Mobile Netw Appl (2011) 16:475–489
DOI 10.1007/s11036-011-0327-1

Enhancing a Fuzzy Logic Inference Engine through Machine Learning for a Self- Managed Network

Panagis Magdalinos · Apostolos Kousaridas · Panagiotis Spapis · George Katsikas · Nancy Alonistioti

Published online: 14 June 2011

Abstract Existing network management systems have static and predefined rules or parameters, while human intervention is usually required for their update. However, an autonomic network management system that operates in a volatile network environment should be able to adapt continuously its decision making mechanism through learning from the system's behavior. In this paper, a novel learning scheme based on the network wide collected experience is proposed targeting the enhancement of network elements' decision making engine. The algorithm employs a fuzzy logic inference engine in order to enable self-managed network elements to identify faults or optimization opportunities. The fuzzy logic engine is periodically updated through the use of two well known data mining techniques, namely k-Means and k-Nearest Neighbor. The proposed algorithm is evaluated in the context of a load identification problem. The acquired results prove that the proposed learning mechanism improves the deduction capability, thus promoting our algorithm as an attractive approach for enhancing the autonomic capabilities of network elements.

Keywords network self-management · future internet · fuzzy logic · data mining

P. Magdalinos · A. Kousaridas (✉) · P. Spapis · G. Katsikas · N. Alonistioti
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Panepistimiopolis, 157-84, Athens, Greece
e-mail: akousar@di.uoa.gr

P. Magdalinos
e-mail: panagis@di.uoa.gr

P. Spapis
e-mail: pspapis@di.uoa.gr

G. Katsikas
e-mail: katsikas@di.uoa.gr

N. Alonistioti
e-mail: nancy@di.uoa.gr

1 Introduction

Several network management frameworks have been specified during the last two decades by various standardization bodies and fora, like IETF [1], 3GPP [2], DMTF [3], ITU [4], all trying to specify interfaces, protocols and information models by taking into consideration the respective network infrastructure i.e., telecom world, internet and cellular communications. The current challenge for the network management systems is the reduction of human intervention in the fundamental management functions and the development of the mechanisms that will render the Future Internet network capable to autonomously configure, optimize, heal and protect itself, handling in parallel the emerging complexity. Designing an autonomic system for network management involves several technologies and disciplines and has received significant research effort during the last decade (e.g., [5–8]).

The majority of the proposals for autonomic systems are based on the so called closed control loop or Monitor-Decide-Execute Cycle (MDE) [9, 10]. Each autonomic element consists of the autonomic manager (AM) that instantiates the MDE cycle and the respective managed resource. The AM monitors resources' state through the available sensors and builds the knowledge model that is used in conjunction with the monitoring data in order to analyze the current status of the managed resource and thereinafter decide or plan the best (configuration) action. Cognition development is an important aspect of future Internet self-managed systems, which

complements and advances the automation of actions. An in-network cognitive cycle will allow a communication system to improve its inference and reasoning capabilities, by exploiting the feedback from previous events or from historic data that are stored locally [10, 11].

One of the roles of the network administrators is to observe various alarms and the values of critical metrics after the execution of specific configuration actions. Thereinafter, using the accumulated experience, they attempt to improve both situation perception and deduction processes in order to avoid redundant re-configurations and achieve early identification of optimization opportunities or faults. Future Internet self-managed networks shall incorporate the necessary algorithms that will facilitate the network administrator and thus render each network element capable to learn from previous adaptations (configuration actions) by assessing their effectiveness. There are many discussions on the applications of machine learning schemes and on the introduction of the learning capabilities, from an architectural point of view [12, 13]. However, the majority of the machine learning algorithms for communication systems evolution are targeting a specific network environment or a specific application e.g., traffic management, radio management [14].

The scope of this paper is to propose a holistic feedback-based learning scheme for an autonomic network management system, which has been designed following the principles of a hierarchical architecture of cognitive managers placed per network element. Each learning scheme affects and is affected by the decision making engine of the cognitive managers for situation perception and/or configuration action selections. In this paper we are enhancing the quality of a fuzzy logic-based decision making engine. The algorithm exploits the merits of fuzzy logic as well as the benefits of the k Nearest Neighbour (k-NN) in order to develop the knowledge base for network node's experience. The accumulated information is used as input for the enhancement of the decision making process through a novel learning framework. The latter is based on the high dimensional Euclidean geometry of the k-Means algorithm [15, 16].

The rest of this paper is structured as follows: in the next section we present some background information which will enable the reader to delve into the details of our work. At first we outline the Self-NET project [17] architectural framework that has been adopted for the design of the proposed learning scheme. The fuzzy logic based decision making engine as well as a number of key data mining algorithms, which are employed in the context of our learning approach are presented. Then we present our algorithm in details and we showcase the validity and viability of our approach through a real life use case from the family of coverage and capacity optimization of wireless access networks. Finally, we conclude the paper and sketch future research direction.

2 Background

In the context of this paragraph we will elaborate on some key concepts which set the methodological foundations of our work. At first we present the network management architecture introduced by the Self-NET project and discuss in detail the corresponding decision making engine. Finally, we present three well known data mining algorithms namely k-Means, k-NN and hierarchical clustering.

2.1 Network management architecture

In the autonomic network vision, each network device (e.g., router, access point), is potentially considered as an autonomic element which is capable of monitoring its network-related state and modifying it based on conditions that administrators have specified. The feedback-based cognitive cycle residing in each network element leads to their autonomy. The Future Internet element, with embedded cognition, includes processes for monitoring and perceiving network node's internal state and environmental conditions, and then planning, deciding and finally adapting according to these conditions. Furthermore, such an element is able to learn from these adaptations (re-configurations) and assess their effectiveness for improving decision making mechanisms.

The learning scheme that is proposed is this paper takes into consideration the Self-NET architecture (Fig. 1) [18], which follows a hierarchical distribution of cognitive cycles, breaking down the respective functional entities and mechanisms for solving network management problems and other self-management operations (e.g., learning, monitoring) to: a) network elements (e.g. access points), b) network compartments (opportunistic/short-term federations of network elements), c) network domains (structured/long-term federations of network elements) and d) the (hyper-domain) network management system that controls the underlying entities and provides the human (e.g., administrator) interface.

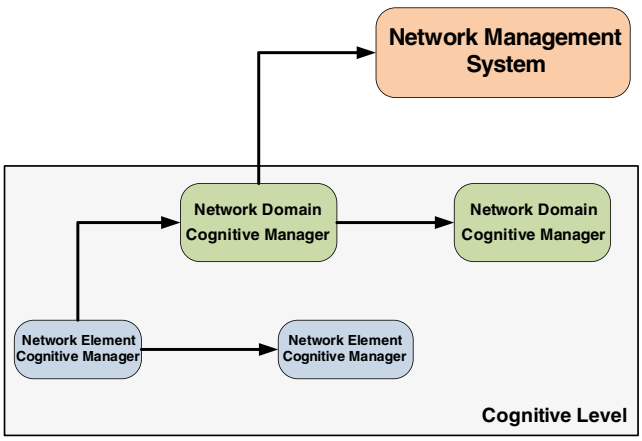


Fig. 1 Self-NET network management architecture

For the implementation of the cognitive cycle at the network element and the domain level, the Network Element Cognitive Manager (NECM) and the Network Domain Cognitive Manager (NDCM) have been introduced, respectively. NECM and NDCM are software agents that use the monitoring and execution mechanisms, which are available by the respective network nodes and incorporate the algorithms and schemes for the development of the cognitive cycle features [19]. The NDCM is a more sophisticated agent that can deal with higher-level knowledge, collected by a set of subsuming NECMs in the specified domain that it manages. NDCMs having this broader network view and consequently knowledge, can proceed with global decision making, thus solving problems that standard NECMs cannot address locally, adjust policies to fit the desired system behavior and learn from previous experiences to improve the impact of future decisions.

The decision making engine of each NECM and/or NDCM includes the problem solving techniques for network nodes efficient adaptation, utilization of the developed knowledge model and situation awareness. The decision making engine involves the identification of faults or optimizations opportunities and the selection of the most appropriate configuration action taking into account the outcomes of the former as well as the alternatives for network node(s) reconfiguration. After this phase, one or more NECMs undertake the application of the respective configuration action.

2.2 Fuzzy logic inference process

The term "Situation Perception" is used to describe all correlations that take place in order to analyze data received by monitoring points and thus identify problems and select appropriate configuration actions. This task is considered as a complex multi-variable problem since multiple optimization goals or faults may arise. Therefore, Fuzzy Logic is the algorithmic tool that has been selected so as to address situation perception building. The Fuzzy Logic based situation perception, takes into account a set of metrics/parameters, and after their joint correlation analysis, maps them to a degree that depicts how the network elements perceives its environment (e.g., load status). This perception is mapped to a value for each state that ranges between 0 and 1, and describes the degree of each state e.g., load, interference, noise.

