Cognitive Software-Defined Networking Using Fuzzy Cognitive Maps

Giovanni Baggio, Riccardo Bassoli *Member, IEEE*, and Fabrizio Granelli, *Senior Member, IEEE*

***Abstract*—Future networks are expected to provide improved support for several different kinds of applications and services. All these services will have diverse characteristics and requirements to be satisfied. A potential technology to upgrade efficiently and effectively current generation networks is virtualisation via network 'softwarization'. This approach requires the combination of software-defined networking and network function virtualisation. Nevertheless, such a new complex network structure will raise further issues and challenges to be solved both reactively and proactively, without human intervention. In order to achieve that, academia and industry have identified the solution in the implementation and deployment of machine learning. Hence, very likely, 5G (and especially beyond 5G) networks will be cognitive virtualised networks.**

In that context, this article proposes a cognitive software-defined networking architecture based on Fuzzy Cognitive Maps. First, specific design modifications of Fuzzy Cognitive Maps are proposed to overcome some well-known issues of this learning paradigm. Second, the efficient integration with a software-defined networking architecture is presented and analysed. Finally, the emulation of a sample network scenario via Mininet is provided to validate the effectiveness and the potential of the new cognitive system and its capability to act and to adapt independently of human intervention.

***Index Terms*—Cognitive networks, Fuzzy cognitive maps, software-defined networking, network virtualisation.**

I. Introduction

A cognitive network is a network which implements 'intelligent' procedures – inspired by the cognition process of real people – to learn from current operating conditions in order to adapt and to make future decisions, targeting an end-to-end goal. This concept was firstly proposed [1], [2] in 2006 and 2008 respectively. The main required characteristic [1] to enable the deployment of cognitive networks is the existence of software-adaptable networks (SANs), which include software-based network functionalities.

During the last few years, growing research on network function virtualisation (NFV) [3] and especially on software-defined networking (SDN) [4], provided a suitable infrastructure to pave the way towards an actual implementation of cognitive networks. Moreover, the importance of NFV and SDN grew towards the design and implementation of future 5G cellular networks [5]: in fact, they are seen as the main technological enablers in order to achieve higher data-rate, lower latency and adaptivity for their diverse service scenarios. Indeed, in the cognitive networking paradigm, network 'softwarization' provides exactly what is missing in current networks:

- the capability to adapt themselves via software reconfiguration;
- a higher level of centralisation in the control plane, surely beneficial to implement more efficiently reasoning and learning processes of a cognitive network.

Nevertheless, the path towards actual deployment of the potential of cognitive networks is still relatively long, as few works target the core of the problem: the choice of an effective decision-making procedure to support efficient data mining, analysis and inference process [2].

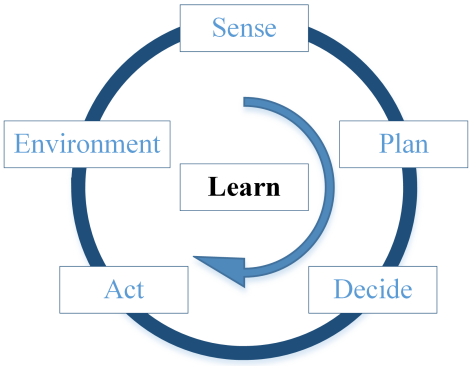


Fig. 1. The cognition loop adapted from [6].

Cognitive networks are based on the so called cognitive loop (Figure 1). A complete cognitive process is based on Sense-Plan-Decide-Act, which is accompanied by learning and final environmental feedback. The Sense state is responsible for information gathering, monitoring and pre-processing. Next, the Plan state interprets the collected data according to a model and use the results for planning. The Decide state identifies the actions to be performed according to policies and processed information, and evaluates eventual alternatives. Finally, the Act state realises the actual action on the network. The environmental information is used as input to re-start the loop and helps in the final check of the effects: it can also send alarms and warnings. All the previous states provide a feedback, which helps the learning algorithm to improve the performance and responsiveness according to the defined policies.

Among the initial works about cognitive networks, some significant ones theorised about the best tools a cognitive entity should use to carry out their optimisation tasks ([1], [7]), while others, instead, adopted specific reasoning techniques: anyway,

G. Baggio is with FBK CREATE-NET, WiN research unit, Trento, Italy (e-mail: g.baggio@fbk.eu).

R. Bassoli and F. Granelli are with the Department of Information Engineering and Computer Science, at the University of Trento, Trento, Italy (e-mail: {riccardo.bassoli, fabrizio.granelli}@unitn.it).

often they did not clearly justify the reason for the choice among those techniques [6]. In the last decade, several publications have investigated the deployment of various machine learning (ML) algorithms to enhance the limited management capabilities of legacy SDN. Nevertheless, in 2011 [8], the authors of this paper were the first to propose a suitable reasoning formalism for cognitive networks based on Fuzzy Cognitive Maps (FCMs), which can be used to perform causal reasoning. The innovative aspect lied in the adoption of such a tool to explicitly exploit cause-effect relationships among variables in the protocol stack of network nodes. The choice of FCMs was based on the idea that variables at different layers of the protocol stack share cause-effect relationships. Moreover, FCMs allow the presence of loops, which are present in several aspects of networking (see, for example, closed-loop operation of flow and congestion control in TCP, automatic rate fallback in WiFi, etc.).

Fuzzy cognitive maps [9] have significantly attracted the attention of the research community in the last fifteen years. The main limitations of FCMs, which have been identified, are [10]: edges' weights are linear, they cannot represent logical operators between ingoing nodes, they cannot model multi-meaning environments, they do not include multi-state concepts, they cannot handle more than one relationship between nodes, they are symmetric or monotonic (that does not happen for various real causal relations), and their first order dynamics cannot handle randomness associated with complex domains.

Learning algorithms for FCMs can be organised into three main families [11]: Hebbian-based, population-based and hybrid approaches. The first type uses the basis of experts' knowledge to lead FCMs to converge into an acceptable solution. The second employs historical data without any expert's intervention. The third is based on both experts' input and historical data. The first detailed implemented architecture for FCM-based cognitive networks presented in [8] and [12] applied a Hebbian-based learning algorithm. These works provided a framework to design FCMs for cognitive networks by overcoming the above limitations, which could have negatively affected their deployment in the context of communications networks. However, the authors of [8], [12] have not clearly defined a criterion for the FCM-based algorithm to detect when the inference process reaches a satisfying solution. Basically, those works provided evidence of the potential of FCMs for cognitive networking, without achieving a general solution in order to predict when the system gets its final status. Still, they represent a relevant milestone toward an actual architecture able to support cognitive networking.

The contribution of this article includes:

- the definition of an enhanced theoretical model for FCMs applied to wired networks. This model capitalises on previous results in [8], [12] to improve FCMs towards their effective deployment in cognitive networks. In particular, the article focuses on some critical aspects affecting classical FCMs' inference process, which – according to the best of authors' knowledge – have not been investigated yet in the scientific literature.
- the theoretical and practical proof that our enhanced Hebbian-based FCM can work effectively without a-priori knowledge of the network status. In fact, by considering the taxonomy of ML-based SDN [13], ML approaches generally employ so called training datasets, i.e. human supervision.
- the design of an actual and effective architecture, where the proposed enhanced FCMs are integrated into SDN-based networks. That is followed by an actual validation of this proposed system on an emulated testbed, therefore proposing one of the first complete implementations of FCM-based cognitive networking to date.

The rest of the paper is organised as follows. Subsection I-A justifies the choice of FCMs for SDN-based cognitive networks, by showing their advantages in respect of other ML algorithms. Section II describes the mathematical notation and problem statement, as originally proposed in [12]. Furthermore, it briefly presents the Hebbian-based algorithm used for learning. Next, Section III fully discusses the novel enhanced model for Hebbian-based FCMs, which improves the theoretical framework previously developed in [12]. Our theoretical solutions improve the correctness/performance of the algorithm towards efficient deployment in real cognitive networks. Section IV deals with the design of a cognitive software-defined networking architecture based on our proposed FCMs. This section considers architectural analysis and evaluation of overhead and latency introduced in SDN paradigm by our enhanced Hebbian-based FCMs. Finally, Section V provides results to verify the effectiveness of our FCM-based SDN in two exemplar virtualised networks, which have been implemented in Mininet environment.

A. Related Works and Motivation

Regarding unsupervised learning and SDN, the exhaustive survey about ML-base SDN [13] identifies two main approaches in the literature: *k*-Means SDN and self-organising map (SOM) SDN. The former is a popular method to classify unlabelled data, so it make it useful both for classical and QoS-aware traffic classification in SDN-based networks. It is mainly limited to clustering problem solving and it is computationally expensive, in particular for large maps with big amount of training data. Recently, k-means have also been applied to SDN to solve controller placement problem [14], [15], [16], which is an important issue in presence of multiple SDN controllers. The latter [17] is also principally employed to solve clustering problems: its data mapping is easier to be understood and capable to handle big datasets, while it is also computationally expansive mainly for large training datasets.

Side by side, reinforcement learning (RL) [18] and deep reinforcement learning (DRL) [19], [20] are additional ML paradigms, in which an 'unsupervised agent' interacts with its environment to learn the best action to perform in order to maximise its long-term reward.

Moreover, the taxonomy of works about ML applied to SDN reveals that ML algorithms are very often limited to specific aspects of SDN network management such as traffic classification, Quality-of-Service (QoS), routing optimisation, resource management or security: there has been no approach yet, which has tried to use unsupervised ML/RL to realise a

sort of SDN-based autonomic network (i.e. an SDN network that manages itself without human intervention).

Regarding routing optimisation, the authors of [21] designed a centralised Cognitive Routing Engine (CRE), based on Random Neural Networks with Reinforcement Learning, was to find the optimal network paths. In 2017, [22] applied RL to improve the performance of routing protocols in SDN. The same year, [23] developed a DRL agent to optimise routing towards a decrease in network latency. Next, [24] designed DRL paradigm for routing optimisation in SDN-based environments. Reference [25] proposed solution based on k-means to optimise routing protocol in SDN.

In the context of resource management, article [26] designed a content delivery framework based on RL and SDN to select the optimal protocol. Next, [27] deployed RL to minimise the long-term reconfiguration cost of roadside unit cloud network based on SDN. In 2017, [28] applied RL and game theory for activation of servers in mobile edge computing based on SDN. In parallel, [29] applied RL and game theory for resource allocation and management in distributed environment based on SDN. Recently, [30] applied DRL for multimedia traffic control in SDN in order to achieve high quality of experience.

Given the above context, this article is the first one to design in detail an FCM-based SDN system for unsupervised virtual network management while providing a quantitative study of its performances. Another pivotal characteristic of this work is the scope of ML paradigm: the aim of our FCM-based controller is not to optimise single aspects of networking but to guarantee efficient and effective self management of the SDN-based network. Thus, the cognitive controller has to control and to manage various aspects of networking to guarantee good routing management while enhancing QoS provisioning via effective management of resources.

The choice of FCMs comes from a major aspect in the area of unsupervised autonomic networks: human monitoring. In fact, humans have to be able to monitor and to understand easily how the cognitive controller thinks and acts: that to prevent unwanted bad events. One of the main advantages of FCM is exactly the clear and easy-to-understand representation of knowledge [9], which does not happen with solutions based on k-Means, SOMs and RL. This is important since in unsupervised systems humans have to be able to control how the unsupervised system thinks and make decisions.

Furthermore, while our design of FCMs is mainly focused on resource/routing management and QoS optimisation (in this article), the FCM-based controller can easily be employed to consider and to manage other aspects of networking by adding further variables to the FCM (e.g. variables referred to actions and quality).

Fuzzy Cognitive Maps have more advantages than other reasoning techniques such as neural networks, Bayesian and Markov networks. Unlike both Bayesian and Markov networks, the inference procedure of FCMs is less complex since it only involves vector-by-matrix multiplications and thresholding operations [31]. Moreover, FCMs are more effective to represent causality loops in a problem; for these kind of problems Bayesian networks cannot be applied [32], [33]. In Markov networks, the presence of loops cannot guarantee the convergence of belief propagation algorithms [12]. On the other hand, the FCM inference procedure is also effective to solve problems involving no loops. In particular, inference is guaranteed to converge to either a fixed point or a limit cycle, provided that concepts take their values in any finite discrete set [34].

Next, when neural networks are employed to model dynamic systems, the obtained solution does not necessarily reflect the actual relationships among its system variables [35]. On the other hand, the edges of an FCM faithfully represent those relations and are, therefore, more appropriate to analyse both direct and indirect cross-layer interactions [33], [12]. Further advantage is the possibility of exchanging/merging multiple FCMs, resembling operations people do when they exchange their opinions [12]. This aspect has its roots in the primary purpose for which FCMs were created, i.e. to allow experts to represent their causal knowledge about some situation. Different people may have different opinions about the same matter, and may encode differently their beliefs, hence drawing conflicting FCMs. Merging helps to smooth (possibly divergent) beliefs and biases, thereby reducing the possibility of biased reasoning. Moreover, weights can be employed to give more or less credit to each FCM. Finally, the 'composed FCM' can contain potentially non-overlapping FCMs, thus, enabling the exchange of knowledge in case the domain of knowledge of cognitive entities is different, which undoubtedly is an advantageous feature when dealing with uncertain scenarios such as communication networks.

The main drawback of legacy FCMs is the automatic synthesis of the maps: FCMs were not originally designed for being constructed starting from observational data but they were initially devised in the social science field as a tool to help experts to express their beliefs about a given matter [33]. For this reason, self-synthesis of FCMs is hard, mostly because, for non-humans, cause/effect relationships between variables are generally more complex to detect than simple correlations [33].

Finally, the results obtained by our designed and implemented FCM-based SDN keep comparable delays with legacy SDN Open Network Operating System (ONOS), thus the reasoning and acting procedures are performed almost transparently. Moreover, our proposed cognitive SDN system can achieve 100% accuracy in average thus overcoming all the legacy ML-based SDN solutions [13].

