Proceedings of the 10th WSEAS International Conference on COMMUNICATIONS, Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp512-518)

Artificial Neural Networks Applied to the Development of Agents for Network Management: Simulation Results

Analúcia Schiaffino Morales De Franceschi[1], Karen Selbach Borges[1], Ricardo Moraes[2], Francisco Vasques[2]
[1]Universidade Luterana do Brasil (ULBRA)
Curso de Ciência da Computação
Grupo de Redes e Telecomunicações
Av. Farroupilha, 8001 – São Luis – Canoas/RS - Brasil

[2]Universidade do Porto
Faculdade de Engenharia
Rua Dr. Roberto Frias s/n – 4200-465 – Porto - Portugal

Abstract. Agents´ paradigms are very popular due to their flexibility, modularity and general applicability to a very wide range of problems. Nowadays, the majority of the agents used in computer networks are passive agents, i.e., the agent has no autonomy, and there is no participation in decision-making actions. In this paper, we present a methodology for agent's development with the employment of artificial intelligence techniques. The methodology has been applied to the baselines development for proactive network management. In preliminary tests, three different recurrent ANNs had been examined; where we have shown that the proposed methodology is adequate to support the active agent´s development.

Key-Words. Intelligent agents, distributed problems, neural networks, proactive management, SNMP.

1 Introduction

Currently, modern Local Area Networks (LANs) are so complex to manage as Wide Area Networks (WANs) [1]. In network management environments there are three important concepts to consider: the managers, the agents and the management objects. These concepts were presented initially in both CMIP (*Common Management Information Protocol*) and the SNMP (*Simple Network Management Protocol*) protocols.

A Network Management System (NMS) consists in a set of software and hardware tools for the network management. Nowadays, as far we know, the managers and the agents implemented in commercial NMSs have no autonomy; they have no participation in the decision-making process. In general, the decisions about the network management are made by the network administrators, who have to deal with urgent requests to solve network failures.

Commonly, the network management is performed in a centralized way [2][3], where both passive managers and agents have been found. Furthermore, it is also found an approach based on policies; these policies are transformed in production rules and have been associated with JESS (*Java Expert System Shell*) [4].

Summing up, the majority of the agents used in computer networks are passive agents. However, the agents´ paradigms are very popular due to their flexibility, modularity and general applicability to a very wide range of problems [5]. There has been a tremendous growth in computer systems that can be viewed as autonomous agents. The agents are being considered as a new theoretical model of computation that more closely reflects the current computing reality than the Turing Machines [6]. Furthermore, the technological developments in distributed computing, robotics and the emergence of object-oriented programming paradigms have been responsible for the increase of this popularity.

For Lesser [7] the agent autonomy is directly associated with the make decisions about the activities that should be performed, about the time they should be accomplished, and about the way that the same agent can manipulate the informations received by other agents. The autonomy may be defined through some policies built into the agent, or from the results of the cooperative interacting with other agents.

In this way, the inteligent agents have been implemented through the Artificial Inteligence (AI) techniques. The autonomy of the agents may be developed using the following AI techniques: Artificial Neural Network (ANN), CBR (Case-based Reasoning) and Evolutionary Computation (EC). The fetaures of the ANN have been studied and encourage the problem-solving complex applications. The non-linearity is a very important feature, mainly if the physical dispose was not linear, such as the case of the computer networks.

As mentioned before, most part of the solutions based on agents have no autonomy in the management network processes. In order to overcome this limitation, we propose the use of ANN to improve the network management. The ANN has been used to learn the knowledgement from the network through examples. Basically, we first select a set of samples from the network, which will be the input data of the ANN; such input data are related to synaptic weight. Afterwards, these initial weights will be (vai sendo) modified in order to approximate the input to the desired

output. The ANN training is repeated for all set of samples until the ANN becomes stable, what means that the weights have no more modifications [8].
This technique will provides in the next generation of network systems the possibility of the system automatically configure itself from the feature of the network.
The remainder of the paper is structured as follows. Section 2 discuss the methodology employed and the behaviors of the management network systems. Section 3 presents the test environment. Section 4 shows the preliminary results. Finally, the Summary and future works advocates a new approach to the results employement.