The Fuzzy Logic Controller (FLC) consists of three parts, namely the fuzzifier, the inference system and the defuzzifier (Fig. 2). The fuzzifier undertakes the transformation (fuzzification), of the input values (crisp values) to the degree that these values belong to a specific state (e.g., low, high) (Fig. 3). Then, the inference system correlates the inputs and the outputs using simple "IF…THEN…" rules; each rule results to a certain degree for every output. Thereinafter, the output degrees for all the rules of the inference phase are being aggregated (Fig. 4). The

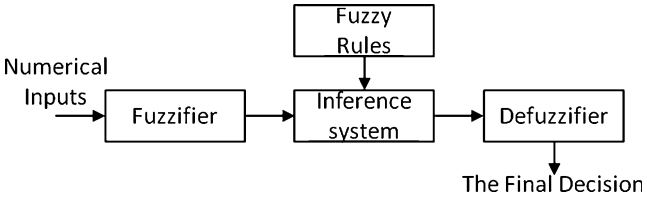


Fig. 2 High level view of a Fuzzy Logic controller

actual output of the decision making process, comes from the defuzzification procedure, which captures the degree of the state of the decision maker (e.g., the network element is x% loaded; the radio link is y% interferenced etc.). The degree is obtained using several defuzzification methods; the most popular is the centroid calculation, which returns the center of gravity of the degrees of the aggregated outputs (Fig. 4) and is calculated using the following formula [20]:

$$u_{COG} = \frac{\sum_{i=1}^{n} u_i \mu_F(u_i) du}{\sum_{i=1}^{n} \mu_F(u_i)} \quad (1)$$

Where u_i are the centers of the membership functions $\mu_f(u)$ and i is the number of rules.

2.3 Data mining algorithms

In the context of the proposed algorithm we employ data mining algorithms in order to enhance the quality of the fuzzy logic inference engine. Due to the fact that our framework targets a large set of devices, ranging from limited capabilities handheld mobile terminals to base stations, the employed algorithms should be fast, accurate and impose minimum memory and time requirements in conjunction with high quality results. Therefore, we opted for hierarchical divisive clustering (HDC) with k-Means and the k-Nearest Neighbour classifier. k-NN classifiers are simple, fast and easy to train while careful selection of k makes their accuracy comparable to that of the best known classifiers. On the other hand, HDC

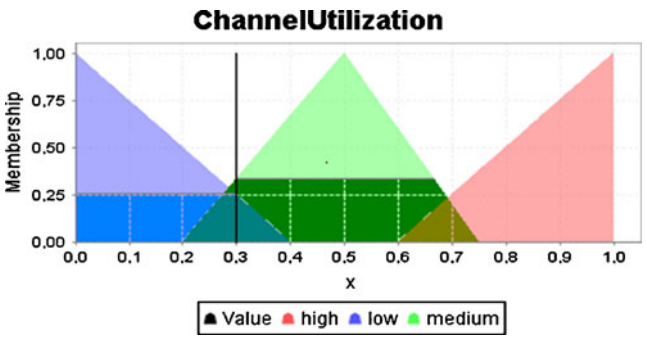


Fig. 3 Fuzzification of the input

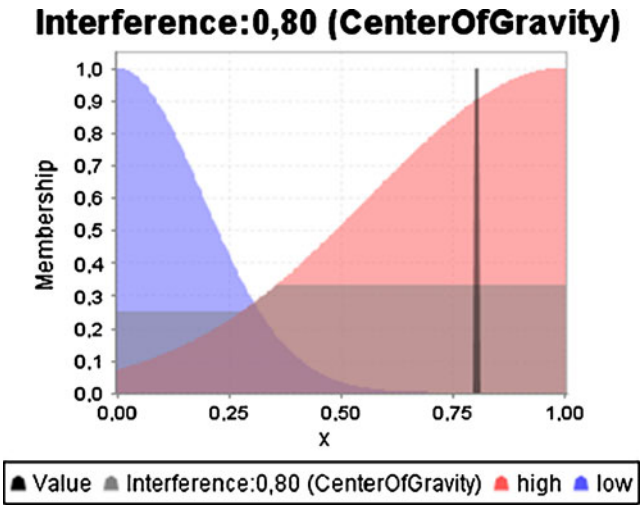


Fig. 4 Output aggregation and defuzzification

in its base form is slow, but is significantly accelerated when combined with k-Means.

Hierarchical clustering [9] is an unsupervised learning method which builds a hierarchy of clusters by following one of the paradigms below:

Agglomerative Observations are considered as forming, separate single-instance clusters and are progressively merged according to a similarity criterion.

Divisive All observations are considered to belong to a single cluster which is divided according to some similarity criterion.

The outcome of this procedure is usually in the form of a dendogram. Agglomeration and division usually take place by employing a similarity function such as the Euclidean distance or the dot-product. The procedure is halted when a specific similarity or dissimilarity threshold is surpassed. Its key feature lays in the fact that it requires as input only the afore-described threshold and provides as output the number of clusters as well as cluster memberships. HDC in its simplest form is extremely slow, since each iteration necessitates the identification of the two most similar points thus its computational complexity is exactly $O(N^2)$ per iteration, where N is the number of points in the dataset while when combined with k-Means, requirements are minimized to O (eN), where e is the number of required loops.

k-Means [16] is a fast and efficient clustering algorithm extensively employed in the area of unsupervised learning. The core idea is to partition a set of *N*, *d*-dimensional, observations into *k* groups such that intra-group observations exhibit minimum distances from each other while inter-group distances are maximized. *k*-Means tries to minimize the following objective function:

$$\min \sum_{i=1^K} \sum_{j=1^N} \left\| x_j - m_i \right\|^2 \quad (2)$$

where x_j is the j-th observation, m_i the center of the i-th cluster and ||x-y|| the Euclidean distance of points $x,y \epsilon R^d$. The centre of each cluster C_i is defined as the mean of all points belonging to C_i. It has to be stressed out that the *k*-means clustering problem is NP-hard even for the elementary case of $k=2$. However the approximation of [21–23], which is employed in practice, provides fast and reliable results. Overall time requirements of k-Means are O(eN) where e is the number of loops before convergence.

Contrary to *k*-Means and hierarchical clustering, *k*-NN [16] is a supervised learning algorithm which employs a set of already labeled observations in order to classify a smaller set of unknown data points. The core idea of the algorithm is to assign a given data point the class label which appears in the majority of its *k* nearest neighbors. Despite its efficiency and reduced memory requirements, *k*-NN quality deteriorates quickly as the number of dimensions grows. As more and more dimensions are added to each observation (i.e. each observation is described by additional variables) the maximum and minimum distance in the dataset tend to quality, a phenomenon known as the curse of dimensionality [24]. The latter can be addressed by employing dimensionality reduction algorithms as a data preprocessing step.

3 Framework modeling

In this section the proposed feedback based learning scheme is described, which capitalizes on the afore-described cognitive managers' hierarchical architecture and the decision making engine of our framework.

3.1 High level description

The goal of the proposed scheme is to continuously adapt the operation policies of the NECMs' inference engines so at to enhance their ability to identify faults or optimization opportunities at the network element (e.g., access point) level. Specifically, the goal is to update the membership functions of the Fuzzy logic inference engine of each NECM that are being used for the situation perception building. Each NECM adapts the behavior of its inference engine according to the calculations that take place at NDCM. The latter utilizes the accumulated experience/information from the underlying NECMs in conjunction with the macroscopic view that has been extracted from previous sessions

Figure 5 provides a high level description of the employed learning scheme. The algorithm employs a hybrid version of the k-NN, k-Means and hierarchical clustering procedures. It is highly decentralized, in the sense that a part of the algorithm is executed at the network elements level (NECM) and another part at the controller (NDCM) level. Each NECM periodically monitors its