II. Preliminaries and Background

A. Software-defined Networking

Software-defined networking is a virtualisation paradigm, which permits a software-based control of data-paths and routing strategies of networks. This technology aims at detaching control and data plane of communication networks. The central entity of SDN virtual architecture is the SDN controller, which updates flow tables and policies at so called SDN switches. Moreover, an application layer is responsible to handle different networking applications such as control of data paths, user authentication and management of mobility. Among various protocols proposed for controller-switches

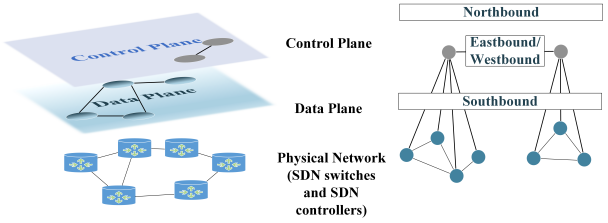


Fig. 2. Structure of SDN virtual network (left side) and structure of SDN architecture (left side), which specifies the logic interfaces of the architecture.

communications, the standard protocol currently in use is Openflow: its aim is to change flow tables of Openflow SDN switches, which contains rules, actions and policies related to data traffic. Next, SDN switches collect statistics to be sent to SDN controller for basic network control.

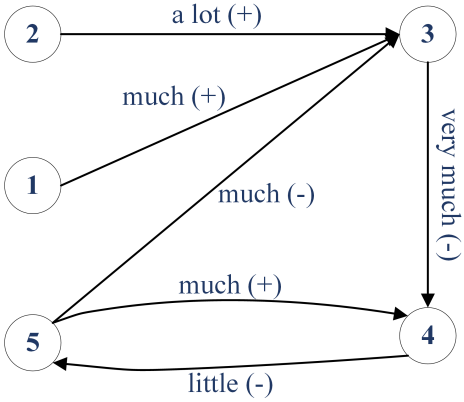
Figure 2 depicts SDN network structure (left side), which has previously been described. The same figure (right side) shows the logic architecture of SDN and its principal interfaces. In particular, the controllers (if multiple are allowed) communicate each other via the eastbound/westbound interface. Next, the northbound interface is the one that translates the different third-party application requirements into network commands. Finally, the southbound interface allows the controllers to set up the flow tables of SDN switches via the Openflow protocol to allow the creation of virtual networks according to the needs of different services.

B. Introduction to Fuzzy Cognitive Maps

The 'cognition loop', graphically represented in Figure 1, is central to any cognitive architecture. Cognitive networking undoubtedly needs a cross-layer approach to operate, which represents the means to provide optimisation, while the cognitive engine deals with learning, adaptation and decision process to achieve end-to-end goals [12].

FCMs are mathematical structures proposed in 1986 [9] as a means for modelling dynamic systems through the causal relationships, which characterise them. Figure 3 shows a simple example of graphical and matricial representation of an FCM. The former is a directed graph, in which a node represents a generic concept and an edge is the causal relation between two connected concepts: especially, the cause is the starting node of the directed edge.

The domains of nodes and the weights of edges can be either discrete or continuous sets. The latter is used in more complex FCMs, where a high level of detail needs to be achieved. For example, by extending a continuous set from $[0,1]$ to $[-1,1]$, it generally results in a greater flexibility of the model. The use of simpler ranges of the same model is normally suited to obtain a preliminary representation of a problem. If a concept is zero, it means the concept is 'off', 'inactive', in a 'low-state' or is completely neglected. On the other hand, a concept set to one means it is 'high' or 'active'. Edge weights measure the degree of causality; values such as $+1$ or -1 denote a strong causal relationship, either positive or negative. An edge labelled with zero means that two concepts are not causally related. Clearly, the graph cannot have loops since concepts



$$F = \begin{pmatrix} 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & -0.2 \\ 0 & 0 & -0.6 & 0.6 & 0 \end{pmatrix}$$

Fig. 3. Example of FCM structure and its respective matricial representation.

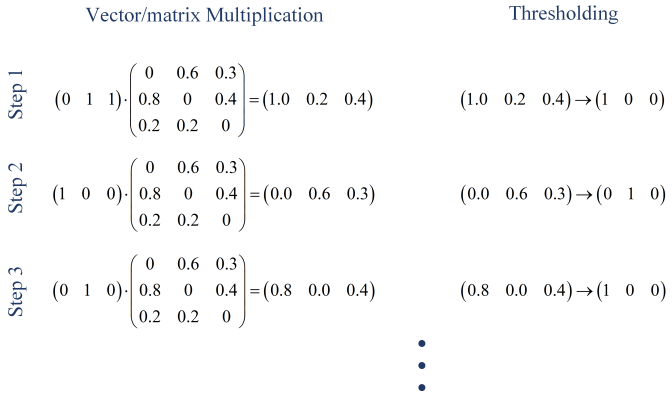


Fig. 4. Example of how the inference process of an FCM works. Thresholding considers the threshold value 0.5.

cannot cause themselves. Thus, the diagonal of the adjacency matrix F of an FCM only has zeros.

The state of an FCM-based system with n distinct concepts is a vector of dimensions $1 \times n$. Then, the inference process consists in various multiplications of this vector by the FCM matrix. Next, the result is thresholded each time, until it converges either to a fixed point or to a limit cycle. Figure 4 depicts a simple example of this inference procedure.

C. Fuzzy Cognitive Maps for Cognitive Networks

FCMs have been proposed to implement reasoning and learning in cognitive networks [12]. In order to target an end-to-end goal, the FCM should consider parameters associated to different layers, and the FCM concepts should be mapped to communication protocol internals. As proposed in [12], the concepts can be grouped into three main categories according

to their relationship with Quality-of-Service (QoS) metrics, to environmental characteristics in which the cognitive entity is, and to the set of actions that a protocol can perform. Mathematically, these concepts respectively form the triple of vectors $(\mathbf{q}, \mathbf{e}, \mathbf{a})$, which represents the system state vector $\mathbf{s}$.

The objective of the FCM is the convergence to a solution state $\mathbf{s}^* = (\mathbf{q}, \mathbf{e}, \mathbf{a}^*)$, achieved by finding a vector $\mathbf{a}^*$ such that the constraints expressed by $\mathbf{q}$ are satisfied before environmental conditions $\mathbf{e}$ change. Since vectors $\mathbf{q}$ and $\mathbf{e}$ are known, the search space of the algorithm is limited to the elements of $\mathbf{a}$: that avoids the problem to be NP-hard.

However, in order to map the concepts in the most effective way, it is fundamental to choose the right domain. Concepts can be expressed either by discrete sets or continuous sets. In particular, it is normally better to avoid continuous sets because they can results into chaotic behaviour [12]. Continuous ranges of values can be converted into discrete sets by applying a thresholding procedure. However, finding the optimal threshold for an application may not be straightforward.

On the other hand, the size of a discrete set can be chosen according to the needs. For example, concepts that convey an ON-OFF relationship can use the set $\{0, 1\}$, and the ones requiring to invert causal relationship can use the set $\{-1, 1\}$.

After defining the concepts and their domains, it is important to select a learning algorithm for FCMs. The following subsection provides some details on one of the existing learning algorithms. That should be considered as a concrete example of how learning can be implemented, but other methods are available and can be used without architectural differences with respect to the solution presented in this paper.

D. Differential Hebbian Learning Algorithm

In our work we borrow the problem statement proposed in [12]: hence, we apply a Hebbian-based algorithm called differential Hebbian learning (DHL) algorithm [36]. The main drawback of this method is that the formula updates weights between each pair of concepts without considering the influence of other concepts.

In order to explain how DHL algorithm works, let's consider a simple FCM with two concepts named A and B. If both A and B change their values from 0 to 1, then it can be inferred that the positive variation of one concept has caused a positive variation of the other. Similarly, if both A and B change their values from 1 to 0, it can be inferred that the negative variation of A has produced a negative variation of B. In both cases the sign of the variation is positive for these two concepts because a positive causal relationship is present.

Let's now suppose that A changes its value to 1 and B, which before the variation of A had the value set to 1, goes to 0. In this case, there is a negative causality relationship because a variation of the first concept led to the opposite variation of the second, thus the edge connecting the two concepts is now negative (for simplicity let's assume the system is memoryless, meaning that the algorithm does not consider the previous states).

Formally, let C_i and C_j denote two generic *concepts*, and let $\dot{C}_i$ and $\dot{C}_j$ denote their variations over time (time derivatives). Moreover, let $\dot{f}^t_{i,j}$ label the variation of the edge's value, connecting concept C_i with concept C_j at time instant t. Then, the differential Hebbian law states that

$$\dot{f}^t_{i,j} = -f^{t-1}_{i,j} + \dot{C}^t_i \dot{C}^t_j \tag{1}$$

where the derivatives represent changes in two generic concepts and the result of their product reveals the correlation among them.

Expression (1) shows that a variation of concept C_i occurs before the variation of concept C_j, the other way around if C_j was the first concept changing then the edge to be updated would be $f_{j,i}$. The negative value of the edge at time $t-1$ is included in the right-hand side of equation (1) to prevent that a spurious simultaneous variation impacts indefinitely.

Next, the edge at step t is computed as the value it had at time step $t-1$ plus the variation:

$$f^t_{i,j} = f^{t-1}_{i,j} + \dot{f}^t_{i,j} = \dot{C}^t_i \dot{C}^t_j \tag{2}$$

Thanks to the term $-f^{t-1}_{i,j}$ in expression (1), the value on the edge can be set to zero when one of the concepts do not change, thereby indicating that the previous causal relationship no longer holds.

As a result, the learning process does not update the knowledge considering only the current causality relationships degree but also keeping in consideration the past value.

Depending on the application, such update may be preferred to be more responsive or vice versa, the responsiveness of the algorithm can be modified by introducing a parameter η in the previous formula known either as *learning rate* or as *decreasing learning coefficient*. In other words, it can be seen as a smoothing parameter, capable to avoid significant oscillation in the values of the edges (the amplitude of these variations depends on the magnitude of the coefficient value). Moreover, parameter η is generally defined in the set $(0, 1]$: values close to zero result in a slowly changing FCM, while values close to the unity produce a highly responsive FCM.

Finally, the learning formula becomes

$$f^t_{i,j} = f^{t-1}_{i,j} + \eta(-f^{t-1}_{i,j} + \dot{C}^t_i \dot{C}^t_j) \tag{3}$$

It is worth notice that if η is set to one meaning that the past value on the edge is not considered when it is updated, then edges can assume values in the discrete set $(-1, 0, 1)$ while a different η results in continuous range of values.

III. Enhanced Fuzzy Cognitive Maps for Cognitive Networks

The work in [12] proposed FCMs based on discrete domains instead of continuous ones and introduced thresholding in form of pre-processing in order to map values into discrete domains. This section demonstrates the potential of using continuous ranges of values, by avoiding thresholding. In particular, we show the significant advantages obtained by interpreting continuous values in the range $[-1, 1]$ as 'directed' (signed) probabilities.

Moreover, we also show it is actually possible to acquire knowledge not only related to the causality degrees among nodes but also to create, modify and delete persisting causal

relationships among them. In other words, the system is able to start from an empty matrix, which means that no causal relationships have been inserted a-priori (i.e. no knowledge is provided to the system), and to build the edges during run time by analysing the environmental cause-effect relationships (i.e. the system can learn during run time). That would allow a relevant capability 'to learn' in an autonomous manner and, in perspective, leading to a significant reduction in human intervention in managing the network.

A. Finding the Desired State

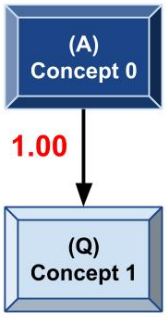
First of all, it is necessary to identify an effective way to detect either when the inference process has reached a satisfying solution or when the process should continue. In [12], this aspect was not completely investigated. Thus, this subsection explains in detail how the inference process works and the meaning of each of the steps performed during its operation.

Figure 5 shows an example of an extremely simple FCM and, on the right, the log of the inference process is reported. A mathematical way to describe the connections between concepts is the use of adjacency matrix since an FCM can be represented as a directed graph. The example in Figure 5 has two concepts so it produces a 2×2 adjacency matrix. The only connection between those concepts is the directed line from Concept 0 to Concept 1. Then, the corresponding adjacency matrix becomes

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{4}$$

Since there are no loops, the algorithm ends with a zero-value array, which means that no further step has to be taken: in fact, the knowledge involved in the variables and their relationships have already been exploited. In order to simplify the explanation, all of the examples reported are based on the assumption that quality concepts are considered in a bad shape or alert state when their value is high. The example of Figure 5 shows a positive causal relationship from an action to a quality concept. Once the latter enters in an undesired state, the inference process is triggered and the consequences for the two possible states of the action variable are investigated. In order to keep this example simple, the causality relationships are supposed to be deterministic, thus we work with integer numbers: however, the same procedure would have also been applied in the continuous case.

By looking at the log (Figure 5, on the right), the repetition of two identical vectors at the end of each trial are produced by the inference process in order to automatically understand when no further steps have to be performed, i.e. when the knowledge of the FCM has already been fully exploited for making a decision. The process finishes with the conclusion that *Concept 0* has to be set to a low value because this choice – according to the information embedded in the FCM – will move the system in a safer condition, so a low state for the quality concept.



(a)

	Action vectors / Inference process status (steps)		Valid
	(A) Concept 0	(Q) Concept 1	
Action vector 1	-1	1	/
Step 1	0	-1	YES
Step 2	0	0	NO
Step 3	0	0	NO
Action vector 2	1	1	/
Step 1	0	1	NO
Step 2	0	0	NO
Step 3	0	0	NO
RESULT	Action "(A) Concept 0" must change status to "Low"		

(b)

Fig. 5. (a) Simple FCM with positive causality (b) Description of FCM inference process. The end of the inference process happens when there is a repetition of two identical vectors at the end of each trial.