2 Agent´s Methodology in NMS

While analyzing the network management process in the whole, a complex problem has emerged. Due to that fact, we have proposed a methodology to develop new applications for network management. This methodology employs agents and artificial intelligence techniques. According to this methodology, there are two kinds of agents to be developed: static or dynamic. In the static agent implementations can be used production rules or feedforward neural networks. Such implementation can be done through *heuristics* obtained from an expert such as a network administrator. This approach allows limited solutions.
Nevertheless, in computer network management, the main agents have a dynamic behavior and the implementation of dynamic agents is more complex. In [10], Barreto proved that the recurrent networks must be applied to reach the solution when problems are classified as dynamic. The static solution applied to a dynamic problem causes two inconvenients: firstly, the static solution cannot reach all the different states found in the dynamic problems; secondly, the state of the dynamic system must be supplied, either explicitly or implicitly, with input data in order to train the NN. However, this procedure leads to a very large NN and to a longer trainning time. Because in this case is necessary to describe all the possible states, so that the neural network can converge to a correct solution.
The static and dynamic behaviors are directly related to reactive or proactive solutions (Figure 1).

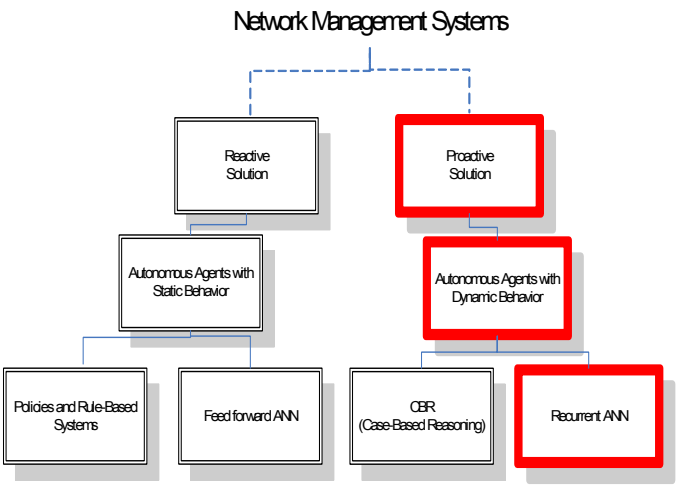

Figure 1. Methodology application

2.1 Reactive Solutions

The diagnostic system is possibly the most commonly used reactive solution in the management network area. In this kind of solution, the characteristics of the problems are input to the system and the problem diagnosis is the output [12]. However, the actions for solving those problems are taken only after the network degradation, which is the main disadvantage of this kind of solution. The diagnostic systems are normally used in the fault management area. Some faults, such as the poor quality of the equipment or the cables´ exposure to the physical environmental conditions are impossible to be prevented [13]. This system may be implemented through the heuristics obtained from one or more network administrators. The first step is to provide a set of features about the environment faults. Normally, the faults are classified as the network layers: physical layer faults, link layer faults, network layer faults, transport layer faults and application layer faults. However, this approach has a specific problem: the static feature. If any aspect of the network were modified, the system would be unusable. This problem was observed in the expert systems developed along the years for the network management area. The tools employed to develop such a system are production rules and feedforward neural networks. The production rules are a declarative solution implemented with the words *if* and *then*. Using feedforward NNs, the input and output patterns are known. In this case, the network is trained to learn the patterns and it has no state changes [14]. These systems are appropriate for small environments or networks with few resources, for which little expansion is predicted.

2.2 Proactive Solutions

When the problem needs a dynamic solution, a dynamic behavior is expected from the agents. To include dynamism in a NN solution, it is necessary to apply a sequential line of time delays between two inputs of a feedforward NN, or using a network with cycles and dynamical neurons (e.g.: Hopfield network and recurrent NN). The methodology based on recurrent NNs applied to the computer network management area was introduced in [10]. Figure 2 shows a recurrent ANN similar to the one employed in this work.

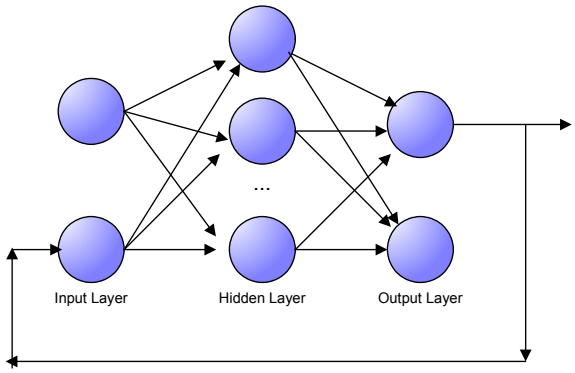

Figure 2 – Recurrent neural network employed

The main goal of the proactive management is detecting possible problems (in a network) before they happen and lost the network active state. Roisenberg [15] presented a

formal concept to autonomous agents using recurrent neural networks based on system theory.