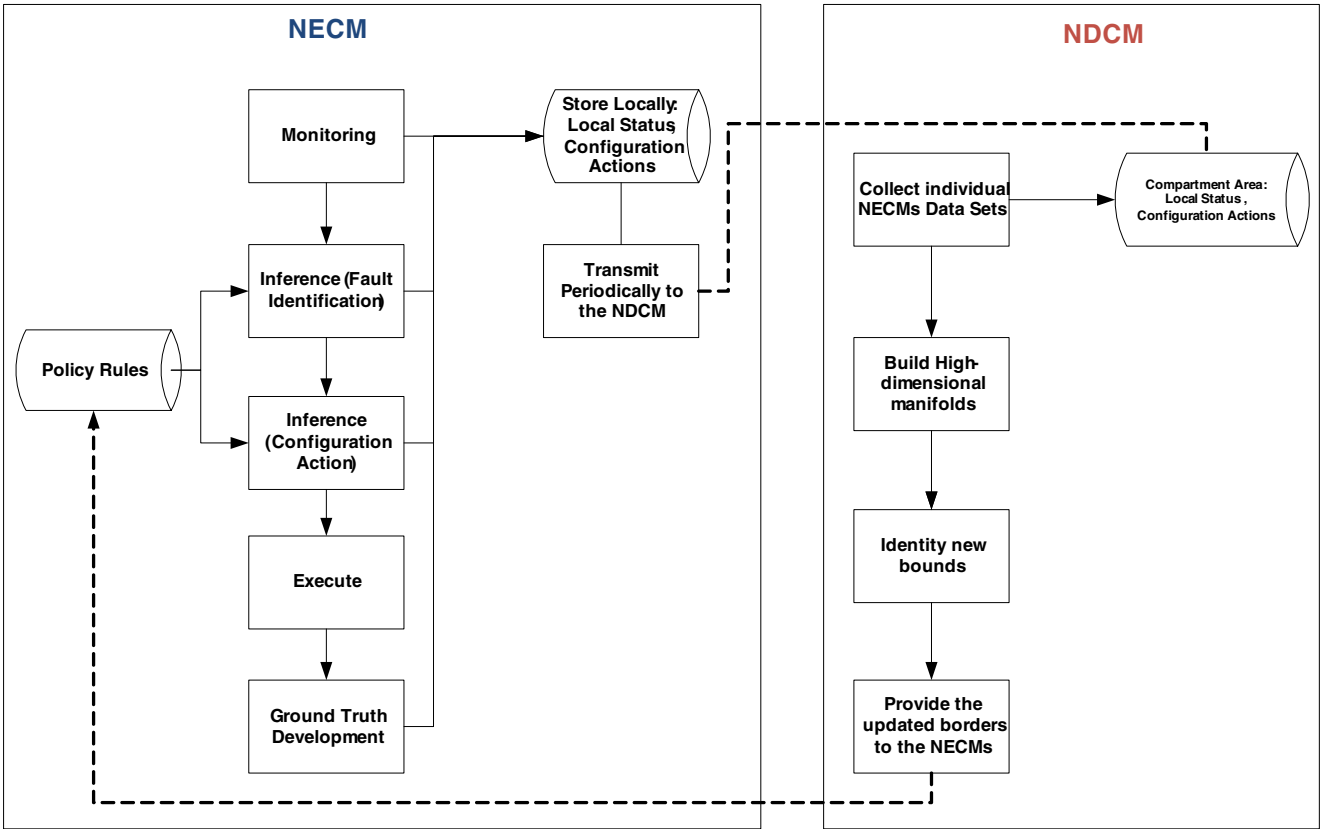


Fig. 5 Problem-agnostic description of the feedback-based learning scheme

operational environment and evaluates the identified information in order to deduce (using a fuzzy logic inference engine) faults or optimization opportunities (e.g., high load, high interference) and –if necessary- an appropriate configuration action to be executed (e.g., channel reallocation, assisted handover of associated terminals). Each NECM evaluates the correctness of each deduction (followed by a configuration action or not) building the so called ground truth. Ground truth characterizes the actual conditions (i.e. load) and is identified by assessing the network node status after the re-configuration.

However, in many cases, according to the network conditions, the range for the characterization of a situation should be adapted according to the effectiveness of the selected configuration action and the specific features of the network environment. Nevertheless, this decision necessitates a holistic view of the network and consequently implies the dissemination of the aggregated set of local observations (NECM) to a higher level –in terms of architecture- entity (NDCM). The NDCM collects individual NECMs data sets and by applying data mining techniques specifies the new bounds for the fuzzy logic inference engine. A detailed analysis of this mechanism is provided in the following section.

3.2 Algorithm presentation

The identified algorithm consists of three distinct steps, namely the monitoring phase, the classification step and the fuzzy logic enhancement procedure. Throughout the presentation we assume that each time a network device (i.e. NECM) monitors its operational environment it extracts a *d*-dimensional vector which can be classified as *True*, indicating that a particular problem has appeared, *False* – no problem- or *Neutral*, implying that although there is currently no problem there is a chance that a problematic situation may appear in the future. Additionally, given a problem, the device triggers a remedy action which is guaranteed to solve the problem; in other words it will enable the device to transit from a *True* state to either a *False* or *Neutral* state. Obviously, upon start-up, the device has no pre-installed knowledge base, apart from the set of fuzzy logic rules and the set of configuration actions.

The monitoring algorithm appears in Table 1. The device monitors its operational environment and extracts a *d*-dimensional tuple (step 4.2) which is evaluated against the set of pre-installed fuzzy logic rules (step 4.3). At this step, the inference process of the NECM for the situation perception (fault or optimization opportunity identification)

Table 1 Monitoring algorithm on the network element level

Input:	Approximation Parameter ε
Output:	Set of labeled observations S, Set of unlabeled observations T
1.	S←O
2.	T←O
3.	i=0
4.	while true
4.1	i++
4.2	Retrieve vector $\vec{Z}_i$
4.3	X_i←fuzzy logic ($\vec{Z}_i$)
4.4	if (X_i=True)
4.4.1	Select and Apply appropriate Solution
4.4.2	Wait for S_{i+1}
4.4.3	if($\|S_{i+1}-S_i\|<\varepsilon$)→$Y_i$=Neutral
4.4.4	else Y_i=True
4.4.5	S=S U {$\vec{Z}_i$, X_i, Y_i}
4.5	else if (X_i=Neutral/False)
4.5.1	Y_i=?
4.5.2	T=T U {$\vec{Z}_i$, X_i, Y_i}
5	return S, T

takes place. If the outcome denotes a problematic situation (step 4.4, label X_i) then the appropriate solution is applied (step 4.4.1). Each cognitive manager per network has a set of solutions or configuration actions that could be enforced. Each problem is associated with one or more solutions. According to the global status of the network device, and possible interactions among different solutions the most appropriate is selected. Given the fact that the applied solution will always solve the problem we compare the $i+1^{th}$ tuple with the i^{th} (step 4.4.3). In case their distance is less than a predefined bound ε we assume that we performed a classification error (false positive, i.e. we classified a Neutral tuple as true) and attribute the correct label Y_i. Y_i corresponds to the actual conditions (ground truth) while label X_i to the fuzzy logic perception of the environment. In any other case, we cannot decide about the label and leave it as it is. As soon as a significant amount of vectors (i.e. *N*) is aggregated we halt the procedure and proceed with the application of the *k*-NN classifier.

The k-NN classifier enables the identification of all missing Yi labels. The set of labeled instances (S) is used as the training set, while all unlabeled records (T) are used as testing set. Recall from the last step that although we can accurately predict the labels of all observations appearing in S, we only have tuples from Neutral and True. In order to overcome this, we artificially generate a small number of tuples which are in advance labeled as False (i.e. all tuples are located in the beginning of the coordinates systems axes). It should be pointed out that this step appears only the first time that the algorithm is executed. In subsequent executions, the training set is populated with previously labeled records. The algorithm appears in Table 2. The successful execution of this step essentially generates a set of correctly labeled observations which can be used in order to quantify the quality of the procedure.

At this point it should be added that the k-NN classification could be replaced with other classification methods as well such as the Support Vector Machines. However, such approach would increase the complexity of the overal scheme, due to the fact that, in general, the tuples cannot be captured by simple SVM classification. This implies that we should proceed in complex transformations and increase the dimensionality of the input tuples. Even though the previous scheme can also successfully classify input data, the limited processing power of the NECMs - where classification takes place- prohibits the use of such approach.

Periodically the stored tuples are validated in order to provide an indirect assessment measure with respect to the quality of the fuzzy logic rules. The evaluation is done according to formula $A = \sum_{i=1}^{N} \frac{|X_i - Y_i|}{N}$. A essentially quantifies the percentage of cases that we made an erroneous decision (i.e. the ground truth label Y_i is different than the fuzzy logic label X_i) and is compared with a predefined tolerance bound A_p. If the number of mistakes is not tolerable ($A>A_p$) then the network element sends all data to the domain controller (NDCM).

The domain controller receives data from all network elements for which condition ($A>A_p$) holds true. All measurements lay in a *d*-dimensional space and by exploiting the ground truth labels (Y_i) we can categorize them in three distinct classes $C_i \in \{True, Neutral, False\}$ which correspond to three high dimensional manifolds (D_T, D_N, D_F). For ease of presentation, in the context of this work, we will assume that data points form hyperspheres, however the work can be extended to address a multitude of high dimensional manifolds.