B. Belief Propagation for Fuzzy Cognitive Maps

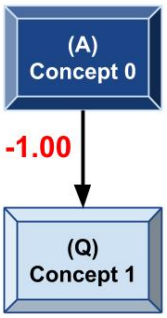
The inference process implementation of [12] suggests that at each step the output of the vector-by-matrix multiplications has to be rounded with a threshold, which maps negative values to zero and positive values to one. There are two main problems related to this approach.

First, it is not possible to propagate negative causal relationships because zero-valued relations stop the inference process. The solution of this issue is to avoid thresholding. Figure 6 depicts an example where there is a negative causal relationship from the action to the quality concept. The corresponding adjacency matrix is

$$\begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} \tag{5}$$

Since negative values in the output of the vector-by-matrix multiplication are brought to zero it is unavoidable that causal relationships are interrupted.

Second, precious information stored in the continuous set $[-1, 1]$ is lost. According to the formula presented in the learning part, the edges can acquire values in the continuous set $[-1, 1]$, which can be treated as signed probabilities: in particular, the sign on these values means that there is a certain probability (the absolute value of the edge) that either



(a)

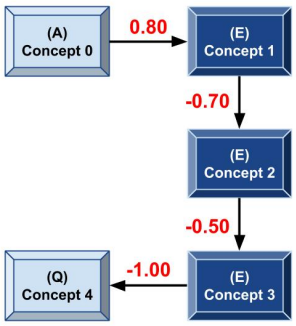
	Action vectors / Inference process status (steps)		Valid
	(A) Concept 0	(Q) Concept 1	
Action vector 1	-1	1	/
Step 1	0	1	NO
Step 2	0	0	NO
Step 3	0	0	NO
Action vector 2	1	1	/
Step 1	0	0	NO
Step 2	0	0	NO
RESULT	No valid solution can be found		

(b)

Fig. 6. (a) Simple FCM with negative causality (b) Description of FCM inference process. Example that shows the system cannot come to a decision if thresholding is applied.

a positive or a negative relationship is present. As an example, let's consider three concepts A, B and C such that A implies B with a value on the edge equal to 0.7, and B implies C with a strength of −0.5; then the inference process will reveal that A indirectly implies C with a value of −0.35, as we can see the negative causal relationships are preserved thanks to this approach. Moreover, since the values in the matrix of the FCM system represent the past experience, referred to a particular cause triggering a consequence with a certain degree of confidence, we can claim that the absolute values on the edges represent probabilities. Hence, in the previous example, the value −0.35 is the probability resulting by the product of the probabilities of the two edges. If A implies B with a positive causal relation of probability 0.7 and B implies C with a negative causal relation of probability 0.5 (note that here the negative sign is missing because it is just related to the causal relation of the probability), then an action on A will result in a consequence to C with a negative causal relation of probability 0.35. Then, since the obtained value is negative, the target concept C has to be considered low. Furthermore, if two probabilities with negative sign had been involved, their product would have been resulted in a positive causal relationship.

Figure 7 shows an example of a chain composed by five concepts connected in a row, with arbitrary values in $[-1, 1]$



(a)

	Action vectors / Inference process status (steps)					Valid
	(A) Concept 0	(E) Concept 1	(E) Concept 2	(E) Concept 3	(Q) Concept 4	
Action vector 1	-1	0	0	0	1	/
Step 1	0	-0.8	0	0	0	NO
Step 2	0	0	0.56	0	0	NO
Step 3	0	0	0	-0.28	0	NO
Step 4	0	0	0	0	0.28	NO
Step 5	0	0	0	0	0	NO
Step 6	0	0	0	0	0	NO
Action vector 2	1	0	0	0	1	/
Step 1	0	0.8	0	0	0	NO
Step 2	0	0	-0.56	0	0	NO
Step 3	0	0	0	0.28	0	NO
Step 4	0	0	0	0	-0.28	YES (score = 0.28)
Step 5	0	0	0	0	0	NO
Step 6	0	0	0	0	0	NO
RESULT	Action "(A) Concept 0" must change status to "High"					

(b)

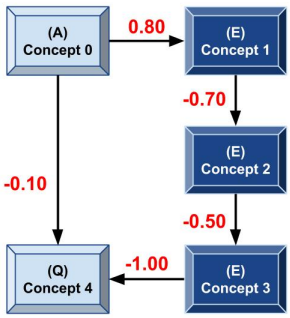
Fig. 7. Example that shows the interpretation of continuous values as probabilities (a) Example of an FCM (b) Description of FCM inference process.

on the edge. The adjacency matrix of this FCM is

$$\begin{pmatrix} 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & -0.7 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6}$$

By interpreting the edges' values as probabilities, it is possible to come to the final decision through a belief propagation path, which provides the highest confidence.

In order to define the meaning of the variable *score* displayed in Figure 7, let's focus on the example reported in Figure 8. There are two causality chains, which both lead to the same quality parameter and enforce the same decision (the



(a)

	Action vectors / Inference process status (steps)					Valid
	(A) Concept 0	(E) Concept 1	(E) Concept 2	(E) Concept 3	(Q) Concept 4	
Action vector 1	-1	0	0	0	1	/
Step 1	0	-0.8	0	0	0.1	NO
Step 2	0	0	0.56	0	0	NO
Step 3	0	0	0	-0.28	0	NO
Step 4	0	0	0	0	0	NO
Step 5	0	0	0	0	0	NO
Action vector 2	1	0	0	0	1	/
Step 1	0	0.8	0	0	-0.1	YES (score = 0.10)
Step 2	0	0	-0.56	0	0	NO
Step 3	0	0	0	0.28	0	NO
Step 4	0	0	0	0	-0.28	YES (score = 0.28)
Step 5	0	0	0	0	0	NO
Step 6	0	0	0	0	0	NO
RESULT	Action "(A) Concept 0" must change status to "High"					

(b)

Fig. 8. Example that shows how combination of paths works (a) Example of an FCM, which has two causality chains, which both lead to the same quality parameter and enforce the same decision (b) Description of FCM inference process.

action variable has to be set to a high state in this case). The adjacency matrix becomes

$$\begin{pmatrix} 0 & 0.8 & 0 & 0 & -0.1 \\ 0 & 0 & -0.7 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{7}$$

The weight associated to each path is used for comparison: in the output of the implementation reported near the FCM scheme, it is specified a *score* value, which is always positive. This value represents a metric that can be used in order to determine which causality chain better supports the choice of a specific decision. It is possible to notice that the system distinguishes between the two possible solutions according to the causality path, which ends with the higher score. This result is a direct consequence of the proposed novel interpretation, which changes how the inference process works. Furthermore, this probabilistic approach enables the system to choose among different possible results. Moreover, the value of the score is strictly related to the confidence that an action will get specific effects: thus allowing to predict the probability the system will be able to recover from an alert state.

By avoiding thresholding and by employing probabilistic weights for the edges of the FCMs, the inference process can now choose the statuses of the action variables. Then, it can achieve – according to the knowledge contained in the FCM – the best result, even when a trade-off is mandatory. In Figure 9, the FCM contains two quality concepts named *(Q) Concept 2* and *(Q) Concept 3*. Its adjacency matrix is

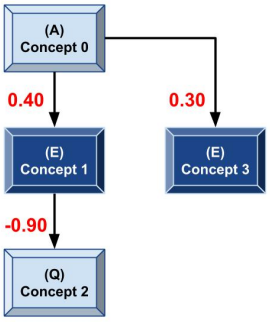
$$\begin{pmatrix} 0 & 0.4 & 0 & 0.3 \\ 0 & 0 & -0.9 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{8}$$

By looking at the causal relationships among the concepts, it is easy to understand that it is impossible to satisfy both the constraints due to the quality parameters (regardless of the status imposed to the action variable). These situations are very common in real world, when actions can frequently have positive consequences but also unwanted ones. However, a decision has to be made: in this example, the decision is *do not act*, thus the action variable remains zero, and maintains the variable *(Q) Concept 2* in an alert state. Then, the system has decided to fulfil the request of the latter variable, despite the second quality concept.

The decision-making process, implemented in the inference algorithm, chooses for the action that has the higher probability to lower the value of at least one quality variable. This result may seem controversial because no relevance has been considered for any of the two quality concepts. Moreover, in this scenario, the system will keep changing the value of the action variable in a continuous loop.

According to the authors, there are possible solutions for this problem, and an appropriate choice has to be found according to the specific system requirements. First, the system can find a solution by noting that quality variables are treated in the same manner: in particular, let's differentiate among quality variables by introducing a weight parameter, associated to each of them. Thanks to these parameters, the inference process can now assign the scores to each solution. In this way, the FCM can provide a higher degree of priority to a quality parameter with respect to the other. However, this simple approach it requires a-priori knowledge since these weight coefficients have to be manually set.

The second solution should be the consideration of fluctuations in the response of the system due to the inference process, as a special feature rather than a limitation. This intrinsic behaviour is caused by the system, which tries to satisfy the request of the first quality concept, at a price of an undesired state for the second quality concept. So the request



(a)

	Action vectors / Inference process status (steps)				Valid
	(A) Concept 0	(E) Concept 1	(Q) Concept 2	(Q) Concept 3	
Action vector 1	-1	0	1	0	/
Step 1	0	-0.4	0	-0.3	YES (score = 0.30)
Step 2	0	0	0.36	0	NO
Step 3	0	0	0	0	NO
Step 4	0	0	0	0	NO
Action vector 2	1	0	1	0	/
Step 1	0	0.4	0	0.3	NO
Step 2	0	0	-0.36	0	YES (score = 0.36)
Step 3	0	0	0	0	NO
Step 4	0	0	0	0	NO
RESULT	Action "(A) Concept 0" must change status to "High"				

(b)

Fig. 9. Example that shows the presence of multiple quality parameters in the FCM (a) Example of an FCM, in which, not thresholding and employing probabilistic weights, allows the inference process to find the best trade-off for the action variable (b) Description of FCM inference process.

of the second quality concept is satisfied but the result becomes that the first quality concept will go back to a bad condition.

Nevertheless, the FCM with the proposed new characteristics can still keep a certain physical magnitude between two bounding levels. Let's suppose a low value of the latter mentioned quantity triggers the high state of the first quality parameter, while a high value would have activated an alert state of the second one. Our implementation maintains this value between these two boundaries. In fact, given these two alerting situations, the system reacts by activating/deactivating the action variable, which can invert the current trend of the physical magnitude. An additional important effect of this novel proposed approach is also the capability to discover the most appropriate action by trials and errors, so that it can operate without any a-priori knowledge.

C. Combination of Different Paths

In the previous examples, we saw that – when there are two paths from an action to a quality concept – the system decides according to their relevance: in particular, this relevance is related to the probability that an action will generate the desired effects.

At this point, it may be useful to wonder if it is possible to improve the decision procedure of Figure 8. Actually, that can be achieved by combining the two possible paths instead of singularly selecting them by adding up their weights to get a more accurate decision. Moreover, it is important to prove whether the standard inference process already performs the combination of the contributions coming from different paths and, in particular, if such behaviour happens regardless the configuration of the FCM.

In Figure 9, the FCM singularly considered different paths. Nevertheless, Figure 11 and Figure 10 show that this may not be always the right method and, more important, it cannot be possible to define a-priori which one of the two methods a generic FCM has to use.

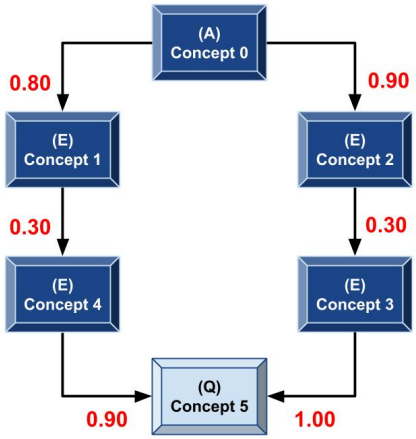
Figure 11 and Figure 10 represent two FCMs, which slightly differ in their own structure: the first has unequal length (expressed as number of hops, concept by concept from the action to the quality variable) of the two chains, while the second has equal lengths. Given the assumption that the numbers of nodes and edges should not affect the decision process methodology, if the two cases are similar then the outcome should be equivalent.

By looking at the logs of the inference process (right side Figure 11 and Figure 10), it is possible to see the previous assumption has not been satisfied: in fact, the final scores are different. It is also worth notice that the score in the second case (Figure 10) could have also been greater than one, if only the values on the edges where slightly different. These inaccurate outcomes cannot be accepted during a reasoning mechanism of a cognition system, thus a solution has to be found.

In the first case (Figure 11), the two causality chains are asymmetric (i.e. they do not enter in the final node at the same time): this leads the system to consider the two contributions separately, and to keep the most convenient one. However, in the second case (Figure 10) – since the two causality chains are symmetric – they infer on the quality parameter during the same step, hence resulting in the combination of their contributions. That means the two probabilities, coming from the end of the two causality chains, are added up when they enter in the final node (the quality parameter).

Each case (Figure 11 and Figure 10) shows a different example of reasoning error: in the first case, the contribution of the least significant path is not considered due to the chain asymmetry, while, in the second, both contributions are combined thanks to the chain symmetry, but the final value comes from the simple addition of these two contributions.