3 Proposed approaches for the NMS

The most part of network management systems (NMSs) have no intelligence to configure the computer networks parameters and make-decisions when failures occur, in an autonomous way.

Normally, the both proprietary and freeware network management systems available in the market, only collect the informations from the network environments and generate reports. Besides, the network administrators, who have set the all parameters to start the management process, do the entire system configuration.

In order to overcome this limitation, the neural networks may be the solution to provide proactive network management. Figure 3 illustrates the two approaches for the network management systems proposed in Section 2. On the left, it is shown the most used methodology; where the SNMP monitor the network and both data and events are reported to the network administrator, who should make decisions. On the right, it is presented the methodology that has been proposed in this work, where the ANN data mining are used. The results have been reported the useful information for the decision-making process.

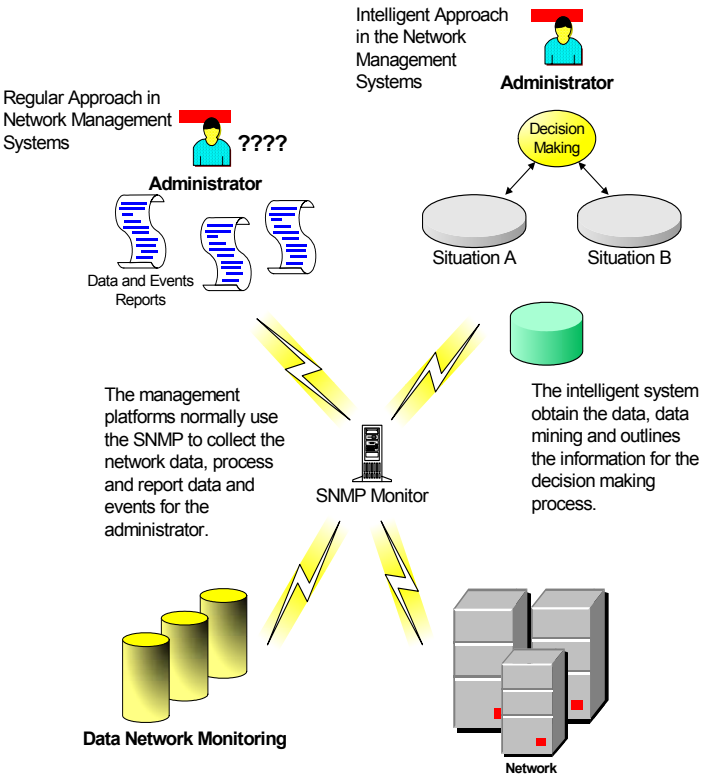

Figure 3 – Two approaches in the network management systems

In such methodology the environment must be configurated to collect examples from the monitored network and the system will be able to analyze the examples to present a diagnosis, i.e., instead of presenting only data to the administrator, the solution should analyze the information and indicate the actions that the network administrator must taken in the network management process.

To reach this goal, we developed a hybrid system, employing the Recurrent NN to recognize the normal profile of network, usually called *baseline*. Afterwards, another kind of NN is used to decision-making process.

In traditional ways, baselining a computer network profile is a difficult task to be accomplished, where a lot of sample activities are required for a long period of time and the normal profile is identified by averages and statistics. On the other hand, the development process of the baselines is easier when a recurrent NN is used correctly. In this process an ANN is developed, which will be trained using examples collected from the network. The examples are input to the input layers and the output layer will provide a function that represents the normal network behavior.

4 Test Environment

In this section, we present a test-bed to evaluate the proposed methology. The experiments were made using samples collected from three interfaces of a CISCO AS 2511 router: one 10 Mbps Ethernet interface and two 2 Mbps WANs interfaces. The MIB-II Interface group objects stored were the *ifInOctets*, which is total number of octets received on the interface, including framing characters; the *ifOutOctets*, which is the total number of octets transmitted out of the interface, including framing characters and; *ifSpeed*, which store an estimate of the interface's current bandwidth in bits per second (Table 1). These variables have been used to estimate the data link utilization rate. There were collected 288 samples a day with five minutes interval between them, during the interval of two weeks.