Table 2 k-NN classification for the extraction of the missing labels

Input:	Set of labeled observations S, Set of unlabelled observations T
Output:	Final set of labeled observations F
1.	F←O
2.	kNN.training set←{S}
3.	kNN.test set←{T}
4.	F←kNN(Training, Test)
5	return F

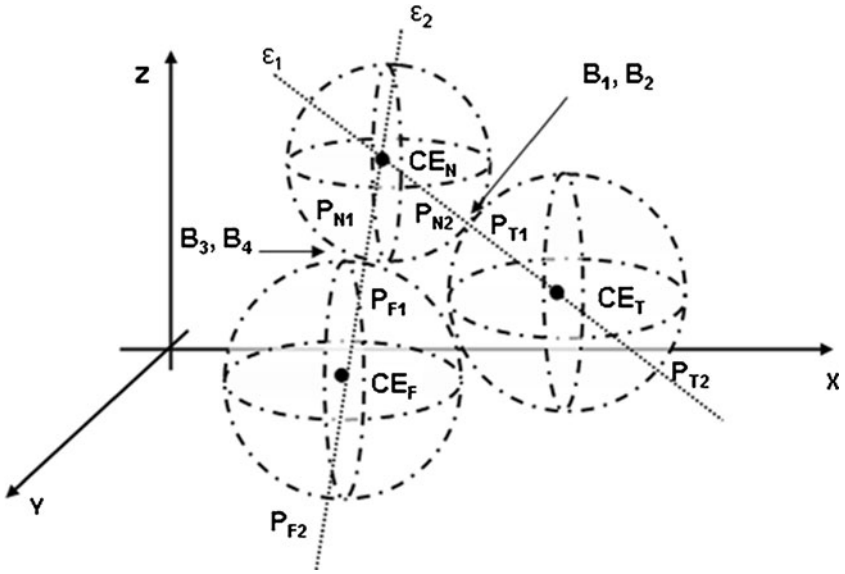


Fig. 6 Graphical representation of data records according to ground truth labels

Each sphere is centered at $CE_i = \sum_{j=1}^{|C_i|} \frac{S_j}{|C_i|}, S_j \in C_i$ and has radius $R_i = \max_{j=1}^{|C_i|} \left\| CE_i - S_j \right\|$. We assume that the cluster of tuples labeled as *True* is centered at $CE_T=(x_1, x_2,\ldots,x_d)$ and has radius R_T, while tuples corresponding to *Neutral* are centered at $CE_N=(y_1, y_2,\ldots,y_d)$ with radius R_N. Similarly, *False* points are centered at $CE_F=(Z_1, Z_2,\ldots,Z_d)$ with radius R_F. Without loss of generality we consider only points CE_T and CE_N. These two points define a line ε which is described by the following set of equations:

$$p_i = x_i + u^*(y_i - x_i), i = 1...d \quad (3)$$

This line intersects with spheres D_T and D_N in four points which can be retrieved by substituting the p_i values into the following hypersphere equations:

$$D_T \rightarrow \sum_{i=1}^{d} (p_i - x_i)^2 = R_T^2 \quad (4)$$

$$D_N \rightarrow \sum_{i=1}^{d} (p_i - y_i)^2 = R_N^2 \quad (5)$$

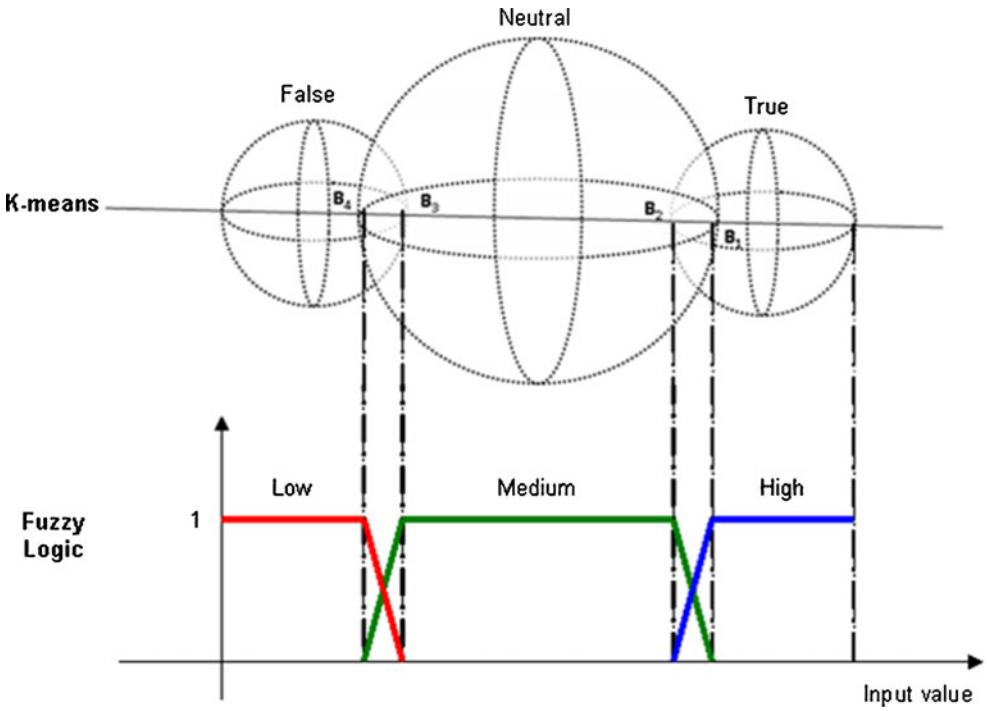


Fig. 7 Geometric interpretation of the approach

Consequently, a simple way of identifying the bounds for the fuzzy logic rules would be to extract the intersection points which

- belong to different hyperspheres and
- exhibit minimum distance from each other.

Simply stated, the two intersection points are provided by:

$$\{B_1, B_2\} = \min\{\|P_{1T} - P_{1N}\|, \|P_{1T} - P_{2N}\|, \|P_{2T} - P_{1N}\|, \|P_{2T} - P_{2N}\|\}. \quad (6)$$

By applying a similar procedure we can also extract points B_3 and B_4 from spheres D_N and D_F. Notice at this point that each B_i corresponds to a tuple $\{b_1, b_2, \ldots, b_d\}$; thus each B can be directly set as a new bound for the fuzzy logic rules. More precisely:

- Point B_1 would correspond to the bound for the *True-Neutral* situation, in other words, B_1 is a point labeled as *True* that exhibits maximum distance from C_{ET} and is closest to C_{EN} than any other point P labeled as *True*.
- Point B_2 would correspond to the upper bound for the *Neutral-True* situation. Similarly B_2 is labeled as *Neutral* and exhibits maximum distance from C_{EN} and is closest to C_{ET} than any other point labeled as *Neutral*. Since D_T and D_N are adjacent, $B_1=B_2$.
- Point B_3 would correspond to the lower bound for the *Neutral-False* situation
- Point B_4 corresponds to the bound for the *False-Neutral* situation (obviously $B_3=B_4$)

A graphical representation of this procedure appears in Fig. 6 where we demonstrate the application of our algorithm in R^3.

Table 3 Quantifying the induced amelioration

Input:	Set of observations S, New Bounds B, Evaluation Bound A_p
Output:	New Rules FL, Evaluation Bound A
1.	FL ← Adapt local rules (B)
2.	i=0
3.	$X_n \leftarrow O$
4.	while i<−S|
4.1	i++
4.2	Retrieve tuple $\vec{S}_i$
4.3	$X_i \leftarrow$ fuzzy logic ($\vec{F}_i$)
4.4	$X_n = X_n U X_i$
4.5	End
5.	$A_n = \Sigma_{i=1}^{\|S\|} \|X_{ni} - Y_i - / - S\|$
6.	if ($A_n < A_p$)
6.1	A=0.9Ap
7.	return A, FL

Table 4 k-NN classification ability in the context of load identification use case

Number of NNs	1	5	10
Correctly Classified Instances	98.7%	98.6%	98.5%

It should be stressed out that hyperspheres comprise an abstraction of the actual manifold formed by the data points so the actual manifold is enclosed in the hyperspheres. However, this abstraction fits our purposes for two reasons. Firstly, its computation is simple and fast thus it can be implemented on any kind of device without imposing any memory or CPU overhead. Secondly, the circumference of the hypersphere will contain at least one point of the class under process, thus indirectly signifying the range of values of that class.

On the other hand, the adjacency of the spheres may yield poor discrimination quality, in the sense that a lot of *True* and *Neutral* cases as well as *Neutral* and *False* cases may have been placed together. The latter, is due to the fact that the hyperspheres enclose a larger area than the actual manifold. In order to overcome this we employ a hierarchical divisive clustering (HDC) approach based on *k*-means. An indirect gain however, is that the application of HDC will take place on the D_T and D_F spheres and not on the larger, D_N sphere. Essentially, the fitting of data into three spheres comprises a fast implementation of the first step of HDC in O(N) time which is significantly smaller than the O(eN) requirement of k-means.

