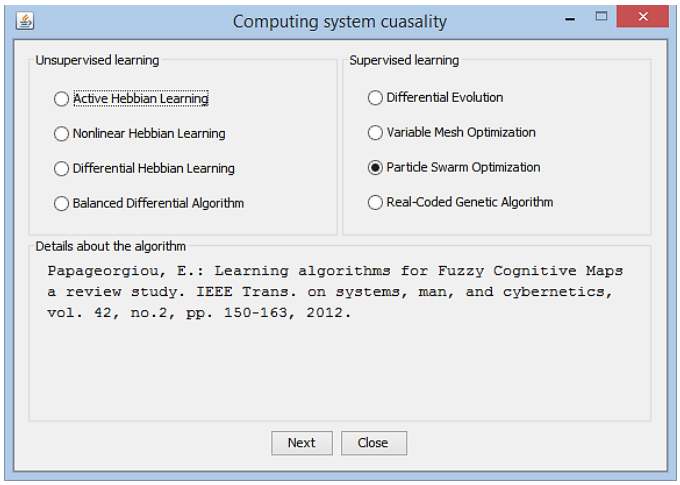

Fig. 9a. Unsupervised learning algorithms based on the Hebb rule and continuous optimizers for estimating the causal weight matrix.

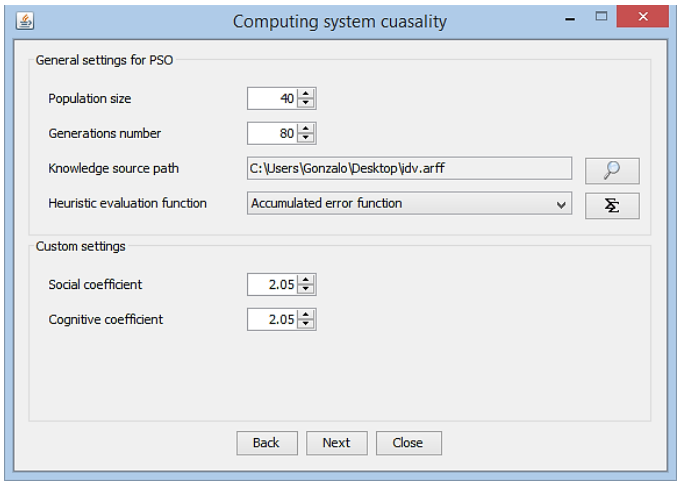

Fig. 9b. Parameters configuration for the selected optimizer (Global-best PSO) when computing the causal weight matrix.

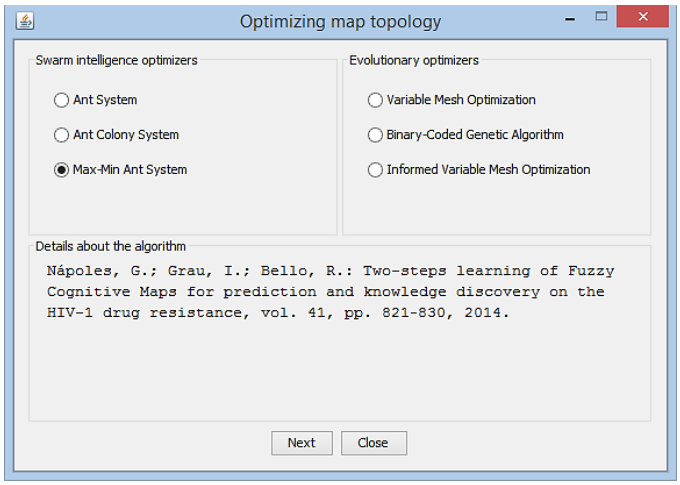

Fig. 10a. Discrete optimizers used for finding the minimal subset of relevant concepts that preserve the overall system modeling.

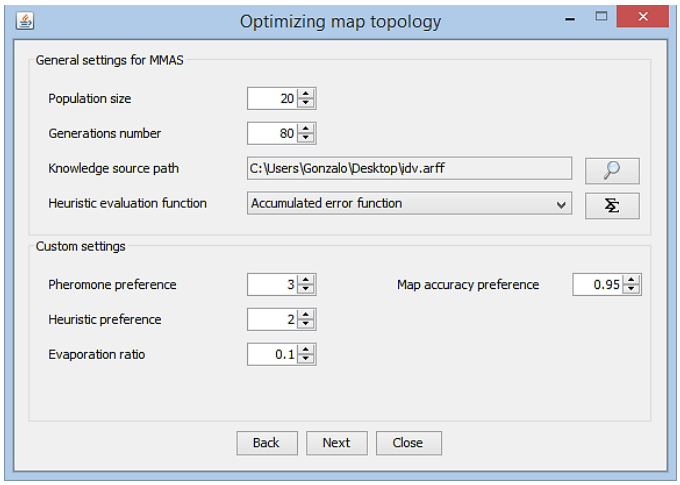

Fig. 10b. Parameters configuration for the selected optimizer (Ant Colony System) when optimizing the network topology.