Even if the values propagating in the map during the execution of the protocol are considered as probabilities, such a behaviour has to be expected since the inference process is based on vector-by-matrix multiplication. This problem, which has been discovered, comes from the union of two



(a)

	Action vectors / Inference process status (steps)						Valid
	(A) Concept 0	(E) Concept 1	(E) Concept 2	(E) Concept 3	(E) Concept 4	(Q) Concept 4	
Action vector 1	-1	0	0	0	0	1	/
Step 1	0	-0.8	-0.9	0	0	0	NO
Step 2	0	0	0	-0.27	-0.8	0	NO
Step 3	0	0	0	0	0	-0.99	YES (score 0.99)
Step 4	0	0	0	0	0	0	NO
Step 5	0	0	0	0	0	0	NO
Action vector 2	1	0	0	0	0	1	/
Step 1	0	0.8	0.9	0	0	0	NO
Step 2	0	0	0	0.27	0.8	0	NO
Step 3	0	0	0	0	0	0.99	NO
Step 4	0	0	0	0	0	0	NO
Step 5	0	0	0	0	0	0	NO
RESULT	Action "(A) Concept 0" must change status to "Low"						

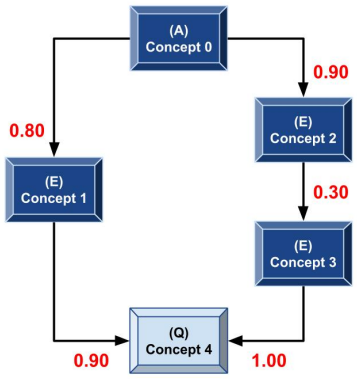
(b)

Fig. 10. Example to show the advantage of combining different paths (a) Example of an FCM, which has unequal lengths (b) Description of FCM inference process, which shows that values are erroneously added.

contradictory concepts. First, FCMs are supposed to be simple and their knowledge easily understandable by humans, and the vector-by-matrix multiplications necessary in the inference process has become one of the most known feature, which is supposed to represent the simplicity of this approach. On the other hand, a cognition process has to be able to fully exploit the stored knowledge in order to perform actions [12].

Then, we can understand that the problem lies in the nature of FCMs. In order to correctly adapt FCMs to solve this issue, we need to introduce some modifications in the inference process implementation:

- If n branches, leading to the same concept, are not composed of the same number of nodes (asymmetric



(a)

	Action vectors / Inference process status (steps)					Valid
	(A) Concept 0	(E) Concept 1	(E) Concept 2	(E) Concept 3	(Q) Concept 4	
Action vector 1	-1	0	0	0	1	/
Step 1	0	-0.8	-0.9	0	0	NO
Step 2	0	0	0	-0.27	-0.72	YES (score 0.72)
Step 3	0	0	0	0	-0.27	YES (score 0.27)
Step 4	0	0	0	0	0	NO
Step 5	0	0	0	0	0	NO
Action vector 2	1	0	0	0	1	/
Step 1	0	0.8	0.9	0	0	NO
Step 2	0	0	0	0.27	0.72	NO
Step 3	0	0	0	0	0.27	NO
Step 4	0	0	0	0	0	NO
Step 5	0	0	0	0	0	NO
RESULT	Action "(A) Concept 0" must change status to "Low"					

(b)

	Action vectors / Inference process status (steps)					Valid
	(A) Concept 0	(E) Concept 1	(E) Concept 2	(E) Concept 3	(Q) Concept 4	
Action vector 1	-1	0	0	0	1	/
Step 1	0	-0.8	-0.9	0	0	NO
Step 2	Holding the evaluation of the left branch (-0.8)					NO
	0	-0.8	0	-0.27	0	
Step 3	0	0	0	0	-0.7956	YES (score 0.7956)
Step 4	0	0	0	0	0	NO
Step 5	0	0	0	0	0	NO
Action vector 2	1	0	0	0	1	/
Step 1	0	0.8	0.9	0	0	NO
Step 2	Holding the evaluation of the left branch (0.8)					NO
	0	0.8	0	0.27	0	
Step 3	0	0	0	0	0.7956	NO
Step 4	0	0	0	0	0	NO
Step 5	0	0	0	0	0	NO
RESULT	Action "(A) Concept 0" must change status to "Low"					

(c)

Fig. 11. Example to show the advantage of combining different paths (a) Example of an FCM, which has unequal lengths (b) Description of FCM inference process,which does not wait the outcome of the longest path (c) Description of FCM inference process showing that inference always finds optimal result.

paths) then, during the inference process, inference on the shorter path has to wait until the one on the longest has arrived.
- When n concepts lead to the same node, their contributions to the final value have to be treated like probabilities, i.e. they cannot be simply added up. Hence, the shared target node is inferred with a probability, calculated as the sum of independent random variables.

The first condition imposes that – regardless of the concept class type – when there is more than one arrow entering a node, a decision for the final value of this variable only has to be computed when all the contributions – coming from other nodes – are available. The second condition instead is related to how these contributions have to be combined together: as discussed before, the values propagating on the edges during the algorithm execution are treated like probabilities, so that the sum is not allowed because it could possibly lead to values out of $[0, 1)$.

As an example, let's consider two causality chains, reaching a certain target node with probability 0.7 and 0.8 respectively. As a consequence, their sum is 1.5, which it may be approximated to 1.0. Then, a causal relationship, represented with the unity value, reflects a fully deterministic connection, which is a nonsense if we consider that the ones closer to the sources are not.

So, the proposed solution overcomes this issue by treating probabilities as they are and by performing the addition accordingly. The sum of independent random-variable formula fits the bill and provides an output in the continuous set $[0, 1)$. It is worth notice that every contribution coming from each causality chain positively contributes to the final outcome. It is also worth to notice that before combining the contributions, the positive and negative causal relationships are separated and only at the end are summed up. In particular, the proposed expression is only used among contribution with the same sign. Formally, given n variables

$$\begin{cases} y[n] = y[n-1] + x[n] - y[n-1] \cdot x[n] \\ y[0] = 0 \end{cases} \tag{9}$$

Referring to examples in Figure 11 and Figure 10, the score variable is now meaningless once there is only one quality variable because all the contributions, coming from different paths, are combined together, thus resulting in a single score value. It is possible to see that the outcomes of the two FCMs are now equal as expected: in the inference process log, it is clearly visible that the implementation waits for the first branch on the left to advance and, once they are ready to enter together in the quality variable, the algorithms proceed with the sum of their probabilities.

D. Causality Loops Removal

If FCMs perform thresholding and map values in the discrete range $[0, 1]$, the inference process can produce matrices with redundant vectors. Then, once the algorithm recognises that the current outcome has already been found in the past steps it just stops. Nevertheless, by avoiding thresholds and by interpreting weights of edges as probabilities, there is the need to propose other solutions to face loops.

First, it would be possible to set a limit in order to determine whether the inference process has to continue through the chain of concepts or it has to stop: in particular when the absolute value of the probability reported along a chain of causal relationships is smaller than the threshold, the latter action has to be performed. Moreover, the value of this limit should be close to zero to allow the system to explore possible solutions when weak knowledge is present on the edges, thus resulting in small absolute values.

The idea behind this first approach comes from the fact that loops in FCMs make the inference algorithm spinning continuously across the involved nodes and this process never stops because a steady solution cannot be found. It is worth notice that cycle after cycle, the output of the inference process converges to an interval close to zero: by definition, if the causality values on the edges are bounded by the continuous set $[0, 1)$, then the absolute value of the output will decrease iteration after iteration. Given that, the introduction of the proposed limit allows the algorithm to indirectly realise when it is stuck in a loop.

The second proposed approach (the one applied in the rest of this work) is based on a preprocessing operation, performed before the inference process starts. During this stage, the cognitive engine runs an algorithm similar to the Spanning tree protocol: that can detect and remove cognition loops from the FCM. This idea is supported by the consideration that, in the matrix representing the knowledge, loops do not supply any additional useful information to the final result of the inference process, thus they can be removed with confidence.

According to this method, there is no need for a limit, which may not be straightforward to be set to an appropriate value. This second approach should be preferred to the first because it avoids useless iteration across the same concepts forming the loop. However, the drawback of this method is a higher implementation complexity and an additional overhead in the inference process.

Figure 12 depicts the FCM before and after the execution of the procedure of loop removal. The adjacency matrix referred to Figure 12(a) is

$$\begin{pmatrix} 0 & -0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{10}$$

Especially, Figure 12(b) reveals that *Concept 3* has been removed from the collections of the considered nodes during the inference process because it does not provide any information since it is part of a loop.

The modifications applied so far on the FCMs' inference process resulted in an augmented complexity of the FCM-based system. The first proposed approach can search faster

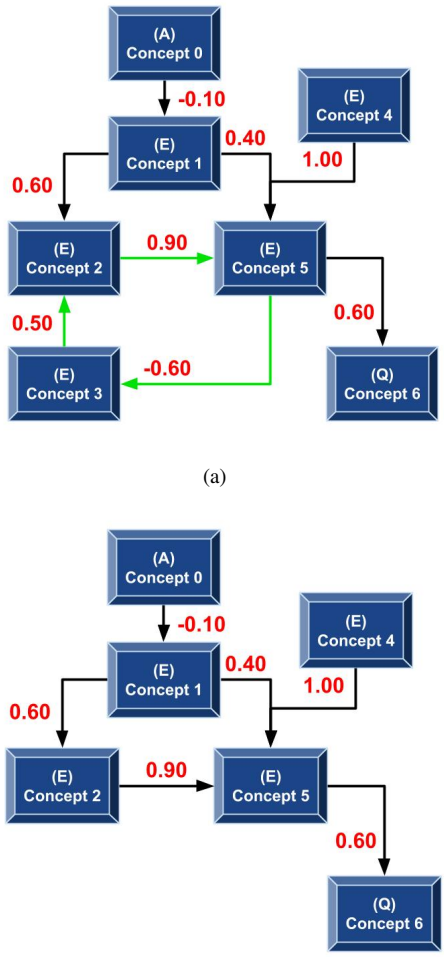


Fig. 12. (a) Starting FCM, before loop removal (b) Ending FCM after loop removal procedure.

and more time-effective compared to the second: actually, the choice of the first would lead to longer and harder results to understand because the output of the inference process would be affected by many irrelevant values, related to the spin of the algorithm around the loop. The system would properly work anyway and the final decision would not change: then, for completeness, both the approaches have been discussed in this subsection.

E. Multiple Action Variables

Besides the entities devoted to reasoning and learning, an FCM-based system also contains two fundamental modules, in charge of performing the transcription of measurements from the real world to the logical one and vice-versa: these units are called sensors and actuators. This subsection will discuss the latter.

In an FCM, there might be more than one variable belonging to the action class: especially, when many causal relationships are missing in the matrix, the decision-making process might have to react to a bad-shaped situation of the system without the possibility to rely on the necessary information. According to the suggestion reported in [12], if no action has been found during the inference process, the system has to randomise the possible actions in order to discover what may lead to any effect. However, there was no indication related to how such a mechanism should be performed.

The task related to the discovery of relationships from the action to the quality variables is particularly critical, mainly because communication networks are dynamic systems rapidly changing, and a certain action may only be effective for a short period of time. By this consideration, it is a fundamental requirement to deploy a system, which can discover the correct set of actions as fast as possible, before the environmental conditions change. An exhaustive search may not be feasible because it could take a long time, since the number of sets increases exponentially with the number of action variables. On the other hand, if each action is tested independently from the others, then the risk is to neglect possible effects, which are only manifested when a combination of actions is examined.

To solve the above issue, this work proposes a novel approach, which contemplates both the previous considerations. The main idea is to select a set of combinations of actions capable to stimulate the system in a relatively small number of steps. That is inspired by how humans perform when they do not have any knowledge about the functions of a certain device, so they can solely rely on their common sense. For example a typical behaviour of a user that does not know how to use a radio, is to push many buttons at the same time until he is able to achieve the desired goal. The user is aware that he will only need to press few buttons in order to reach his objective. However, since he does not know which ones belong to the correct combination, he will continuously try different possibilities.

According to the aforementioned requirements, there is the need to operate quickly over the network through actions in order to discover new causal relationships. The formula used in this article to generate k sequences of actions when in the system there are N action variables is

$$y_{k,n} = (-1)^{k+\lfloor c \rfloor} \tag{11}$$

where $c = \frac{n-1}{N} 2^{\lfloor \frac{k}{2} \rfloor}$, $1 \leq n \leq N$ and $1 \leq k \leq \lfloor 2(\lceil \log_2 N \rceil + 1) \rfloor$. In particular, it represents the starting point, which is used by the cognitive engine to generate action matrices to determine and to set the initial value of each action variable.

This formula has been derived from the generic family of discrete transforms; in our scenario, the one-dimensional sequences at different frequencies are treated as scrambled action patterns. Actually, the authors have proposed this formula because of its interesting following properties:

- half of the generated sequences are equal to the other half but inverted, and there are always at least $\lfloor N/2 \rfloor$ active actions at each iteration to boost the action impact on the system;
- regardless of the position in the actions array, every iteration each action has the same chance to be activated;
- the number of action sequences grows logarithmically with the length of actions array.

Next, expression (11) produces the output matrices in Figure 13, which are particularly effective to train different combi-

-1	-1	-1	-1
1	1	1	1
-1	-1	1	1
1	1	-1	-1
-1	1	-1	1
1	-1	1	-1

-1	-1	-1	-1	-1
1	1	1	1	1
-1	-1	-1	1	1
1	1	1	-1	-1
-1	-1	1	-1	1
1	1	-1	1	-1
-1	1	1	-1	-1
1	-1	-1	1	1

Fig. 13. Matrices to numerically derive the formula to generate k sequences of actions when in the system there are N action variables.

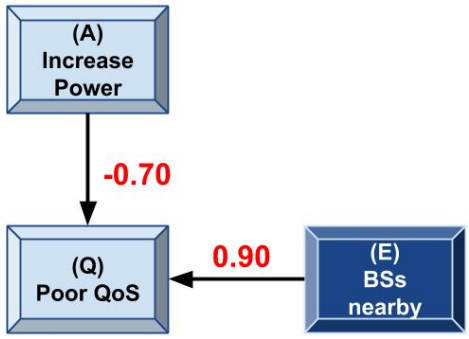
nations in a small amount of iterations. As just discussed, the number of steps is now logarithmic hence the system can be probed in reasonable amount of time. In particular, it is bounded by a range that is larger compared to the case where each single action is singularly tested, but smaller compared to the case of exhaustive search, which is an NP-hard problem. The performance achieved by the cognitive SDN system presented in Section IV are satisfactory, using relation (11). However, the training of proposed FCMs may be further optimised and improved: that is out of the scope of this work. Further methods of FCMs' training can be found in [11].