Afterwards, k-Means is applied on the two spheres which correspond to *False* and *True* and direct the algorithm to split it into two clusters, *False* or *True* and *Neutral*. The result will be two adjacent spheres maintaining elements belonging to both classes. The new sphere corresponding to *Neutral* is merged with the initial *Neutral* class. The division continues on the resulting *True* and *False* clusters until we start experiencing loss in the *Recall* (Recall=Retrieved Relevant Records/Total Relevant Records) or high *Precision* (Precision= Retrieved Relevant Records/Total Retrieved Records) in conjunction with high *Recall* (high F-measure value, where F-measure=2*Recall*Precision/Recall+Precision). The geometric interpretation of our approach is depicted in Fig. 7. The algorithm simply augments the sphere corresponding to

Table 5 Bounds used in the membership functions of the fuzzy logic rules

	PER	CU	AT
Low	$FL_{1,2,3}$:[0…0.01]	$FL_{1,2}$: [0…0.2] FL_3: [0…0.5]	$FL_{1,2,3}$: [0…10]
Medium	$FL_{1,2,3}$: [10^{-4}…0.05]	$FL_{1,2,3}$: [0.1…0.9]	$FL_{1,3}$: [9…16] FL_2: [5…20]
High	$FL_{1,2,3}$: [7*10^{-3}…1]	$FL_{1,2}$: [0.8…1] FL_3:[0.5…1]	$FL_{1,2,3}$: [15…25]

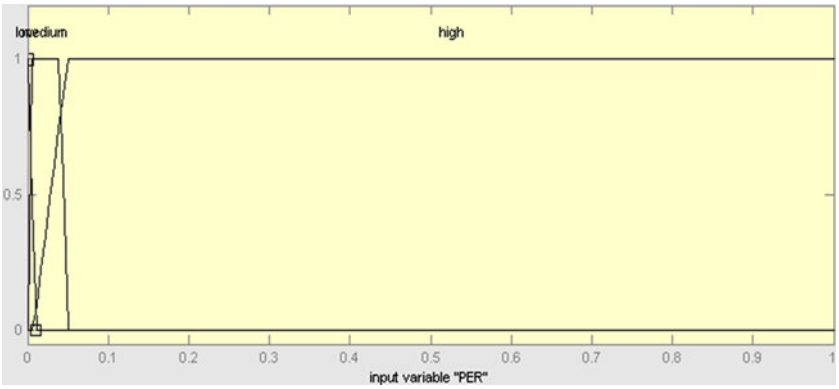


Fig. 8 PER initial membership functions configuration

Neutral cases and shrinks the other two by extracting falsely classified points. When the procedure is halted then we have three overlapping hyperspheres. The intersecting points of the line defined by the spheres' centers with the spheres correspond to the desired solution.

We can quantify the amelioration induced by the algorithm by employing the procedure appearing in Table 3. Due to the fact that every network element holds a copy of the fuzzy logic rules, NDCM adapts its local rules (step 1) and re-evaluates the whole set of observations (steps 2–4.5). In the end, we quantify the amelioration induced by our algorithm by comparing the derived fuzzy logic labels with the existing ground truth labels. If the derived value is lower than the predefined bound A_p, we define a new stricter bound (step 6.1), otherwise we consider than no further amelioration can take place. The new rules together with the new evaluation bound of the FL membership functions are transmitted back to the underlying network elements (NECMs), where the local fuzzy logic inference engine is updated.

The computational load induced to NDCM due to the application of our algorithm is O(ceN) where c is the number of iterations required before the procedure reaches its halting condition, N the number of tuples, e the number of iterations required by k-Means to converge and d the dimensionality of data. The network load is O(Nd+Xd), where X is the number of network elements. Specifically, O (Nd) is due to the transmission of the original tuples from the NECMs to NDCM and O(Xd) is required for the transmission of the results back to the network elements.

4 Performance analysis

In this section we present the experimental evaluation of our approach, which verifies the expected performance. Thus, our algorithm emerges as an attractive solution for learning in a communication network context. The aim of the experimental assessment process is threefold:

- To validate the effectiveness and efficiency of our algorithm on real life problems and highlight its scalability,
- To demonstrate the enhancement of fuzzy logic decision making due to the application of our algorithm and
- To experimentally validate the applicability of k-NN for the derivation of the ground truth labels in the first step of the algorithm.

The obtained results prove the suitability and viability of our algorithm for network management problems in the context of future Internet networks.

The experimental analysis was carried out for the problem of load identification, which is under the coverage and capacity optimization umbrella. The analysis could be extended for any inference process that an NECM should execute. In this use

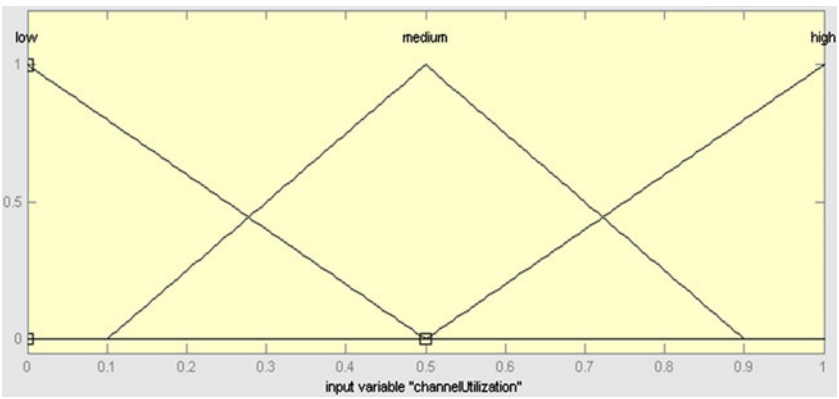


Fig. 9 Channel Utilization initial membership functions configuration

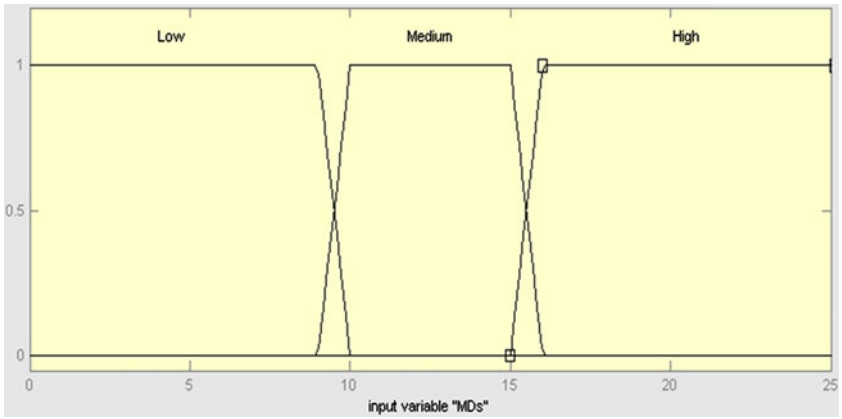


Fig. 10 Number of Associated Terminals initial membership functions configuration

case, each access point monitors its operational environment (Packet Error Rate, Channel Utilization, and Number of Associated Terminals) and attempts to identify potential (high) load situations. If such a problem appears (high load) then they collaborate in order to select the most appropriate configuration action, which in this case is the optimal reallocation of the associated terminals among the available homogeneous or heterogeneous access points in the corresponding network area. The load in injected by the video stream that the terminals associated at the Soekris devices consume.

In our testing facilities we have deployed a heterogeneous wireless network environment consisting of several IEEE 802.11 Soekris access points (AP) and an IEEE 802.16 base station (BS) [27], each embedding an NECM implemented in Java [30]. Soekris devices are low-power, low-cost, advanced communication computers that act as re-

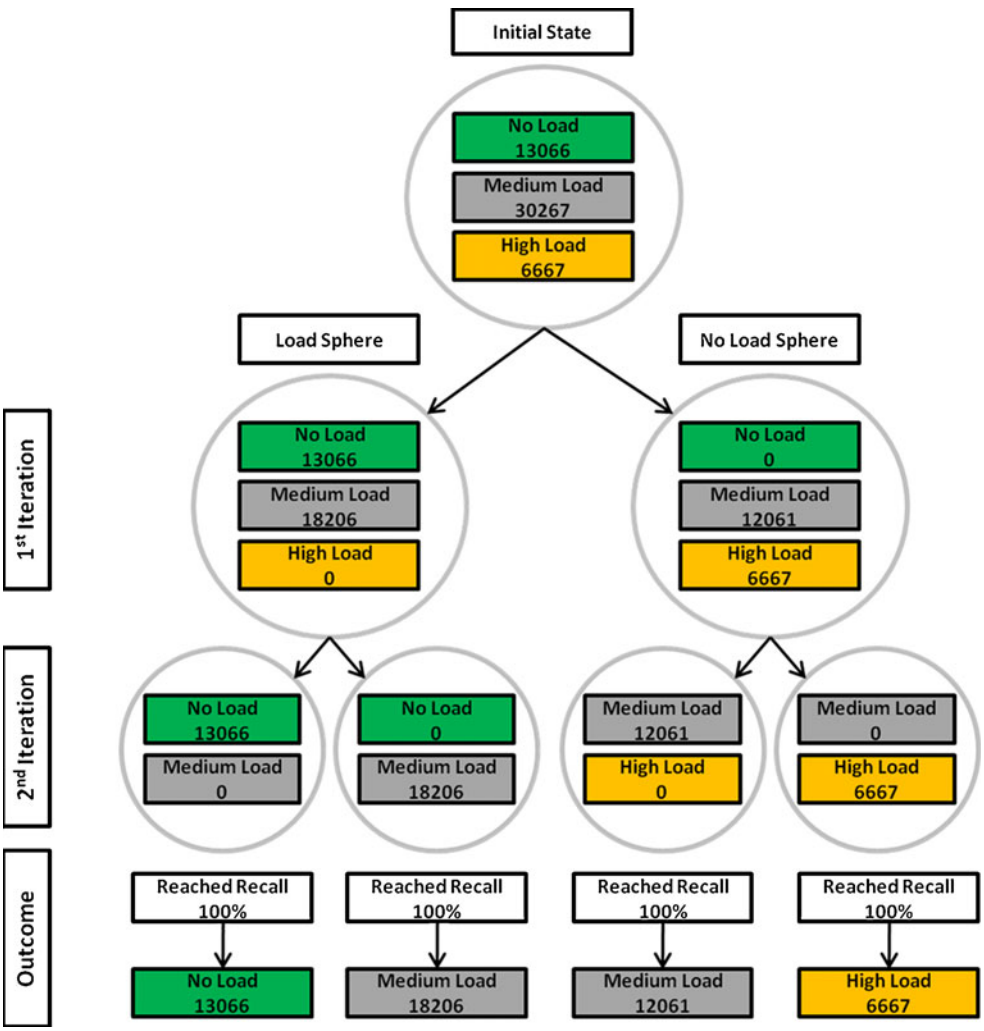


Fig. 11 Dendogram corresponds to FL1 derived after applying our algorithm on the original dataset