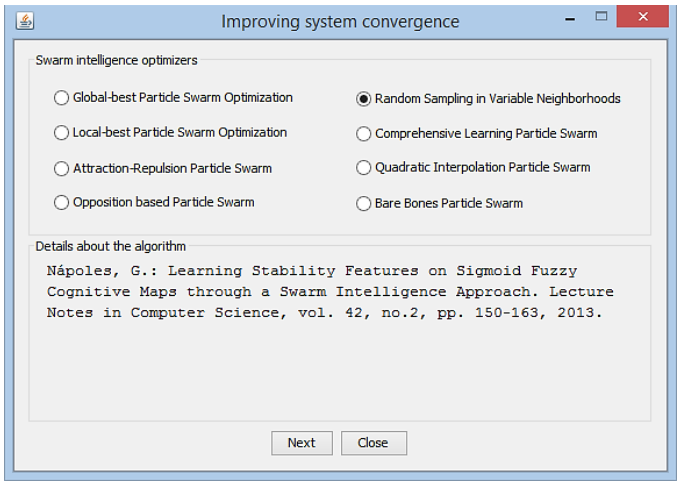

Fig. 11a. Particle Swarm optimizers for estimating the family of Sigmoid functions that allow improving the map convergence.

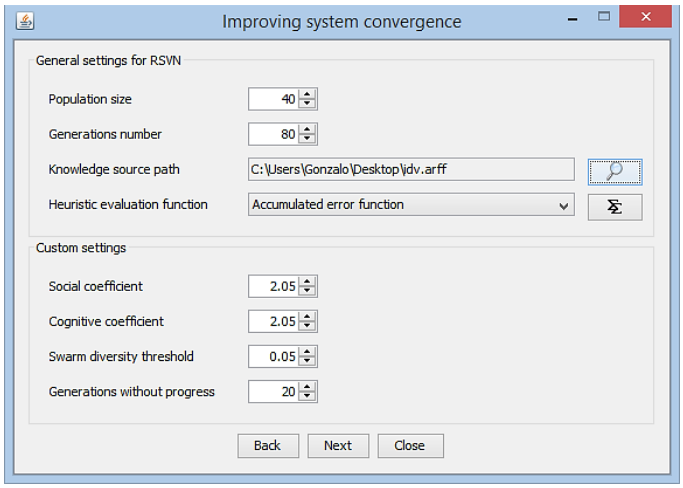

Fig. 11b. Parameters configuration for the selected optimizer (Ant Colony System) when improving the system convergence.

A. *Results of the first learning algorithm*

As mentioned, the estimation of a proper weight matrix for a map implies to estimate at most n^2 causal weights, being n the number of concepts involved in the modeling. In our study case the maximal number of parameters to be estimated correspond to ATV (i.e. 3481 causal values). The following figure portrays the accuracy achieved for each map, once the learning process is done. These values were averaged after 10 independent trials since the learning algorithms implemented for tuning the causal weights are stochastic process, so we could computed different solutions in two independent executions.

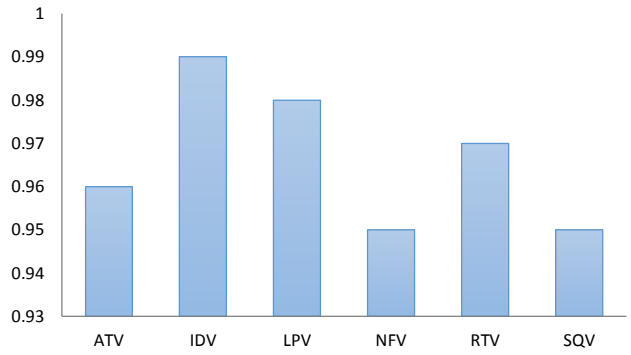

Fig. 12. Accuracy achieved for each FCM once the weight adaptation stage is completed, averaged after 10 independent trials.