Table 1. MIB Objects

Identifier (OID)	Description	Data format (ASN.1)
1.3.6.1.2.1.2.2.1.10	ifInOctets	Counter 32
1.3.6.1.2.1.2.2.1.16	ifOutOctets	Counter 32
1.3.6.1.2.1.2.2.1.5	ifSpeed	Gauge

We tested three different topologies of recurrent neural networks (RNN) alternating their features. These neural networks are able to approximate an arbitrary nonlinear function to any desired degree of accuracy. The dynamism is represented through the cycles that give feedback in all of implemented networks.

The first RNN (Figure 4) has one input neuron, one hidden neuron and one output neuron. The hidden layer was defined with 500 sigmoidal units of neurons.

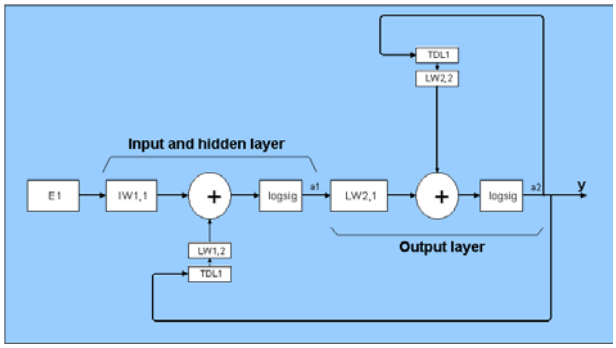

Figure 4 - Recurrent NN.

The second RNN has three input neurons and three hidden neurons. The three hidden layers was defined with 500 sigmoidal units of neurons. The output of neurons has a sigmoid activation function and has connection with the output layer. And another three recurrent connections, one for each hidden neurons. Finally, the third tested RNN has five input neurons, five hidden neurons and one output (Figure 5). The five hidden layers was defined with 500 sigmoidal units of neurons. It has five cycles to feedback, one with the output and other five with the hidden neurons.

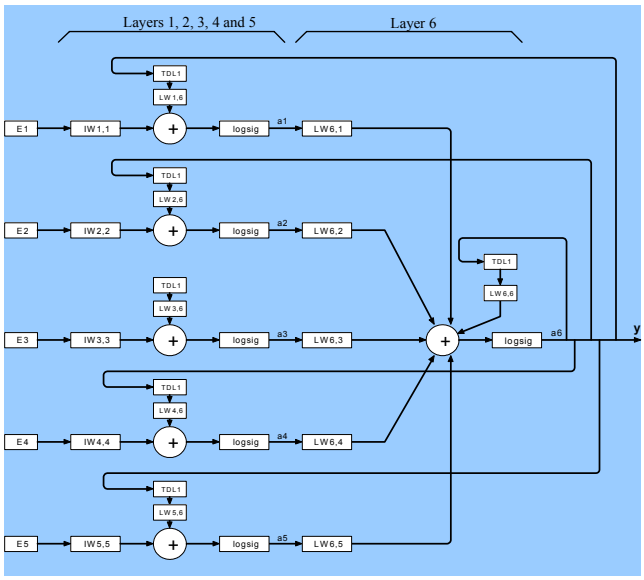

Figure 5 – Five input and hidden layers and one output layer.

5 Preliminary results

As mentioned in section 4, three different topologies of recurrent neural network were tested. Firstly, each RNN was trained using examples; the goal of this process was to obtain a function that would represent the way the system works based on degree of accuracy.

Figures 6, 7 and 8 show the behavior of the training when illustrating the goal error rate *vs.* the number of epochs. Figure 6 represents the RNN with 1-input and 1-hidden layer. Figure 7 shows the behavior of the training when the RNN with 3-inputs and 3-hidden layers is used. Finally, Figure 8 represents the RNN with 5-inputs and 5-hidden layers.