F. Environmental Variables as Blocking Conditions

As previously explained in Subsection III-E, it is possible to have more than one action variable, for which different combinations are tested in order to discover the best set of action values, according to the knowledge stored in the map. However, in real world performing and acting or, more generally, changing the value of a variable has an impact and a cost.

Let's suppose the controller of a cellular base station decides to increase the output power of an antenna to provide a better service for a set of users. That choice can be made because in that particular area there are many customers, who are consuming services characterised by a high demand of bandwidth. Anyway, the situation is hardly sustainable by the serving base stations because it is not possible to deploy modulations with higher spectrum efficiency, due to the distance. As a consequence, the FCM-based system can enter in a bad-shape situation, and the inference process can be run to find a solution. A proposed action may be to increase the output power of the antenna because the system has learnt in the past that it is possible to improve the QoS by increasing the feeding current. Nevertheless, when a base station operates with more power it can provide a better service to the users, but there is also the risk that this increment will interfere with other base stations located nearby.

Figure 14 displays an FCM, which already yields this knowledge, represented by a number on the relative edge, whose absolute value is particularly high: that means the system is aware that the increase of output power through an antenna leads to interference very often. Figure 14 and Figure 15 report the outputs of the two executions of the inference process. In the first case, the blocking environmental concept *BSs nearby* is not active, representing a scenario where there



(a)

	Action vectors / Inference process status (steps)			Valid
	(A) Increase Power	(E) BSs nearby	(Q) Poor QoS	
Action vector 1	-1	0	1	/
Step 1	0	0	0.7	NO
Step 2	0	0	0	NO
Step 3	0	0	0	NO
Action vector 2	1	0	1	/
Step 1	0	0	-0.7	YES (score 0.7)
Step 2	0	0	0	NO
Step 3	0	0	0	NO
RESULT	Action "(A) Increase Power" must change status to "High"			

(b)

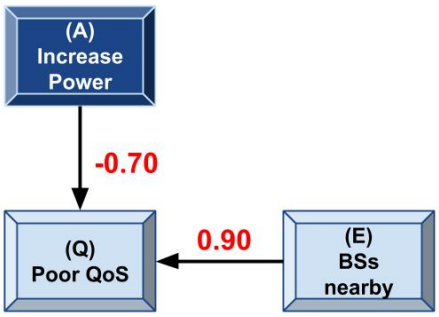
Fig. 14. Inference process where the blocking environmental concept *BSs nearby* is not active (a) Fuzzy cognitive map (b) Description of FCM inference process.

are no neighbouring base stations: thus it is possible to increase the power, moving the system to a safe condition where the state of the variable *Poor QoS* is low. On the other hand, the second case has a high environmental parameter (see the log), so the inference process stops and communicates to the system that it is better not to act. Indeed in the last scenario, the *Poor QoS* alert state of the variable could not have been resolved by acting since it would be resulted in a worse situation.

G. Negative Nodes Values

Despite significant part of the literature referred to FCMs, this work proofs that adding negatives values to concepts can boost the performance of the system. Instead of range $[0, 1]$, the idea is to use the discrete set $[-1, 0, 1]$, particularly suitable for modelling concepts, for which low states (zero values) yield a double meaning.

According to the authors, the zero value should be assigned to a concept when its state is inactive. Next, a negative value should be acquired by the variable when a concept has an



(a)

	Action vectors / Inference process status (steps)			Valid
	(A) Increase Power	(E) BSs nearby	(Q) Poor QoS	
Action vector 1	-1	1	1	/
Step 1	0	0	0.2	NO
Step 2	0	0	0	NO
Step 3	0	0	0	NO
Action vector 2	1	1	1	/
Step 1	0	0	0.2	NO
Step 2	0	0	0	NO
Step 3	0	0	0	NO
RESULT	No valid solution can be found			

(b)

Fig. 15. Inference process that takes into account neighbouring base stations as blocking environmental concept (a) Fuzzy cognitive map (b) Description of FCM inference process.

effect on the system, so when its influence is the inverse of the high state case. As an example, let's suppose there is a positive causal relationship between the concepts *Financial Crisis* and *Poverty*, where the first is the cause and the second is the effect. The inference process of the FCM-based system will reveal that, if the cause is in high state, then we can expect the propagation of this status to lead to a high state also in the poverty concept; on the other hand, when the *Financial Crisis* variable takes the low value then the *Poverty* concept should acquire the same status as well.

This example conveys that the zero-valued concept may often be related to the fact that the economy is in a steady-state condition, thus we cannot claim anything about poverty; on the other hand, low *Financial Crisis* value may specifically represent a growing economy, which has an active negative impact on *Poverty* concept. In other words, in particular scenarios, it is important to distinguish between an inactive and a negative-state of a concept otherwise it would not be possible to accurately model the parameters of the system.

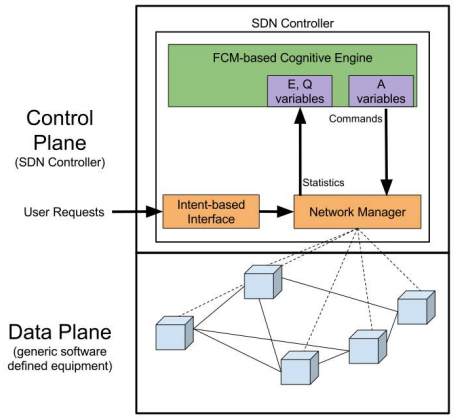


Fig. 16. Reference architecture for Cognitive SDN based on FCMs.

IV. A Cognitive Software-Defined Networking Architecture

The SDN concept of centralising the control plane and separating it from the data plane [4] sets up fertile ground in order to effectively include the cognition processes in the networks. This section clarifies why it has been decided to locate cognition processes in the control plane of SDN. Figure 16 depicts the reference architecture of the proposed cognitive SDN architecture.

By considering cloud-computing vision, system management is centralised: therefore, the idea is to exploit this characteristic of SDN by placing an FCM-based *Cognitive Engine* over the SDN controller. In this way, the cognitive module gains direct access to all network information processed by the network manager, hence allowing learning and acting through a unique interface. The network manager's interface with the FCM-based cognitive engine is used to exchange network statistics, and commands to update network characteristics.

Recent developments in SDN theory have shown the need of designing intent-based interfaces, for exposing network functionalities to the upper layers. An intent-based approach starts from the idea that a tenant should not obtain both network details and access to the controller; nevertheless, a tenant should only specify what the scope is, and then the controller should perform effective operations to achieve the requirements.

This approach has been widely adopted by OpenFlow controllers [37]. Among those, it is particularly interesting the work that has been performed on the ONOS controller [38], which contains an intent-based interface that is capable to perform advanced reasoning operations.

The controller of the proposed cognitive SDN system has an intent-based interface to implement 'controller neutrality'. In fact, users' requests represent the input requirements from the tenant. The intent-based controller is capable to translate (compile) intents into data plane rules according to the current network conditions: basically, once it retrieves the intention from the user, it tries its best to accomplish the designated

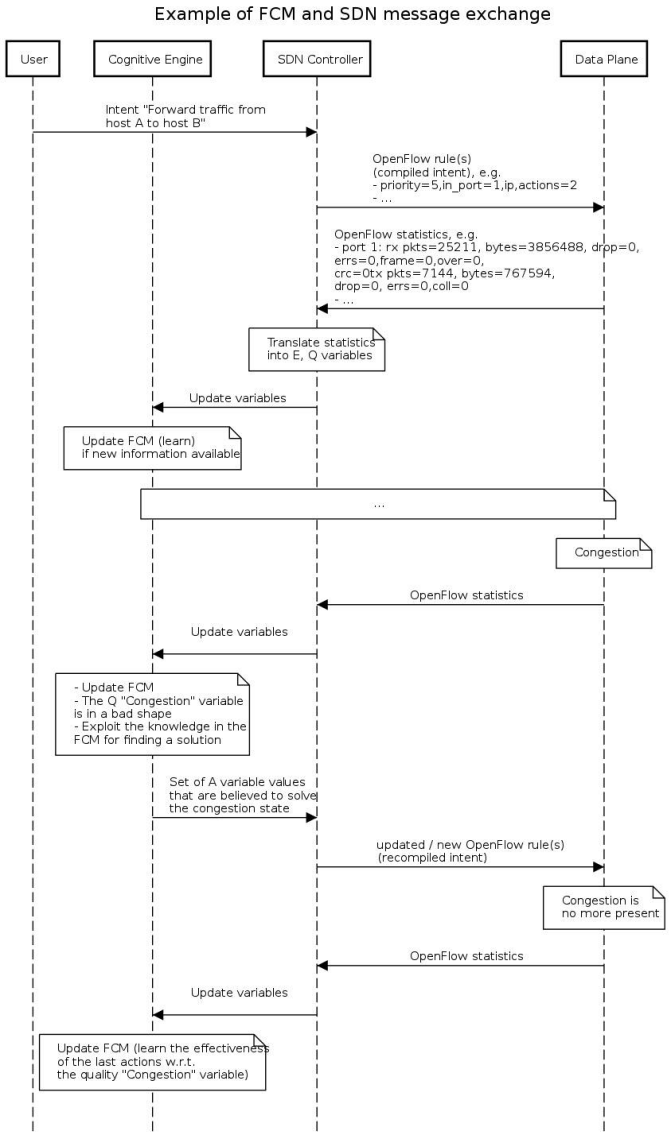


Fig. 17. Example of signalling between the FCM-based cognitive engine and SDN (controller and data plane).

task, and it manages to react to the events in the network (e.g. broken link) in order to maintain the intent alive. Especially, in the latter case, the controller recompiles the intent in order to adapt the corresponding actions on the network to the current environmental situation. These operations cannot be performed by following a rigid analytical model of the controlled system, since very often the control logic can rely to little a-priori knowledge, and the effectiveness of certain actions strongly depends on the infrastructure characteristics and the operating context, that have to be empirically discovered.

Figure 17 shows an example of signalling between cognitive engine and SDN system. The process starts with the user, who sends the requirements to the SDN controller. Next, SDN controller updates OpenFlow tables at the data plane. The data plane sends network statistics to the SDN controller, which are interpreted and converted into variables for the FCM. Then, these values are passed to the FCM-based cognitive engine, which uses them for the learning/decision process. Suddenly, a congestion happens. This event provides different statistics to the SDN systems, which are given to the FCM to be analysed. That results in further learning and subsequent autonomous adaptation to the new network conditions. The last part of the signalling has the same structure but shows the procedures applied when congestion is solved.

A. Latency and Overhead

In order to react to network events, a certain amount of time is needed for the cognition process to find a solution: this can be split in

- the time required for the messages exchange between the cognitive engine and the controller, plus the time required for the inference process to complete,
- the time that has to be waited because the cognition process is still evaluating whether the application of certain actions are having any effect in solving an alarm state.

In the first case the FCM has already enough knowledge for applying a set of actions in response to an alarm state, the reaction time of the cognitive engine is usually very short and only depends on the computation and message exchange capabilities of the system.

The latency overhead due to message exchange strongly depends on the application-specific design choices. In the most general case, each variable update (cognitive engine input) and action (cognitive engine output) requires one message exchange with the network controller; in case of large FCM maps some optimizations could be introduced, for example by performing the threshold comparison of the measurements in the controller and notify the cognitive engine only when a change on status has occurred.

Regarding the computational overhead, there are two main activities that are performed by the cognitive process: learning and reasoning; the following measurements are aimed at showing the impact of these tasks to the overall system performances. It is worth to report that these measurements have been obtained with a non-optimized cognitive engine implementation, meaning that these absolute latency values can be further reduced with proper implementation choices. From performance perspective, the goal is to discover how the proposed system scales when more complex network scenarios with many variables are considered. Nevertheless, the order of magnitude of these values can be considered suitable for most networking scenarios, as these never exceed few milliseconds with a 100-variable FCM with many causal relationships.

Figure 18 depicts the measured amount of time required for completing the FCM update process, by considering also the size of the map. The graph shows three boxplots related to the time spent for the update for three FCM sizes, from a small FCM with only 10 variables to a more complex with 100. The boxplot representation has been chosen because it allows to show in a compact manner statistical information of a set of data (in this case 160 samples for each FCM size has been used), obtained by feeding the cognitive engine (the actual implementation) with random variable changes, in order to simulate a complex network with many events. As

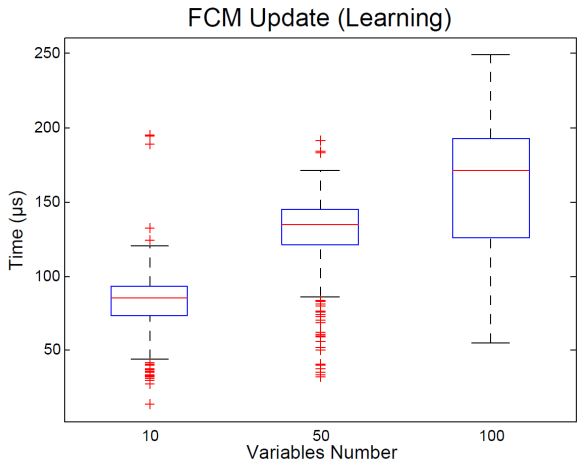


Fig. 18. Time needed for the FCM updating process by considering the size of the map. The boxplots are related to the time spent for the update for three FCM sizes, from a small FCM with only 10 variables to a more complex one with 100. On each box, the bottom and top blue edges of each box indicate the 25th and 75th percentiles respectively, the central red mark indicates the median, and the two horizontal bars comprising the box are the minimum and maximum values of the range of samples.