It should be stated that the above accuracies are not a result of this paper, however, assuredly they reflect the strength of the implemented learning algorithm for tuning large FCM. Besides, it becomes more relevant if we consider that the proposed tool (which actually is the core of this paper) is completely focused on pattern classification problems, were effectively recognizing such patterns is a complex issue.

The following figure illustrates, as an example, the learning progress for the inhibitor IDV where experts can inspect relevant statistics (e.g. problem features, accuracy, stability status). They are updated step-by-step, which allow visualizing the algorithm progress in real-time. Moreover, we included a panel for plotting the best evaluation computed at each step.

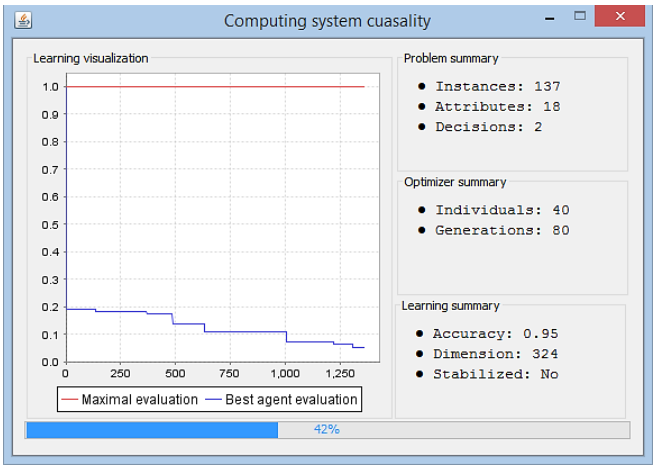

Fig. 13. Window summarizing relevant statistics related to the algorithm progress, when computing the causal weight matrix.

B. *Results of the second learning algorithm*

As mentioned the *protease* protein is defined by 99 amino acids, resulting in six FCM with at most 99+1 neurons. Although the authors used attributes directly related with drug resistance, some of these concepts could be non-relevant for the model and could be removed in order to facilitate the knowledge discovery process. Next figure shows, as an example, the progress of this optimization procedure for the inhibitor IDV. This window also includes numerous statistics such as the map norm (i.e. number of neurons preserved by the method), the reduction ratio and the ΔError (i.e. error induced when removing a subset of concepts, in terms of accuracy). If this value is negative then the accuracy can be improved, which means that removed nodes involve one or multiple contradictory causal relations. More explicitly, in the example $\Delta Error = -0.02$, when removing 7 specific nodes the system is capable of recognizing more patterns (2% of the total number of instances used for testing the model).

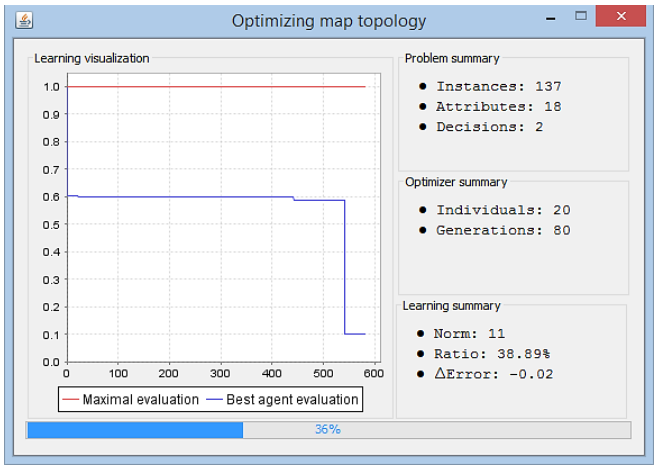

Fig. 14. Windows summarizing relevant statistics related to the algorithm progress, when optimizing the network topology.

Next figure summarizes the number of map concepts at the beginning (as suggested the feature selection) and after applying the learning algorithm. Since the MAX-MIN Ant System [33] is a stochastic method, we averaged 10 independent trials in order to obtain a more realistic value. In short, the reader can observe that this learning algorithm is capable of optimizing the network topology, without significantly affecting the prediction ability of the system (i.e. $\Delta Error \leq 001$ for all drugs).