programmable 802.11 access routers [28]. Moreover, several single-RAT (i.e. WiFi) and multi-RAT (i.e. WiFi, WiMAX) terminals are located in the corresponding area, consuming a video service delivered by VLC-based service provider [29]. For the cooperation of the underlying NECMs, an NDCMs is deployed. The NDCM is the place where the learning phase operates, based on the data that are provided by the NECMs. Further about the testbed can be found in [25].

This testbed was employed in order to extract the dataset used in the experimental assessment. We collected 50.000 tuples; each tuple was described by three variables indicating the status of an access point at time t_x, namely *Packet Error Rate (PER)* ranging in [0…1], *Channel Utilization (CU)* ranging in [0…1] and *Number of Associated Terminals (AT)* in [0…25]. 10% of the dataset has been sampled from the deployment of the testbed and has been manually labeled (Y_i labeling), while the rest has been artificially generated according to the distribution derived from the initial set. The resulting dataset consists of 6,667 tuples marked as *Load* (*True* according to the algorithmic notation), 9,956 marked as *No Load* (*False*) while the remaining 33,377 correspond to the *Medium Load* case (*Neutral*).

In the first experiment we validate the ability of k-NN to support the derivation of the ground truth label for the first step of the algorithm (Table 2). In order to accomplish that we have experimented with various values for the number of nearest neighbors using the implementation provided by Weka [26]. In most cases where k-NN is employed, the value of k is chosen heuristically. In our case, we have run a number of experiments with k set to 1, 5 and 10. The results have been assessed through a 10-fold cross validation procedure, while all experiments verified our initial intuition regarding the applicability of k-NN in the context of our problem exhibiting, a classification rate larger than 98%. The latter lead us to the additional that any value between 1 and 10 will provide results of adequate quality. The overall results are presented in Table 4.

At a second step, we attempt to validate the effectiveness and efficiency of our approach by simulating a real life situation. We have evaluated all observations with three pre-configured fuzzy logic decision making controllers (FL) implemented in MATLAB [30–33]. The rules used for the

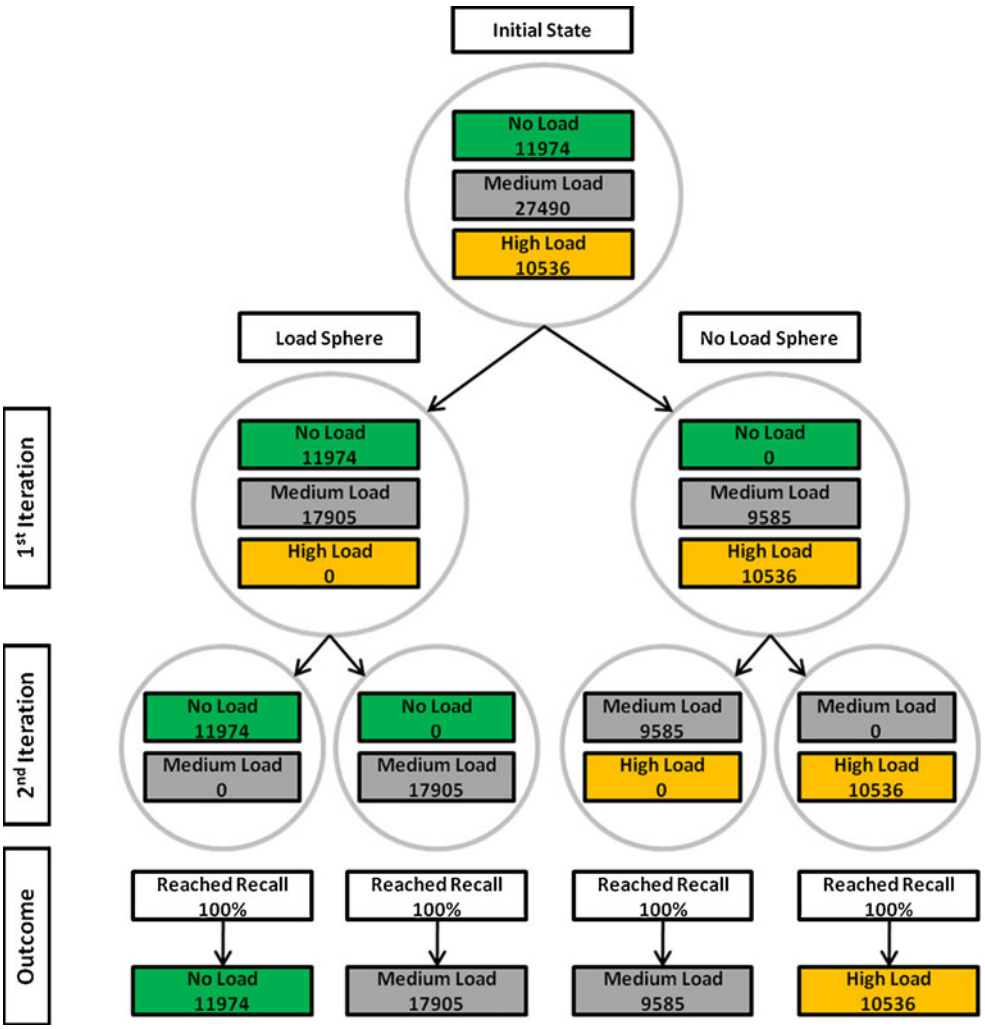


Fig. 12 Dendogram corresponds to FL_2 derived after applying our algorithm on the original dataset.

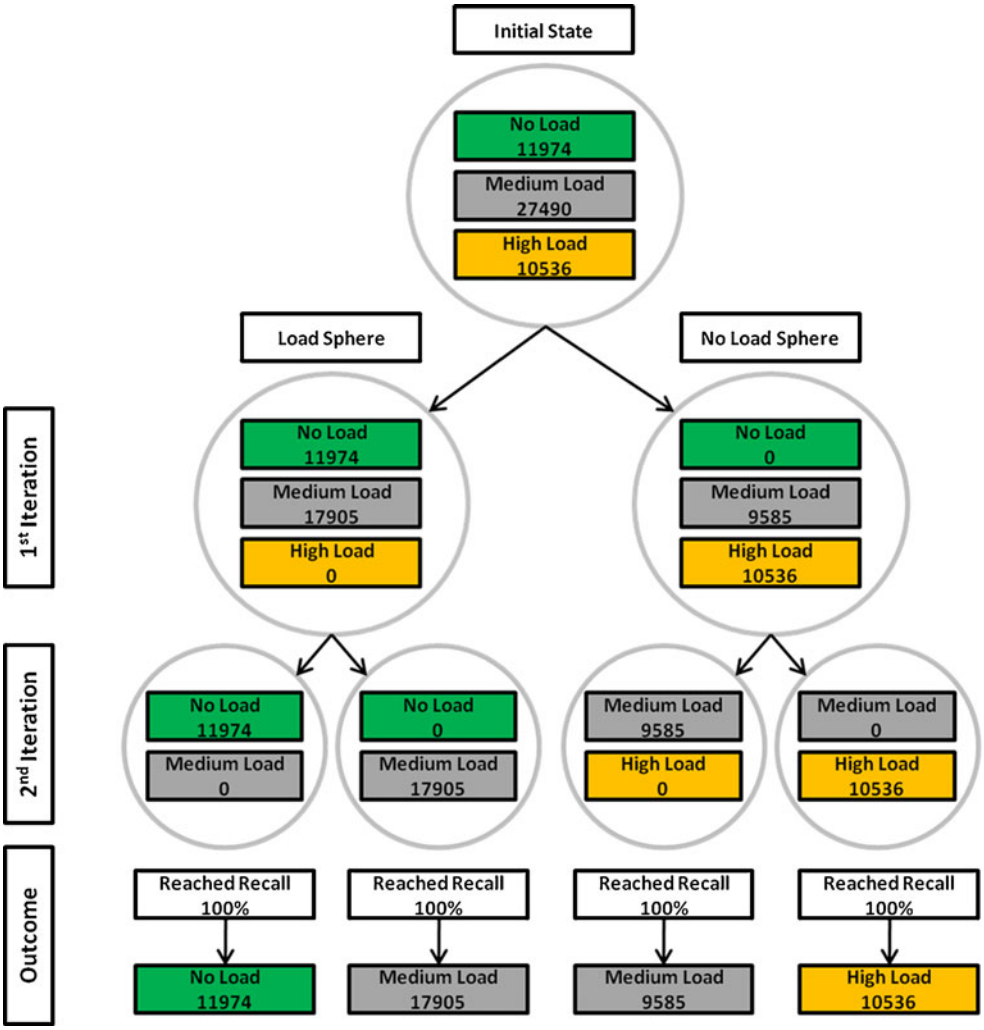


Fig. 13 Dendogram corresponds to FL_3 derived after applying our algorithm on the original dataset

decision making procedure follow the format of equation (7), while membership functions have been configured according to the bounds appearing in Table 5.