		FCM Size = 10 Max links = 56		FCM Size = 50 Max links = 1389		FCM Size = 100 Max links = 5556	
Number of links	Percentage of links w.r.t. FCM size	10	**18%**	200	**14%**	1000	**18%**
		30	**54%**	700	**50%**	3000	**54%**
		50	**90%**	1200	**86%**	5000	**90%**

Fig. 19. At the top of the table, the maximum amount of cognitive relationships is reported for each FCM size. Then, these values are used as references for comparing them with three percentage levels of FCM link filling.

previously stated, the main information obtained is related to the increase of processing time when the FCM grows; results show that time spent for the computation has approximately doubled with a FCM size change from 10 to 100 variables, proving that the updating process can scale well.

As results show, these operations does not demand high resources, since the update process only involves one variable status change and others that happened within the same persistence time interval. Indeed, the reason behind the increase of processing time with the FCM size is related to the higher number of causal relationships that are usually present.

Different from the learning process is the reasoning one. The exploitation of the available information contained in the FCM requires the execution of the inference process which however, from the computational perspective, does not contain any exponential algorithm, meaning that it is expected to scale.

Figure 20, Figure 21 and Figure 22 report the time required to find a suitable solution to an alarm state by considering both the number of FCM variables and the number of causal relationships. For these tests, it has also been considered the number of causal relationships, since it proves to impact the final performance measurements. Each graph is related to a fixed number of variables (from left to right, 10, 50, 100 respectively, with 20 samples per box) where the time spent for

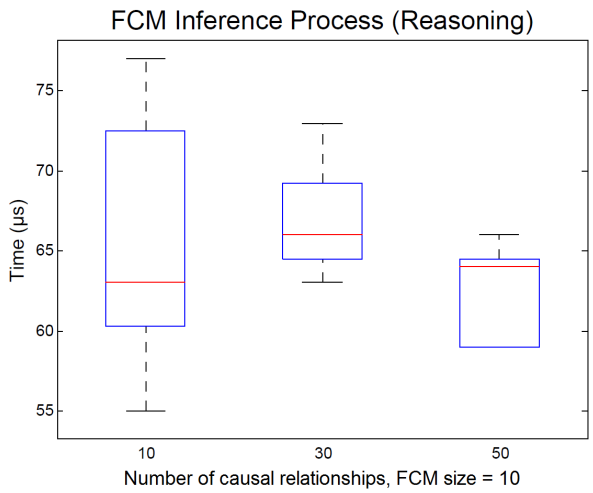


Fig. 20. Time needed for the FCM reasoning process by considering the number of causal relationships, due to the small FCM size (10) there is no relevant differences in the computational time.

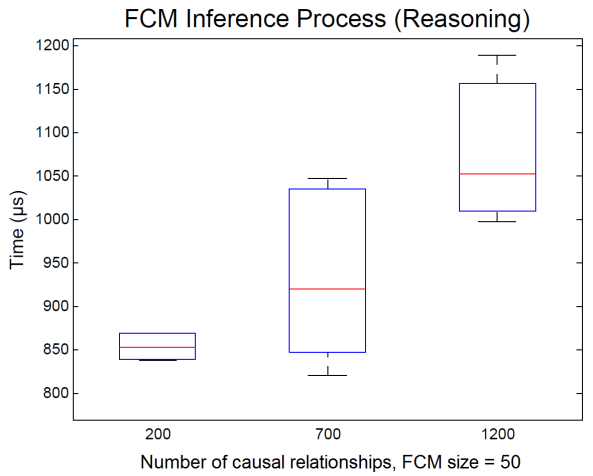


Fig. 21. Similarly to Figure 20 it has been reported the processing time required for different amounts of causal relationships, because of the greater FCM size (50) their values show some small differences among them.

the computation is reported for different amount of information stored in the FCM. For each FCM size, the maximum number of causal relationships that can be established changes: it is worth to notice that this value is given by $N^2 \cdot 5/9$ because not all the causal relationships are stored where N is the FCM matrix size (or number of variables). For example, it is not possible to establish a causal relationship between two action variables. Table 19 helps to map the absolute number of links into the corresponding percentage of FCM information; these values have been chosen in order to allow a fair comparison, since for each FCM size it is reported the time spent for the inference process when the FCM is full at 20%, 50% and 90% approximately. The obtained results show that the proposed reasoning technique is capable to provide action plans in short amounts of time, which is aligned with the one reported by ONOS.

Figure 23 shows the time spent by the inference processes for different FCM sizes and different amounts of learnt

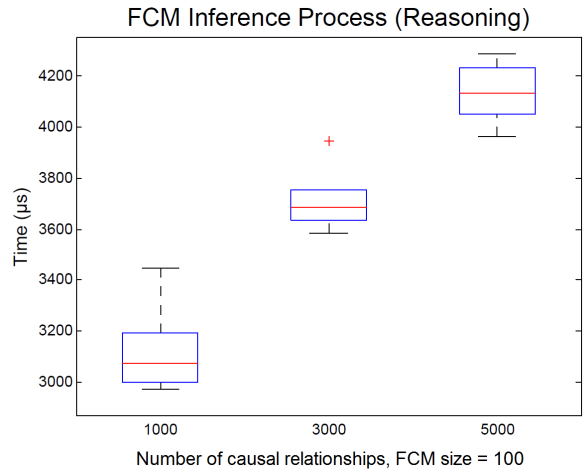


Fig. 22. With the greatest reasonable FCM size (100) the differences in the processing times depending on the number of causal relationships have increased with respect to Figure 20 and Figure 21. However, it is worth to notice that these have never exceeded the half millisecond, proving the suitability of the proposed approach.

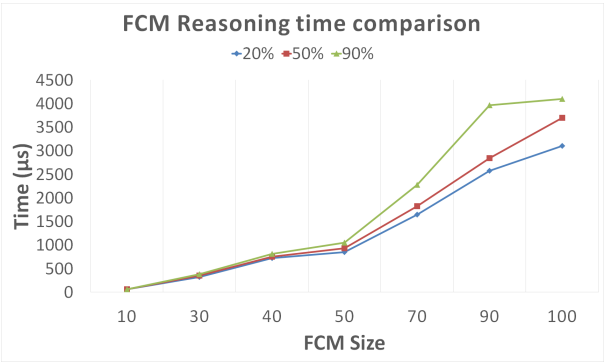


Fig. 23. Time spent by the inference process, considering both different FCM sizes and different amounts of learnt relationships.

relationships. The results reveal that there is a small time difference between an almost-empty FCM and an almost-full one, only for large FCMs; on the other hand, smaller FCMs shows negligible difference in time spent.

Figure 24 depicts the comparison between the latency due to legacy ONOS-SDN networks with different characteristics and the additional overhead (in time) due to set up of new proposed FCM-based SDN system (with different FCM sizes). This temporal analysis comes from the fact that 5G and beyond 5G networks will have latency as a main requirement: thus a cognitive SDN framework, managing a virtual network, should not affect significantly the latency of legacy SDN systems. The picture clearly shows how negligible the overhead (additional delay) of the FCM inference process is with respect to operations in legacy SDN networks. For example, with a large 100-variables FCM, the impact on the delay for setting up an end-to-end flow through 300 switches is of (3.63 ms / 3273 ms) $\approx 0.11\%$, meaning the overall time required by the controller to act on the network is less than 3277 ms.

The goal of comparing order of magnitudes is to show that the proposed FCM-based SDN paradigm introduces negligible temporal overhead when the Cognitive Engine is implemented

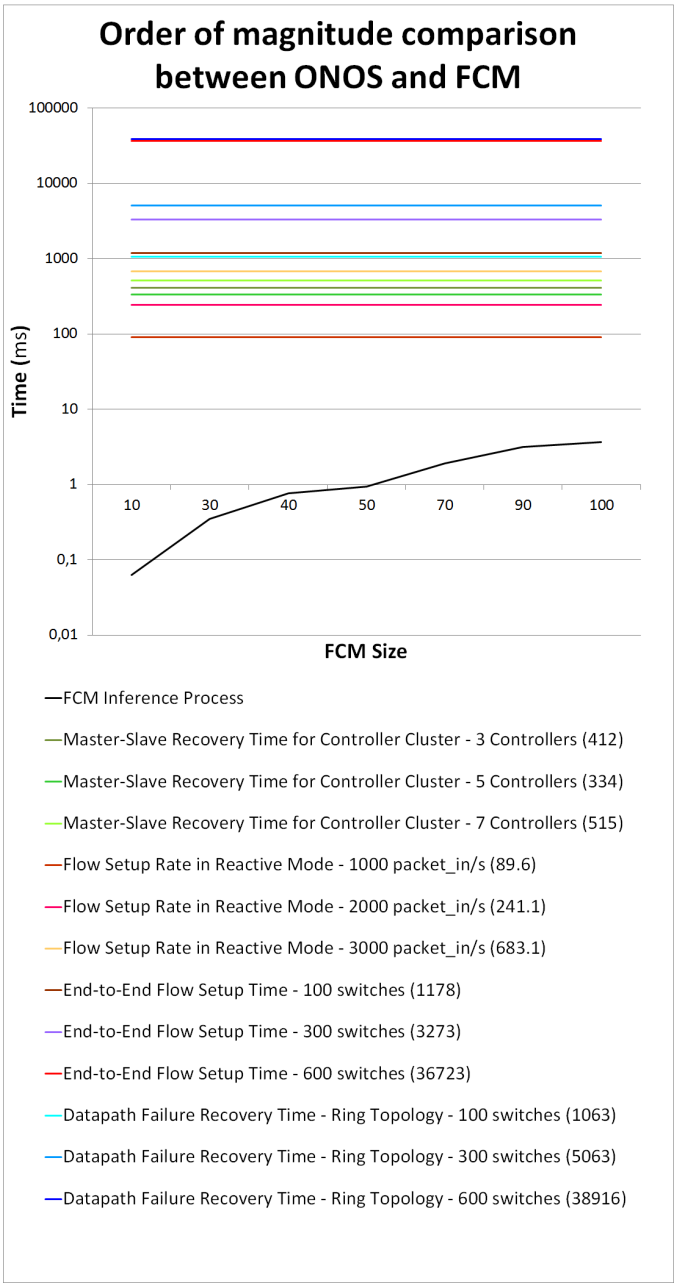


Fig. 24. Representation of delay of legacy ONOS-SDN systems and of the additional latency introduced by our Hebbian-based FCM algorithms. The comparison of their orders of magnitude shows that our FCM solution introduces negligible delay in legacy ONOS-SDN networks. The horizontal axis considers the changing size of FCM, thus it is referred to FCM line (other ONOS-referred lines remain constant). In order to increase the readability of the picture, in the legend, the respective values of vertical axis are reported between round brackets.

on top legacy SDN controller. Indeed, the proposed approach is not meant to replace existing SDN solution, but rather to add new more effective cognitive functionalities. Figure 24 reports the average processing time required by the FCM inference process, and other time-based measurements obtained from some ONOS performance tests [39]. By looking at the picture, the impact of FCM operations on ONOS-SDN is in the order of $[10^{-4}, 10^{-2}]$ ms (for FCM size = 100) and in the order of $[10^{-6}, 10^{-4}]$ ms (for FCM size = 10). It is worth to notice that FCM-based performances depend on factors such as the FCM size and the number of relationships, while the delay of ONOS-SDN controllers mainly depends on SDN-specific tasks, where network latency between controllers and switches plays a relevant role. The same horizontal axis on the graph is referred to FCM size and only affects the curve related to FCM-based SDN. In fact, setting up an OpenFlow rule through a long path of switches has no correlation with having a large FCM, since a single action variable on the FCM can trigger the actuation of many configuration at a time. It is also important to remember that absolute delay times strongly depends on the implementation choices and optimisations: from this perspective, with respect to the software that has been developed for validation, there is room for improving the implemented FCM-based solution, which can lead to a further reduction of processing time.

B. Learning and Reasoning

Both sensing and acting operations require accurate design choice in order to correctly translate the quantities from logical to real world; this stage is particularly critical because it deeply affects system performance. At this point, we can start analysing how an FCM-based system implementation should work with the concepts and their changes of status, from the theoretical point of view of Differential Hebbian Learning. When it comes to implement such approach, other parameters have to be introduced. The first question that has to be answered is "if A changes its status, how long should the system wait for event(s) X(es) for considering it(them) the effect(s) of cause A"? The correct interval time Δ depends on the application, some a-priori knowledge about the reactivity of the system may help the designer in choosing the more appropriate value, otherwise this parameter should be set after some trials: if no edges are built in the FCM when the supervisor is aware that concepts has changed their status, then a higher Δ value should be chosen. Figure 25 reports the scheme of implementation of the cognitive process: in particular the diagram shows that, apart from the learning process, represented in the first block, the system is triggered when events related to quality concepts enter in a bad shape condition i.e. there is an alarm. The Φ time is related to the persistence of the selected action(s): especially, in order to detect whether an action(s) has(ve) been successful, it is mandatory to wait for a certain amount of time in order to allow the effects of such an action to happen. Usually it is better if the Φ value is smaller than the Δ time, because otherwise it would not be possible for the FCM to connects the effects to the actions. Processing the events maintaining

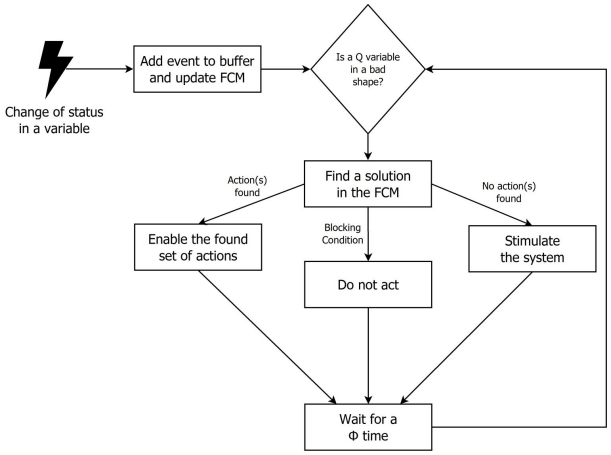


Fig. 25. Scheme of the implementation of the cognition process.