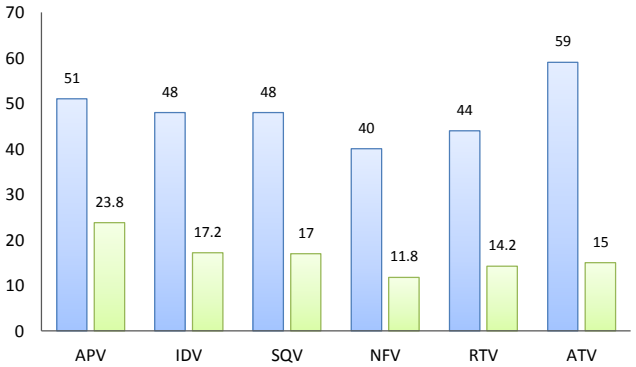

Fig. 15. Number of concepts at the beginning and averaged number of concepts after 10 independent trials.

C. *Results of the third learning algorithm*

As a first analysis, the stability of the decision neuron for a randomly selected mutation is measured. Figure 16, 17 and 18 show the activation value of the resistance over the time for two scenarios: the solid line represents the FCM response without any modification, whereas the dashed line represents the system response using the sigmoid functions estimated by the learning method. From these simulations we can conclude that our model induces better stability. In this case, only the resistance node was monitored, since the decision concept allows predicting whether a new mutation will be susceptible or not.

From these results four scenarios may be formalized: (i) the convergence on stable systems was improved, (ii) cyclic patterns were removed, (iii) the variability response on chaotic systems was reduced, although the system remains chaotic, and (iv) the chaotic behavior was corrected, leading to a perfectly stable map which comprises the most desirable outcome.

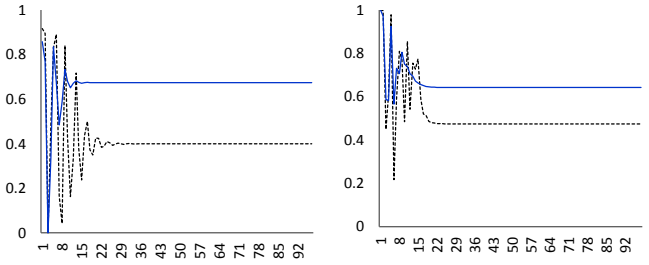

Fig. 16. Activation value of the resistance neuron for a) drug IDV b) drug RTV where the solid line is the map response using the same function for all neurons, and the dashed line represents the map output using the family of functions estimated by the algorithm.

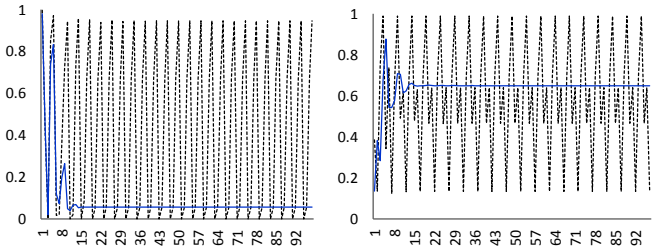

Fig. 17. Activation value of the resistance neuron for a) drug ATV b) drug APV where the solid line is the map response using the same function for all neurons, and the dashed line is the map output using the family of functions estimated by the algorithm.

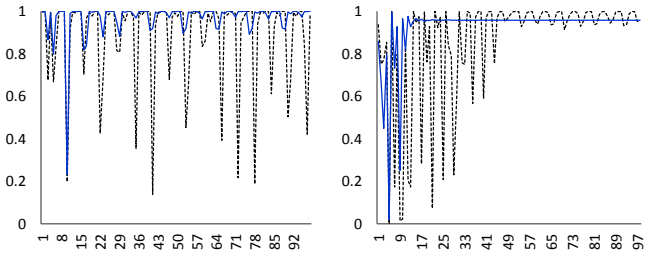

Fig. 18. Activation value of the resistance concept for a) drug SQV b) drug NFV where the solid line is the map response using the same function for all neurons, and the dashed line represents the output using the family of functions estimated by the algorithm.

The reader may observe that the system response changes for next drugs: IDV, RTV and ATV. In such cases the classification rate does not suffer any change since the resistance for a drug is measured in a range instead of using a single value. This range is computed by adopting a pre-defined biological cut-off which allows classifying a mutation in susceptible (0) or resistant (1). However, we noticed that some FCM achieved better accuracy, which is an unexpected positive result. For instance, let us study the behavior of the mutation "FKLDVFMIIVVSVTVNML" for the map IDV. This sequence has high level of resistance for the drug IDV, which means that the higher the activation value of the resistance concept, the better the accuracy for this instance. However, after applying the learning algorithm (see Figure 16a) the FCM computed higher resistance value.