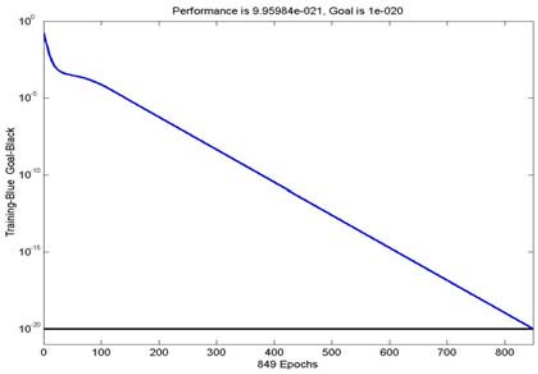

Figure 6 - The relationship between the goal training and the number of epochs (1 layer-1 hidden-1output)

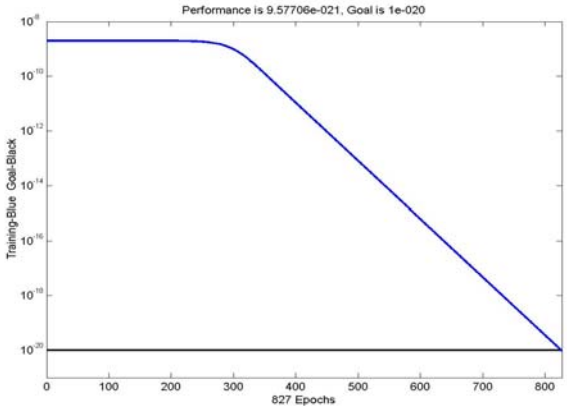

Figure 7 – The relationship between the goal training and the number of epochs (3 layers - 3 hiddens - 1output)

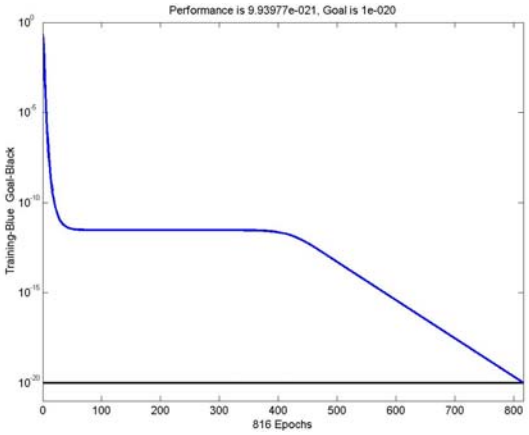

Figure 8 - The relationship between the goal training and the number of epochs (5 layers - 5 hiddens- 1output)

There are two alternatives to stop the training process of NN: the number of epochs and the error goal. An epoch is a time for the neural network interaction that is the sum of the synaptic weights in agreement with the input values. The neural network has been trained during several epochs using the layers and the transfer function for adapting the learning. In this paper, the NNs are trained through the sum of the current and previous inputs, using a feedback of the input data to create the adapted output. This output will be adjusted through the decreasing of the error goal.

The goal of the learning process is to evaluate the probability that the model has generated from the input data, and in this case, the numbers of examples are used to increase the quality of the estimation performed by the model.

Analyzing the Figures 6, 7 and 8 is possible to observe the initial error goal rate and, the error reached in the end of the training process. As it was expected, the smaller number of epochs necessary for training was obtained in the case of neural network with five input layers and five hidden layers. However, it was not obtained significant different values related to number of epochs. In all of the training tests the neural networks spend less than a thousand times of epochs. The three tested RNN learned the data set through the error goal. The epochs spent for the training were respectively 849, 827 and 816 epochs.

The next experiment simulates the neural network. This is a very important step because it is used to verify if the neural network really learned from the examples.

The simultation function represents the data with the calculated weights during the training. Then, in order to evaluate if the system was able to learn the behavior of the network through examples during the training process, the function obtained was simulated with the examples. When the network was simulated with the same samples we must to generate in the output the same behavior. Figures 9, 10 and 11 ilustrate that in all cases the tested neural networks were able to learn the examples. The results obtained in the training process are equal to the results obtained in the simulating process (lines are superposed). We explore the efficiency of the proposed RNN running several simulations.