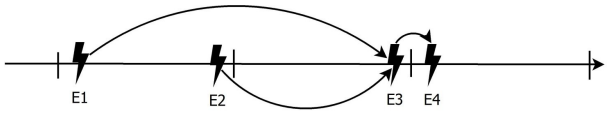


Fig. 26. Differential Hebbian Learning behaviour.

their asynchronous nature, i.e. as they occur in the real world, allows the system to be more reactive to the changes and thanks to this approach it is possible to provide the ability to find the whole causal relationships happening among the concepts. On the contrary, with the previous approach, some of these relationships would have been lost. In Figure 26 and Figure 27, a comparison between the two systems is provided: in the old method, it is clearly visible that the relationships between events E1 and E2 has not been considered. Instead the causality connections between the couples E1→E3 and E3→E4 have been registered even though. It is present a large duration difference between the two cases (the first lasted much longer than the second). With our novel approach instead, as we can see in Figure 27, two events are considered cause-effect when the difference of their timestamps is smaller than Δ time and greater than $\mu \cdot \Delta$, with $0 < \mu < 1$. The parameter μ is mandatory in order to prevent the enforcement of false causal relationships, indeed if two events occurred very close one to respect of the other: then it is likely that they have to be considered contemporary meaning that there is no relations between the two. In the example, the causal relationships E1→E2 and E2→E3 have been registered since the duration of each of them is shorter than Δ and longer than $\mu \cdot \Delta$.

Finally, the delay is related to the learning coefficient and

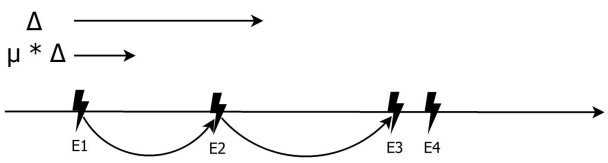


Fig. 27. Behaviour of our new FCM-based SDN method.

the persistence timer: in order to learn new causality relationships, the cognitive engine has to wait for events to happen within a temporal window and for reacting it has to wait for discovering whether the selected actions set is effective. An excessive reduction of the values of these parameters prevents the cognitive process from establishing causal relationships, or it leads to the establishment of false ones. The choice of the most appropriate parameters is necessary to tune properly system performances while a trade-off between reactivity and correctness has to be found. These parameters and their impact are expressed in Figure 19.

C. Number of Network Variables

In the cognitive engine, variables represent the network environment that has to be managed; the number of variables to be defined has an impact on the reasoning granularity, which eventually affects the system performances. A small number of variables prevents the cognitive process from discovering more subtle relationships, leaving only the most trivial ones (for example, a system with only "Move users" and "Congestion" variables would not provide information about other aspects, for example the user distribution and the network load).

On the other hand, many variables could be hard to define and could possibly overload the cognitive engine with reasoning operations with small significance; for example, a concept that has no correlation with the considered network environment should be discarded. There is not a particular bound on the number of variables in an FCM, however, the authors propose to start with a large number of variables, and then remove the ones that have not been part of any causality relationship. In any case, there is the indication that a large FCM is composed by "dozens of concepts", meaning that the maximum size of 100 variables, contemplated for the performance evaluation, is close to the highest reasonable number.

D. Convergence

The proposed cognitive engine, with its capability to build new knowledge and change it according to network conditions, requires a specific definition of convergence. The learning process can be considered successfully completed when, for the same input provided to the cognitive engine, there is a set of optimal actions that can be obtained by simply using the pre-discovered knowledge: these actions lead to the same outcome that was already learnt. In this condition of "static environment", the causal relationships are enforced and the cognitive process converges. However, the main objective of the proposed work is to design a controlling system capable to react to the dynamics of real-life networks, where some actions are only effective for a certain amount of time, after which new solutions must be found. From this perspective, it is safe to claim that the cognitive process never converges because the controlled network continues to change its characteristics, it is a continuous adaptation to new conditions.

V. Results and Discussions

In order to validate the FCM-based SDN, proposed in Section III, a test network has been developed in Mininet [40],

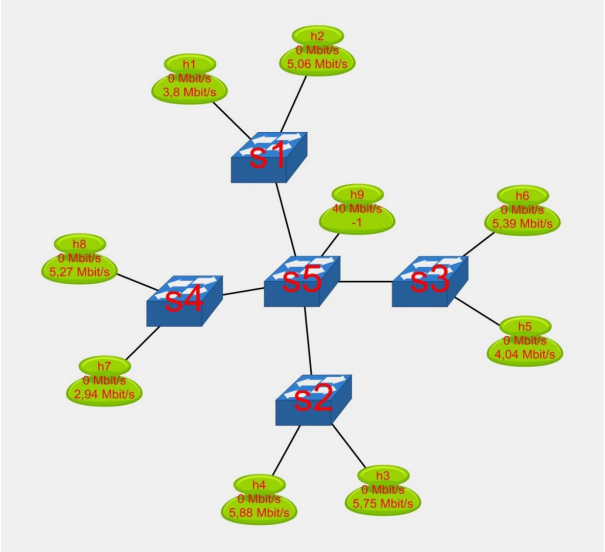


Fig. 28. First scenario to test the performance of cognitive SDN.

composed by data and control plane. The reason to maintain the two components independent – even in the validation process – relies on the fact that the cognitive process can actually work regardless of the controlled infrastructure.

Figure 28 depicts the first testing environment. As stated above, the SDN network is emulated in the virtual machine of Mininet, which exploits the network name-spaces functionality of the Linux kernel and virtualises the full-network stack of many virtual hosts. Such virtual hosts are the nodes of the network: each of them runs a Java program, that connects to the controller via the Control Network. The role of that Java software is to manage the activity of the nodes, i.e. this agent can start an instance of Iperf and can generate traffic towards another node if requested by the controller. Moreover, it can periodically report the current throughput to the controller. Next, given these measurements, the controller can detect whether a congestion state is occurring, i.e. it gets information for the cognitive engine to perform reasoning.

The reason to separate the Experimental from the Control Network, is related to the fact that the experiments, which run on the former network, must not interfere with the service information, exchanged between the nodes and the controller. This design choice is particularly important: let's consider the controller runs tests on the Experimental Network and deeply stresses the network infrastructure, where links are often at their maximum capacity. Meanwhile, the control traffic must not interfere with the measurements related to the experimental network activity.

The whole virtual network is created and managed by the Controller, which is connected to a Python agent that controls the Mininet API, and sets up and configures the actual networks in real time. In particular, in the initialisation stage, the former component configures the network topology on the experimental network. Besides that, the controller is also in charge of leading the experiment, so it can act on the network infrastructure. Furthermore, the controller can both redistribute the users in the network (changing the switch to which users

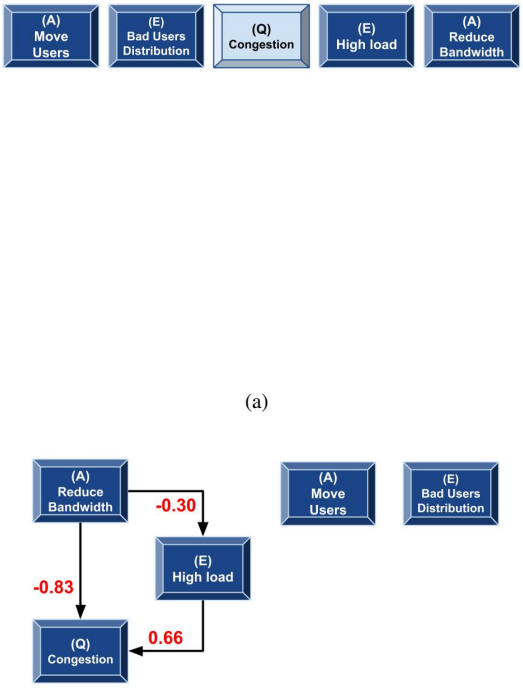


Fig. 29. Fuzzy Cognitive Maps referred to scenario in Figure 28 (a) This FCM underlines that system starts with no a-priori knowledge (b) The Cognitive Engine learnt the effectiveness of reducing bandwidth.

are associated), and reduce the throughput of each users: these two elements represent the FCM's Action variables of the cognitive system.

Side by side, the FCM's Environmental variables are: *Bad User Distribution*, which is related to a non-homogeneous users' distribution among the switches, and *High Load*, related to the overall network utilisation. Finally, the FCM's Quality variable is *Congestion*, which is triggered whenever at least an user cannot achieve a throughput greater or equal to its profile bandwidth (minimum guaranteed bandwidth). The users can also randomly move across the network switches, and intermittently start and stop the Iperf generated throughput: this feature is useful to simulate when users move and not always transmit.

The remainder of this section shows the results, which demonstrate how the Cognitive Engine is actually capable autonomously 'to unlearn' old beliefs once they do not hold to the current environmental situation any more, and immediately after, to learn the new effective countermeasures for an alarm state. In Figure 28, the congestion state is obtained since all the clients transmit to the receiving node *h9*.

Next, Figure 29(a) represents the related FCM, which highlights that the controller has no a-priori knowledge. The controller starts by activating the *Move Users* variable, but since hosts are already homogeneously distributed it has no effect; once it realises that this action is not suitable to the current network condition, it tries to activate the *Reduce Bandwidth* variable, which becomes effective since the congestion state

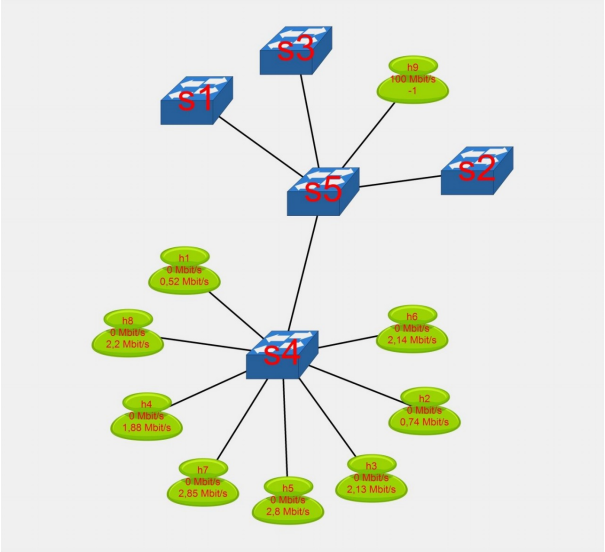


Fig. 30. Scenario to test the performance of cognitive SDN, where users' connections with switches have been changed.

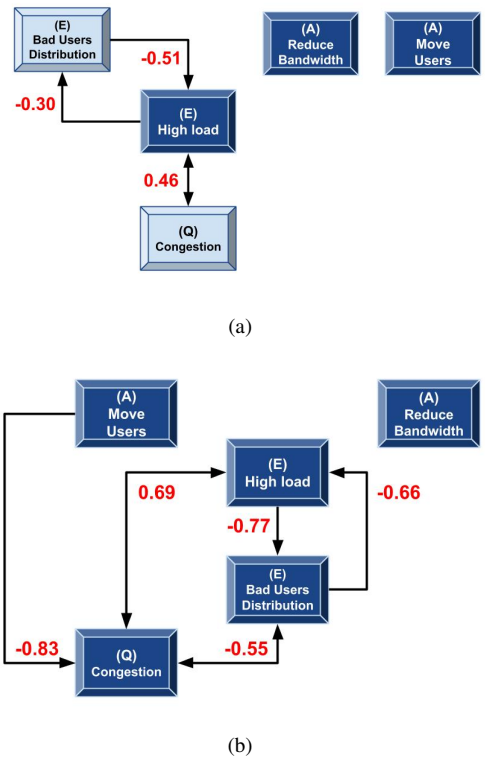


Fig. 31. Fuzzy Cognitive Maps referred to scenario in Figure 30 (a) Cognitive Engine erased past knowledge and now goes on by trials. The objective is to get safe state for *(Q) Congestion* (b) The Cognitive Engine learnt that redistributing users among switches is effective. Creation of new relationship between *(A) Move Users* and *(Q) Congestion*.

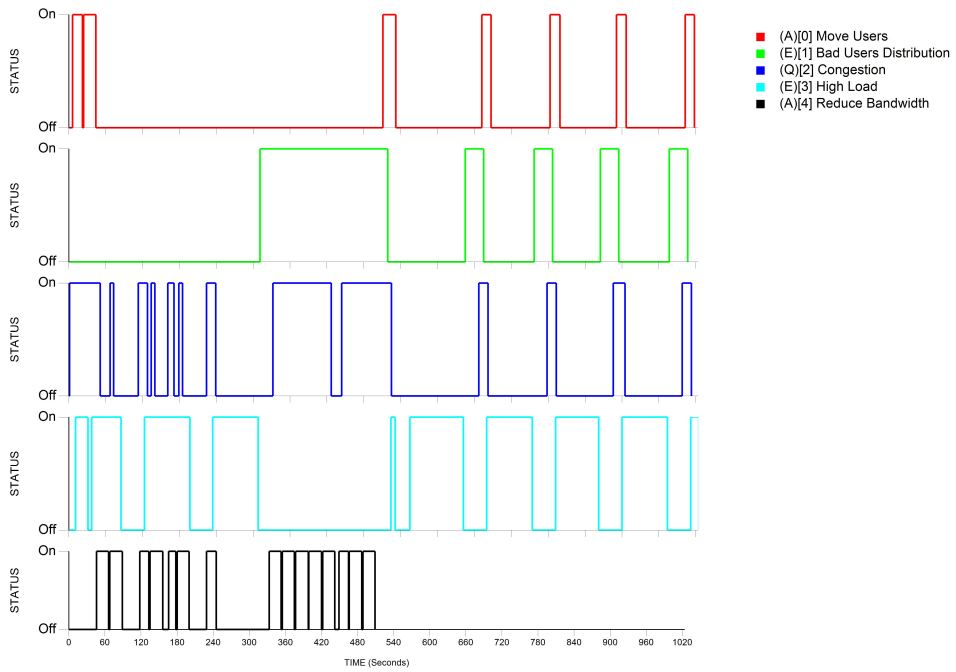


Fig. 32. Graph which highlights the three main moments of the experiment. For clarity, some values have been slightly changed in order to allow a better representation: in reality the response of the Cognitive Engine to an event is much quicker than what seems to be from the graph. The two actions available at the Cognitive Engine are reported by the red and black plots, while the blue one reports the congestion state. Next, the light-green and light-blue lines are related to the two environmental variables. For all of them, the 'on' and 'off' values are related to the active and non-active variable status respectively. The feedback to the cognitive system is retrieved each time: after a change of status of some action variables, there are other environmental or quality variables that change their status.

is not present any more.