IF PER IS low AND CU IS low AND AT IS High THEN Interference IS low, Load IS low (7)

The shape of the membership functions (MF) is related to their special characteristics. More specifically, the *AT* the MFs are trapezoidal. The key characteristic of this MF is its simplicity and is mainly used to describe inputs that have a homogeneity degree and linear behavior. Similarly, for the strict nature of the PER and its relation to the QoS we have decided to use the trapezoidal MFs, which describe in a satisfactory manner the considered error ranges for ideal ("low"), acceptable ("medium") and non acceptable ("high") [31]. Finally, the triangular MFs have been selected for the "Channel Utilization" parameter due to the linear affect of this input to a WiFi AP [32]. In order to test the effectiveness of the proposed solution we have used three initial configurations of the fuzzy logic decision making controller so as to capture more generic and more targeted configurations of the network equipment (Table 5– $FL_{i,}$ where i captures the initial configuration, ranging from 1 -very generic- to 3 -more targeted-). Figures 8, 9 and 10 present the first (more generic) initial configuration of the fuzzy logic controller. Similarly, we have configured the fuzzy logic controller in the other two configurations.

An interesting outcome after observing the original dataset is that the results will be heavily influenced by the values of the *AT* parameter, while on the other hand *PER* seems to provide little information in the clustering process. The latter is due to the fact that these variables are in different scale. Indeed, *AT€N* takes values from the range

Table 6 Classification results after the enhancement of the fuzzy logic rules

	FL_1	FL_2	FL_3
Original	65.64%	71.86%	75.40%
After k-means HDC	76.73%	84.09%	86.06%
Amelioration	16.8%	17.01%	14.13%

Table 7 The bounds extracted from k-Means after clustering on the normalized dataset

	PER	CU	AT	
Low	FL_1	[0 … $3.2*10^{-3}$]	[0 … 0.375]	[0… 5.23]
	FL_2	[0 … $2.83*10^{-3}$]	[0 … 0.337]	[0 … 4.26]
	FL_3	[0… $3.03*10^{-3}$]	[0 … $3.03*10^{-3}$]	[0… $3.03*10^{-3}$]
Medium	FL_1	[$2.87*10^{-3}$ … $1.17*10^{-2}$]	[0.357 … 0.756]	[4.29 … 16.18]
	FL_2	[$2.43*10^{-3}$ … $1.13*10^{-2}$]	[0.312 … 0.741]	[3.03 … 15.74]
	FL_3	[$2.69*10^{-3}$ … $1.10*10^{-2}$]	[0.33… 0.719]	[4.29 … 16.18]
High	FL_1	[$1.16*10^{-2}$ … 1]	[0.747… 1]	[15.89 … 25]
	FL_2	[$1.12*10^{-2}$ … 1]	[0.728 … 1]	[15.3 … 25]
	FL_3	[$1.08*10^{-2}$ … 1]	[0.698 … 1]	[15.89…25]

[0…25] while $PER \epsilon R$ and specifically in [0…1] with the majority of its values concentrated in [0…0.015]. In order to overcome this issue we choose to normalize the values of PER and AT through formulas (8) and (9) respectively.

$$PER_i = \frac{PER_i}{\max_{j=1}^{N}(PER_j)}, i = 1 \ldots N \tag{8}$$

$$AT_i = \frac{AT_i}{\max_{j=1}^{N}(AT_j)}, i = 1 \ldots N \tag{9}$$

Figures 11, 12 and 13 present the results after executing the algorithm on the dataset. The algorithm uses the decision obtained from the fuzzy logic controller and divides the input tuples into three classes according to the identified state of the network element (i.e. *Load, Medium Load, No Load*). Then the tuples identified as *Load* are used as input to the clustering algorithm that iteratively divides the set into two clusters, *Load* and *Medium Load,* until the halting condition is satisfied; the same procedure is being held for the tuples identified as *No Load*. The clustering algorithm has been implemented in Java.[1]

When the halting condition is validated we acquire a good approximation of the actual sets, while any potential overlapping corresponds to the intersection of the membership functions. The bounds obtained from these experiments appear in Table 7. By employing the derived points as the new bounds for the membership functions we apply the algorithm of Table 3 and obtain the classification results appearing in Table 6.

Based on the experiments we conclude that the algorithm performs significantly well and tends to provide rules which converge to decisions closer to the ground truth, independently of the initial configuration of the network element's decision making engine. For the three initial configurations of the fuzzy logic controller the achieved amelioration is of 14–17%. The presented amelioration focuses on the situation awareness of a cognitive manager. The characterization of events, which is a situation awareness phase, is the pilot for the successful optimization or fix of the network system. If the cognitive manager (i.e. NECM) cannot assess effectively the local status, then the performance of the network in many cases will not be improved by applying a reconfiguration action. Thus, the correct labeling of events is an important task for autonomic network management systems, where the learning process has merit (Table 7).

[1] The full dataset, accompanied by the source code used in the experiments and a detailed description of the testbed is available at http://kandalf.di.uoa.gr/MONAMI/

5 Conclusions

In the context of this paper a novel solution, for enhancing the decision making capability of self-managed network elements is proposed. In a future Internet environment, the network elements will be required to operate in diverse and volatile environments. Consequently, additional operational effort is required by network operators and/or manufacturer, since static a priori deduction mechanism and policy deployment is not enough. The proposed feedback solution emerges as a viable answer to the aforementioned problem, enabling network elements to operate having acceptable success rates in their decision making mechanisms. The success of the approach lays in the combination of fuzzy logic with traditional data mining techniques, a key factor for enabling the decision making mechanism to evolve and adjust its operation to a constantly changing network environment. Based on our experimentation the initial deduction of the Inference engine (without Learning) is effective in average between 65% and 75%. In order to assess this percentage as satisfactory or not we have to compare it with a legacy (non-autonomic) system. The experimental analysis showcases a significant amelioration in the classification ability of a network element due to the application of our algorithm, an amelioration which with adequate preprocessing reaches the scale of 14–17%. Future research will concentrate on incorporating more complex high dimensional manifolds in the original algorithm, introducing more complex clustering algorithms for HDC as well as transitioning to a fully autonomous, unsupervised system with no built in rules of other kind of knowledge.

Acknowledgment This work is supported by the European Commission Seventh Framework Programme ICT-2008-224344 through the Self-NET Project (https://www.ict-selfnet.eu). We also wish to thank the special issue editors, as well as the anonymous reviewers for their constructive suggestions and comments.