VI. Final Remarks

In recent years the FCM-based analysis has become a fruitful research field. During the design phase of these systems experts determine graph concepts and causal relations (i.e. the direction and intensity) among such entities. Despite the clear advantages of this method as a way for representing the experts' knowledge, learning algorithms are frequently required. This fact comprises a serious problem: codifying these learning methods frequently requires basic knowledge in computer sciences or mathematics, therefore reducing the applicability of FCM on solving real-life problems. Additionally, existing computational frameworks are mainly oriented to conceptualization problems.

This paper discussed some relevant features of FCM TOOL, which is a computational framework for designing, learning and simulating FCM-based systems. This experimentation tool has several functionalities that allow to experts designing a system by only using the Graphical User Interface. Moreover, the FCM TOOL incorporates numerous learning algorithms which belong to the novel trends in learning procedures for FCM (i.e. learning methods for FCM-based prediction systems). These algorithms include models for computing a causal weight matrix with high prediction ability, models for optimizing the network topology without affecting the system accuracy and models for improving the global convergence on sigmoid FCM.

With the purpose of reducing the computational complexity when solving an optimization problem (e.g. the minimal subset of relevant concepts), we used population-based search methods since they are able of estimating good solutions in a reasonable execution time. However, we could adopt exact algorithms (e.g. branch and bounds) as well. That is why we also included in this paper the mathematical formulation of some learning procedures implemented in the framework. As well, we illustrate how these learning algorithms work, by using a complex classification task concerning the HIV resistance to existing drugs.

From the software engineering perspective, the authors were focused on designing an intuitive computational tool that could be expanded to other applications domains. However, it should be stated that most functionalities on the tool were developed for models involving a single decision concept. Actually, the feature work will focused on extending the tool functionalities to more generic pattern classification problems. Since it is really easy to include new methods and algorithms to the software, our final goal is to freely distribute the source code of FCM TOOL among researches of the FCM community.

REFERENCES

[1] B. Kosko, "Fuzzy Cognitive Maps," Int. J. of Man-Machine Studies, vol. 24, pp. 65–75, 1986.

[2] B. Kosko, "Fuzzy Engineering," Prentice-Hall, New York, 1997.

[3] G. Nápoles, R. Bello and K. Vanhoof, "Learning Stability Features on Sigmoid Fuzzy Cognitive Maps through a Swarm Intelligence Approach," Lecture Notes in Computer Science, vol. 8258, pp. 65–75, 1986.

[4] M. León, G. Nápoles, L. Mkrtchyan , B. Depaire, K. Vanhoof, "Tackling Travel Behaviour: An approach based on Fuzzy Cognitive Maps," Int. J. of Computational Intelligence Systems, vol. 6, 1012–1039, 2013.

[5] G. Nápoles, I. Grau, M. León and R. Grau, "Modelling, Aggregation and Simulation of a Dynamic Biological System through Fuzzy Cognitive Maps," Lecture Notes in Computer Science, vol. 7630, pp. 188–199, 2013

[6] E.I. Papageorgiou and J.L. Salmeron, "A Review of Fuzzy Cognitive Map research at the last decade," IEEE Trans. on Fuzzy Systems, vol. 21, pp. 66–79, 2013.

[7] G.A. Papakostas, Y.S. Boutalis, D.E. Koulouriotis, B.G. Mertzios, "Fuzzy cognitive maps for pattern recognition applications," Int. J. of Pattern Recognition and Artificial Intelligence, vol. 22, pp. 1461–1468, 2008.

[8] Mohr, S.: Software Design for a Fuzzy Cognitive Map Modeling Tool. Tensselaer Polytechnic Institute, 1997.

[9] Contreras, J.: "Aplicación de Mapas Cognitivos Difusos Dinámicos a tareas de supervisión y control," Universidad de los Andes, 2005.

[10] M. León, G. Nápoles, C. Rodriguez, M.M. Garcia, R. Bello and K. Vanhoof, "A Fuzzy Cognitive Maps Modeling, Learning and Simulation Framework for Studying Complex System," Lecture Notes in Computer Science, vol. 6687, pp. 243–256, 2011.

[11] B. Kosko, "Hidden patterns in combined and adaptive knowledge networks," Int. J. of Approximate Reasoning, vol. 2, pp. 377–393, 1988.