At this point, a new causal relationship is created between variables *Reduce Bandwidth* and *Congestion*, and it is enforced each time the controller acquires new evidences about the effectiveness of the action (see Figure 29(b)). Furthermore, a new relationship between *Reduce Bandwidth* and *High Load* is discovered: here, the latter is triggered when there is a congestion because it is in this period that the bandwidth on the gateway is saturated. After few minutes, the bandwidth on the gateway is increased to 100 Mb/s, resulting in a network condition where congestion disappears. As depicted in Figure 32, the Cognitive Engine do not act any more, but the acquired knowledge is preserved.

At this point of the experiment, we manually move the users among the switches, resulting in the network topology of Figure 30: the variable *Congestion* is triggered due to the bottleneck in the link s4–s5 (*h9* remains the receiver), and the Cognitive Engine – which is completely unaware of the new change in the infrastructure – has to react. It first tries to enable the *Reduce Bandwidth* action variable, that FCM has proven to be effective in the past, but, since the hosts have already reached the lowest guaranteed throughput, this action cannot solve the issue.

After few trials, during which the causal relationship between *Reduce Bandwidth* and *Congestion* variables is weakened, the controller returns in its original situation (Figure 31(a)), where it has to discover which action is more suitable for bringing the quality variable in a safe state. Then, the controller discovers that *Move Users* action variable – which was previously ineffective – has now proved to be able to reduce the congestion, thus a new causal relationship between this action variable and *Congestion* is created and enforced (see Figure 31(b)).

In order to let the controller strengthen the new belief, we have manually forced the users to move towards a single switch, resulting in new congestion states being triggered. The final result not only contains information between the currently working action variable for solving the congestion state, but also includes certain causal relationships among different environmental variables. Figure 32 shows the overall experiment variables statuses over time. Especially, it is possible to recognise three main moments: first, *Reduce Bandwidth* is the effective action, second, the latter action is not working any more due to the changed network topology, third the Cognitive Engine realises that the new effective action was *Move Users*.

During the entire test duration, the Cognitive Engine has mixed the learning and testing stages: at the beginning, for the first 40 seconds, due to the complete absence of a-priori knowledge the cognitive process has learnt by test and trials: the only effective action to solve the congestion state was to reduce the bandwidth. In the next period, this belief has been enforced by the successive congestion states that raised and for which the Cognitive Engine has managed to successfully handle them (i.e. the congestion variable has switched from 'on' to 'off' state after the action had triggered). This phase continues until the 330th second, after which the network scenario completely changes: due to deep variations of users' displacement, the

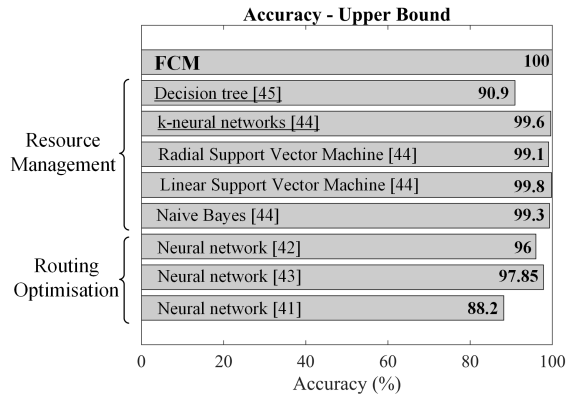


Fig. 33. Comparison of upper bounds achieved by ML-based SDN solutions. Upper bounds on accuracy of legacy solutions come from [13, Table VIII]. The underlined paradigms are belonging to either unsupervised ML or RL.

reduction of bandwidth becomes not effective. Once again, the Cognitive Engine has to learn that by acting (period 330-520 seconds) during which the congestion state remains to an active level. Finally, the-out-of-dated knowledge model is demolished and the Cognitive Engine is ready to experiment with new action patterns; it tries by moving the users which, differently from the initial scenario, is now an effective action (see from the 520th seconds till the end). Finally, excluding the periods during which knowledge had to be built either for lack or out-of-dated knowledge, the system has shown an accuracy of 100% (7 out of 7 congestion status in the first phase and 5 out of 5 in the second one).

Figure 33 depicts a comparison of upper bound of accuracy of legacy ML-based SDN and the proposed FCM-based SDN. In the context of routing optimisation, [41], [42], [43] calculated the accuracy of neural networks applied to SDN. In the context of resource management, [44] calculated the accuracy of some supervised ML algorithms (Naive Bayes, Linear and radial support vector machine and k-neural networks) and [45] evaluated the accuracy of unsupervised ML and RL applied to SDN. As it is possible to see, the proposed algorithm can reach the maximum of accuracy overpassing the legacy ML-based SDN paradigms employing either supervised ML, unsupervised ML or RL. Furthermore, the novel FCM-based SDN handles both routing optimisation and resource management while other legacy solutions only address one of them.

VI. Conclusion

The main scope of the work is to propose the design and the evaluation of a cognitive SDN architecture, employing FCMs as learning algorithm. First, by borrowing the model presented in [12], the authors have discussed and proposed novel characteristics required by FCMs to enable their deployment in the context of cognitive networks. Of particular interest, the authors have proposed the interpretation of continuous edge values as 'signed' probabilities to adapt optimally FCMs to the communication networks' environment and relationships among the related concepts. An efficient method for FCMs' training has also been defined. Next, the authors has showed in detail the proposed architecture for cognitive SDN. The characteristics of this architecture have been studied and its performance has been evaluated in some sample scenarios. It is important to notice the capability of proposed cognitive SDN to adapt and to learn, also without human intervention. Moreover, the presented general cognitive architecture can also work with different learning techniques.

To the best of authors' knowledge, this is the first work describing in detail FCM-based SDN networks, with a specific learning algorithm and with exhaustive tests. Future works will be focused on the following points: (i) further optimisation of FCMs, by using both more advanced learning techniques and training methods; (ii) the experimentation of other learning technologies alternative to FCMs; (iii) the evaluation and test of FCM-based SDN in further more complex scenarios.

References

[1] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. Mackenzie, "Cognitive networks: adaptation and learning to achieve end-to-end performance objectives," *IEEE Communications Magazine*, vol. 44, no. 12, pp. 51–57, Dec. 2006.

[2] B. S. Manoj, R. R. Rao, and M. Zorzi, "CogNet: a cognitive complete knowledge network system," *IEEE Wireless Communications*, vol. 15, no. 6, pp. 81–88, Dec. 2008.

[3] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[4] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[5] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 1617–1655, thirdquarter 2016.

[6] C. Fortuna and M. Mohorcic, "Trends in the development of communication networks: Cognitive networks," *Computer Networks*, vol. 53, no. 9, pp. 1354 – 1376, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128609000085

[7] J. R. P. Mähönen, M. Petrova and M. Wellens, "Cognitive wireless networks: Your network just became a teenager," in *25th Conference on Computer Communications*, 2006.

[8] C. Facchini, F. Granelli, and N. da Fonseca, "Cognitive service-oriented infrastructures," *Journal of Internet Engineering*, vol. 4, no. 1, pp. 269–278, 2010.

[9] B. Kosko, "Fuzzy cognitive maps," *Int. J. Man-Mach. Stud.*, vol. 24, no. 1, pp. 65–75, Jan. 1986. [Online]. Available: http://dx.doi.org/10.1016/S0020-7373(86)80040-2

[10] E. I. Papageorgiou and J. L. Salmeron, "A review of fuzzy cognitive maps research during the last decade," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 66–79, Feb 2013.

[11] E. I. Papageorgiou, "Learning algorithms for fuzzy cognitive maps – a review study," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 2, pp. 150–163, Mar. 2012.

[12] C. Facchini, "Bridging the gap between theory and implementation in cognitive networks: developing reasoning in today's networks," Ph.D. dissertation, University of Trento, Dec. 2011.

[13] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[14] B. P. R. Killi, E. A. Reddy, and S. V. Rao, "Cooperative game theory based network partitioning for controller placement in sdn," in *2018 10th International Conference on Communication Systems Networks (COMSNETS)*, Jan. 2018, pp. 105–112.

[15] H. Kuang, Y. Qiu, R. Li, and X. Liu, "A hierarchical k-means algorithm for controller placement in sdn-based wan architecture," in *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, Feb. 2018, pp. 263–267.

[16] M. Aslan and A. Matrawy, "A clustering-based consistency adaptation strategy for distributed sdn controllers," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 441–448.

[17] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52 – 65, 2013, twenty-fifth Anniversay Commemorative Issue. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012002596

[18] R. S. Sutton and A. G. Barto, *The Problem*. MITP, 1998. [Online]. Available: https://ieeexplore.ieee.org/document/6282959

[19] Y. Li, "Deep reinforcement learning: An overview," *CoRR*, vol. abs/1701.07274, 2017. [Online]. Available: http://arxiv.org/abs/1701.07274

[20] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[21] F. Francois and E. Gelenbe, "Optimizing secure SDN-enabled inter-data centre overlay networks through cognitive routing," in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sep. 2016, pp. 283–288.

[22] S. Sendra, A. Rego, J. Lloret, J. M. Jimenez, and O. Romero, "Including artificial intelligence in a routing protocol using software defined networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 670–674.

[23] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," *CoRR*, vol. abs/1709.07080, 2017. [Online]. Available: http://arxiv.org/abs/1709.07080

[24] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: Optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 64 533–64 539, 2018.

[25] K. K. Budhraja, A. Malvankar, M. Bahrami, C. Kundu, A. Kundu, and M. Singhal, "Risk-based packet routing for privacy and compliance-preserving SDN," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Jun. 2017, pp. 761–765.

[26] R. Haw, M. G. R. Alam, and C. S. Hong, "A context-aware content delivery framework for QoS in mobile cloud," in *The 16th Asia-Pacific Network Operations and Management Symposium*, Sep. 2014, pp. 1–6.

[27] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for RSU clouds in support of the internet of vehicles," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 133–144, Apr. 2015.

[28] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," *CoRR*, vol. abs/1711.09012, 2017. [Online]. Available: http://arxiv.org/abs/1711.09012

[29] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "A game theoretic approach for distributed resource allocation and orchestration of softwarized networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 721–735, Mar. 2017.

[30] X. Huang, T. Yuan, G. Qiao, and Y. Ren, "Deep reinforcement learning for multimedia traffic control in software defined networking," *IEEE Network*, vol. 32, no. 6, pp. 35–41, Nov. 2018.

[31] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2392–2431, Fourthquarter 2017.

[32] Z.-Q. Liu, "Causation, bayesian networks, and cognitive maps," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 27, 12 2000.

[33] D. Fletcher, D. Nguyen, and K. Cios, "Autonomous synthesis of fuzzy cognitive maps from observational data: Preliminaries," in *2005 IEEE Aerospace Conference*, Mar. 2005, pp. 1–9.

[34] A. K. Tsadiras, "Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps," *Information Sciences*, vol. 178, no. 20, pp. 3880 – 3894, 2008, special Issue on Industrial Applications of Neural Networks.

[35] T. R. Newman, B. A. Barker, A. M. Wyglinski, A. Agah, J. B. Evans, and G. Minden, "Cognitive engine implementation for wireless multicarrier transceivers," *Wireless Communications and Mobile Computing*, vol. 7, pp. 1129 – 1142, 11 2007.

[36] J. A. Dickerson and B. Kosko, "Virtual worlds as fuzzy cognitive maps," in *Proceedings of IEEE Virtual Reality Annual International Symposium*, Sep. 1993, pp. 471–477.

[37] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2181–2206, Fourthquarter 2014.

[38] ONOS. Onos overview white paper. [Online]. Available: http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf

[39] ——, "Open network operating system (ONOS) wiki," Sep. 2017. [Online]. Available: https://wiki.onosproject.org/

[40] Mininet. Mininet – an instant virtual network on your laptop (or other pc). [Online]. Available: http://mininet.org/

[41] L. Yanjun, L. Xiaobo, and Y. Osamu, "Traffic engineering framework with machine learning based meta-layer in software-defined networks," in *2014 4th IEEE International Conference on Network Infrastructure and Digital Content*, Sep. 2014, pp. 121–125.

[42] C. Chen-xiao and X. Ya-bin, "Research on load balance method in SDN," *International Journal of Grid and Distributed Computing*, vol. 9, pp. 25–36, 01 2016.

[43] A. Azzouni, R. Boutaba, and G. Pujolle, "Neuroute: Predictive dynamic routing for software-defined networks," *CoRR*, vol. abs/1709.06002, 2017. [Online]. Available: http://arxiv.org/abs/1709.06002

[44] D. D. Testa, M. Danieletto, G. M. D. Nunzio, and M. Zorzi, "Estimating the number of receiving nodes in 802.11 networks via machine learning techniques," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–7.

[45] J. Bendriss, I. G. B. Yahia, and D. Zeghlache, "Forecasting and anticipating SLO breaches in programmable networks," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Mar. 2017, pp. 127–134.