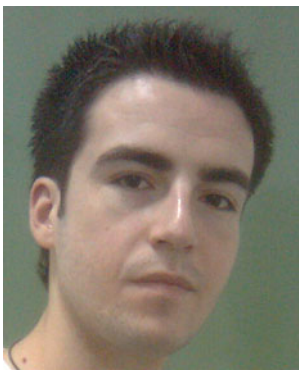
References

1. The Internet Engineering Task Force, http://www.ietf.org
2. 3rd Generation Partnership Project (3GPP), http://www.3gpp.org
3. Distributed Management Task Force, http://www.dmtf.org
4. International Telecommunication Union, http://www.itu.int
5. Siqueira MA, Verdi FL, Pasquini R, Magalhães M (2006). An architecture for autonomic management of ambient networks. Autonomic Networking, (pp. 255–267), Springer
6. Foley C, Balasubramaniam S, Power E, Ponce de Leon M, Botvich D, Dudkowski D, Nunzi G, Mingardi C (2008) A framework for in-network management in heterogeneous future communication networks, modelling autonomic communications environment. Lecture Notes in Computer Science 5276:14–25. doi:10.1007/978-3-540-87355-6_2
7. Jennings B, Van der Meer S, Balasubramaniam S et al (2007) Towards autonomic management of communications networks. IEEE Communications Magazine 45(10):112–121
8. Chaparadza R, Papavassiliou S, Kastrinogiannis T, Toth A, Liakopoulos A, Wilson M (2009) Creating a viable evolution path towards self-managing future internet via a standardizable reference model for autonomic network engineering. Architecture 136–147. doi:10.3233/978-1-60750-007-0-136
9. Kephart JO, Chess DM (2003) The vision of autonomic computing. IEEE Computer 36(1):41–52. doi:10.1109/MC.2003.1160055
10. Dobson S et al (2006) A survey of autonomic communications. ACM Transactions on Autonomous and Adaptive Systems 1 (2):223–259. doi:10.1145/1186778.1186782
11. Mitola J (2000) Cognitive radio: an integrated agent architecture for software defined radio, Ph.D. dissertation, KTH, http://www.lib.kth.se/Fulltext/MITOLA000608.PDF
12. Ryan WT, Friend HD, Dasilva LA, Mackenzie AB (2006) Cognitive networks: adaptation and learning to achieve end-to-end performance objectives. IEEE Communications Magazine 44 (12):51–57. doi:10.1109/MCOM.2006.273099
13. Dietterich T, Langley P (2007) Machine learning for cognitive networks: technology assessment and research challenges. In: Mahmoud Q (ed) Cognitive networks: towards self-aware networks. Wiley, New York
14. Soysal M, Schmidt EG (2010) Machine learning algorithms for accurate flow-based network traffic classification: evaluation and comparison. Performance Evaluation 67(6):451–467. doi:10.1016/j.peva.2010.01.001
15. Bagnasco R, Serrat J (2009) Multi-agent reinforcement learning in network management, scalability of networks and services. Lecture Notes in Computer Science 5637:199–202. doi:10.1007/978-3-642-02627-0_21
16. Han J, Kamber M (2007) Data mining: concepts and techniques, the Morgan Kaufmann series in data management systems
17. Self-NET Project, https://www.ict-selfnet.eu
18. Kousaridas A, Nguengang G, Boite J, Conan V, Gazis V, Raptis T, Alonistioti N (2010) An experimental path towards Self-Management for Future Internet Environments. In: Georgios Tselentis, Alex Galis, Anastasius Gavras, Srdjan Krco, Volkmar Lotz, Elena Simperl, Burkhard Stiller (eds) Towards the Future Internet - Emerging Trends from European Research., pp 95–104
19. Mihailovic A, Chochliouros IP, Kousaridas A, Nguengang G, Polychronopoulos C et al (2009) Architectural Principles for Synergy of Self-management and Future Internet Evolution. ICT Mobile and Wireless Commun. Summit, Santander
20. Merentitis A, Triantafyllopoulou D (2010), Transmission power regulation in cooperative cognitive radio systems under uncertainties, Wireless Pervasive Computing, 5th International Symposium on, doi:10.1109/ISWPC.2010.5483742, Piscataway USA: IEEE Press
21. Clancy C, Hecker J, Stuntebeck E, O'Shea T (2007) Applications of Machine Learning to Cognitive Radio Networks. IEEE Wireless Communications 14(4):47–52
22. Lloyd S (1982) Least squares quantization in pcm. IEEE Transactions on Information Theory 28:129–137
23. H.S. Mahmood and R. Gage (2003). An architecture for integrating cdma2000 and 802.11 WLAN networks. in Proc of IEEE 58th Vehicular Tech. Conference, 2003 (VTC 2003-Fall), vol.3, pp. 2073–2077
24. Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is "nearest neighbor" meaningful? Database Theory Lecture Notes in Computer Science 1540:217–235. doi:10.1007/3-540-49257-7_15
25. Magdalinos P, Makris D, Spapis P, Papazafeiropoulos C, Kousaridas A, Stamatelatos G, Alonistioti N (2010) Coverage and Capacity Optimization of Self-managed Future Internet Wireless Networks, Towards a Service-Based Internet. Lecture Notes in Computer Science 6481:201–202. doi:10.1007/978-3-642-17694-4_23
26. Weka (Waikato Environment for Knowledge Analysis), http://www.cs.waikato.ac.nz/ml/weka/
27. AN-100U/UX Single Sector Wireless Access Base Station User Manual, RedMAX, Redline Communications, 2008
28. Soekris Engineering net5501, http://www.soekris.com/net5501.htm
29. VLC: open-source multimedia framework, player and server, http://www.videolan.org/vlc
30. Java Programming Language, http://www.oracle.com/technetwork/java/index.html
31. Tee A., Cleveland J.R, Chang J.W., Implication of End-user QoS requirements on PHY & MAC, IEEE 802.20 Working Group on Mobile Broadband Wireless Access, http://www.ieee802.org/20/Contribs/C802.20-03-106.ppt
32. Andrews N., Kondareddy Y., Agrawal P. (2010) Channel management in collocated WiFi-WiMAX networks, System Theory, 42nd Southeastern Symposium on, pp. 133–137, doi:10.1109/SSST.2010.5442848
33. Matlab—The Language of Technical Computing, http://www.mathworks.com/products/matlab

Dr. Panagis Magdalinos is a researcher in the SCAN group of Lecturer Nancy Alonistioti, in the Department of Informatics and Telecommunications of the University of Athens (UoA). In 2010 he acquired his PhD diploma entitled "Linear and Non Linear Dimensionality Reduction for Distributed Knowledge Discovery" from the Department of Informatics of the Athens University of Economics and Business (AUEB). He also holds an M.Sc. in Information Systems from AUEB and a B.Sc. in Informatics and Telecommunications from UoA. Since 2004 he has participated in a number of European research projects, namely E²R, E²R II, E3 and SelfNET. His research interests focus on supervised and unsupervised knowledge extraction from distributed data collections (e.g., Learning and Mining in Distributed Environments, Parallel Data Mining).

Mr. Apostolos Kousaridas received his B.Sc. degree in Informatics and his M.Sc. degree in Information Systems from the Department of Informatics at Athens University of Economics and Business. He has worked as a software engineer for the "Greek Research and Technology Network" (GRNET) as well as for the department of Information Systems Design at the Hellenic Railways Organization. He holds an "Ericsson Award of Excellence in Telecommunications" for his M.Sc.Thesis and a performance scholarship from the Greek National Scholarship Foundation. Since mid-2005, he serves as a researcher in the SCAN group, in the University of Athens, and has participated in E²R, E²R II, E3, Self-NET CONSERN and UNIVERSELF EU-funded projects. His main areas of interest are complex self-organizing networks, network economics, e-commerce, and mobile services. He serves as a University of Athens delegate at the ETSI Autonomic Future Internet (AFI) Industry Standardization Group (ISG). He is PhD candidate in the Department of Informatics & Telecommunications at the University of Athens and his field of study is cognitiveand adaptive communications systems.

Mr. Panagiotis Spapis received the diploma in Electrical and Computer Engineering from the University of Patras (UoP), Greece, in 2008. Since September 2008 he serves as research fellow in the SCA-Networking Lab (Self-evolving Congitive & Autonomic Networking) of the National and Kapodistrian University of Athens (NKUA) and has participated in several EU funded FP7 research projects (E3, Self-NET, CONSERN, UNIVERSELF). His main research interests lie in decision making and learning techniques and their application in cognitive networks. Parts of his work has been publicated in several conferences. He is currently pursuing his PhD in the above thematic areas. Also, he is a member of the Technical Chamber of Greece since November 2008.



Mr. George Katsikas has received his B.Sc. degree from the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens (NKUA) in June 2010 and he is currently pursuing his M.Sc studies in Communication Systems and Networks in the same department. He serves as a research fellow in the Self-Evolving Cognitive and Autonomic Networking (SCAN) Group of NKUA and participates in Self-NET, Co-N-Sern and UniverSelf EU-funded ICT Projects. His main research interests are Network Management, Machine Learning, B3G Communication Systems and Software Engineering.

Lecturer Nancy Alonistioti has a B.Sc. degree and a PhD degree in Informatics and Telecommunications (Dept. of Informatics and Telecommunications, University of Athens). She has working experience as senior researcher and project manager in the Dept. of Informatics and Telecommunications at University of Athens. She has participated in several national and European projects, (CTS, SS#7, ACTS RAINBOW, EURESCOM, MOBIVAS, ANWIRE, E²R, LIAISON, E³, SELFNET, SACRA, CONSERN, UNIVERSELF etc) and has experience as Project and Technical manager of the IST-MOBIVAS, IST-ANWIRE, ICT-SELFNET projects, which had a focus on reconfigurable mobile systems, cognitive mobile networks and FI. She has served as PMT member and WP Leader of the FP6 IST- E²R project. She is co-editor and author in "Software defined radio, Architectures, Systems and Functions", published by John Wiley in May 2003. She has served as lecturer in University of Piraeus and she has recently joined the faculty of Dept. Informatics and Telecommunications of Univ. of Athens. She is TPC member in many conferences in the area of mobile communications and mobile applications for systems and networks beyond 3G. She has over 55 publications in the area of mobile communications, reconfigurable, cognitive and autonomic systems and networks and Future Internet.