[12] S. Bueno and J.L. Salmeron, "Benchmarking main activation functions in Fuzzy cognitive maps," Expert Systems with Applications, vol. 36, pp. 5221–5229, 2009.

[13] A.K. Tsadiras, "Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps," Information Science, vol. 178, pp. 3880–3894, 2008.

[14] M. León, G. Nápoles, M.M. Garcia, R. Bello and K. Vanhoof, "Two Steps Individuals Travel Behavior Modeling through Fuzzy Cognitive Maps Pre-definition and Learning.," Lecture Notes in Computer Science, vol. 7095, pp. 82–94, 2011.

[15] E.I. Papageorgiou and A. Kannappan, "Fuzzy cognitive map ensemble learning paradigm to solve classification problems: Application to autism identification," Applied Soft Computing, vol. 12, pp. 3798–3809, 2012.

[16] I. Grau, G. Nápoles, M.M. Garcia, R. Bello, "Predicting HIV-1 Protease and Reverse Transcriptase Drug Resistance using Fuzzy Cognitive Maps" Lecture Notes in Computer Science, vol. 8259, pp. 190–197, 2013.

[17] E.I. Papageorgiou and W. Froelich, "Multi-step prediction of pulmonary infection with the use of evolutionary fuzzy cognitive maps," Neurocomputing, vol. 92, pp. 28–35, 2012.

[18] R. Poli, J. Kennedy and T. Blackwell, "Particle Swarm Optimization – An overview," IEEE Trans. on Evol. Computation, vol. 1, pp. 37–57, 2007.

[19] Z. Michalewicz, "Genetic algorithms + data structures = evolution programs", Springer-Verlag, 1992.

[20] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, pp. 341-359, 1997

[21] E.I. Papageorgiou, C.D. Stylios, P.P. Groumpos, "Active hebbian learning algorithm to train fuzzy cognitive maps," Int. Journal of Approximate Reasoning, vol. 37**,** pp. 219–249, 2004.

[22] J. A. Dickerson, B. Kosko, "Virtual worlds as fuzzy cognitive maps," Presence, vol. 3**,** pp. 173–189, 1994.

[23] E.I. Papageorgiou, C.D. Stylios, P.P. Groumpos, "Fuzzy cognitive map learning based on nonlinear hebbian rule," Australian Conference on Artificial Intelligence, pp. 256–268, 2003.

[24] A. Huerga, "A balanced differential learning algorithm in fuzzy cognitive maps," International workshop on qualitative reasoning, 2002.

[25] G. Nápoles, I. Grau, R. Pérez-García, R. Bello, "Learning of Fuzzy Cognitive Maps for simulation and knowledge discovery," Studies on Knowledge Discovery, Knowledge Management and Decision Making, R. Bello, ed., Atlantis Press, Paris, pp. 27–36, 2013.

[26] M. Dorigo, E. Bonabeau, G. Theraulaz: "Ant algorithms and stigmergy. Future Generation Computer Systems," vol. 16, pp. 851–871, 2000.

[27] R. Bello, A. Puris, Y. Gomez, "Feature Selection through Dynamic Mesh Optimization," Lecture Notes in Computer Science, vol. 5197, pp. 348–355, 2008.

[28] G. Nápoles, R. Bello and K. Vanhoof, "How to improve the convergence on Sigmoid Fuzzy Cognitive Maps?" Intelligent Data Analysis, in press.

[29] G. Nápoles, I. Grau, M. Bello, R. Bello, "Towards swarm diversity: Random Sampling in Variable Neighborhoods procedure using a Lévy distribution," Computación y Sistemas, vol. 18, pp. 79–95, 2014.

[30] G. Nápoles, I. Grau, R. Bello, R. Grau, "Two-steps learning of Fuzzy Cognitive Maps for prediction and knowledge discovery on the HIV–1 drug resistance, Exp. Sys. with App.," vol. 41, pp. 821–830, 2014.

[31] V.A. Johnson, V. Calvez, H. Günthard, "Update of the Drug Resistance Mutations in HIV-1," Topics in HIV Medicine, vol. 21, pp.6–14, 2013.

[32] S.Y. Rhee, et al. "Human immunodeciency virus reverse transcriptase and protease sequence database," Nucleic Acids Research, vol. 31, pp. 298–303, 2003.

[33] T. Stützle, H. H. Hoos, "MAX–MIN ant system," Future Generation Computer System, vol. 16, pp. 889–914, 2000.