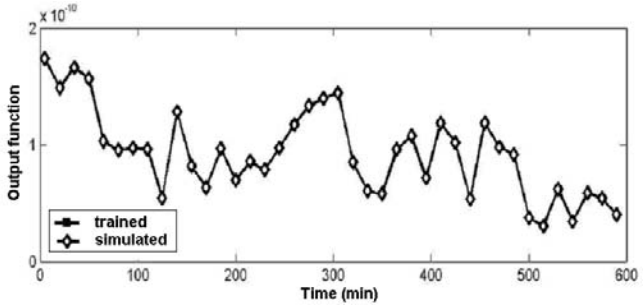

Figure 9 – The results of the RNN with 1input - 1hidden – 1output layer

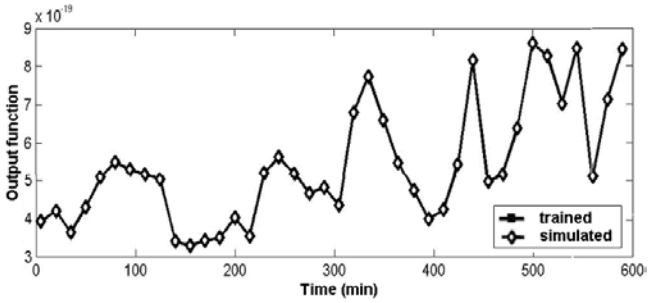

Figure 10 – The results of the RNN with 3 inputs - 3 hidden layers

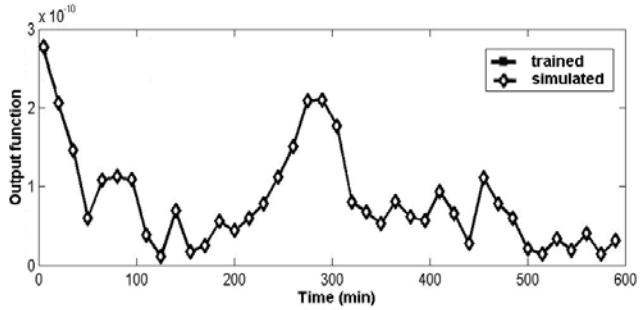

Figure 11 – The results of the RNN with 5 inputs - 5 hidden layer

In the next experiment, the RNN have been simulated with samples collected from a different date in the same data-link interface.

Figure 12 shows two lines. The black dotted line represents the behavior of the training data set; the broken dotted line has been outlined by the simulation of the data in a different date as mentioned above. Both lines have some dotted similarity, as expected. Figure 13 illustrates the training and the simulation tests reached by the RNN with five input neurons and five hidden neurons.

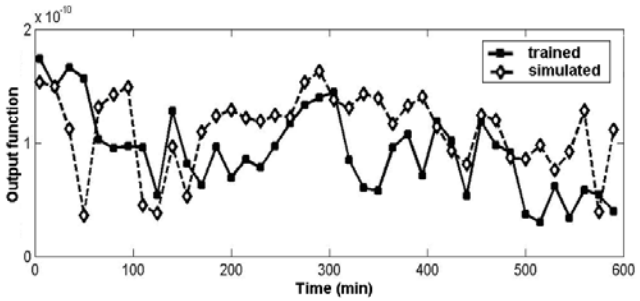

Figure 12 **–** The results with samples collected in different dates

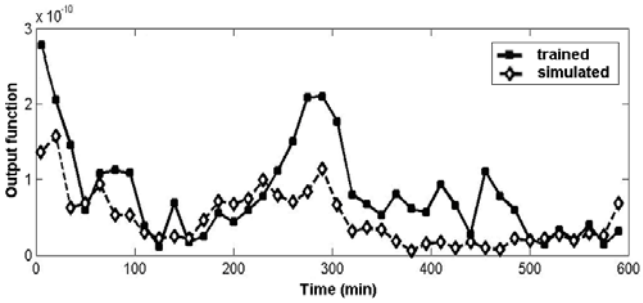

Figure 13 - The results trained and simulated

Finally, the last experiment with the three topologies of RNN has used a data set collected from another kind of data-link interface. In this test, the result may be observed in the figures 14 and 15. The dotted and broken lines must be distant, as expected.

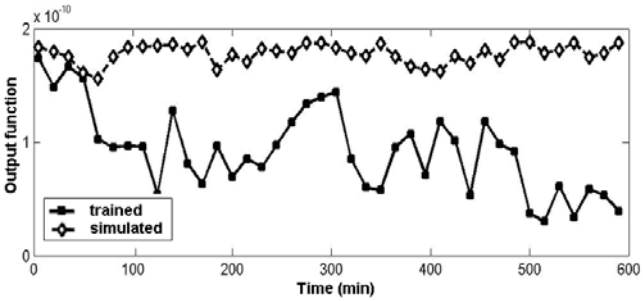

Figure 14 – The tests with two different interfaces using RNN with 1-input and 1-hidden layer

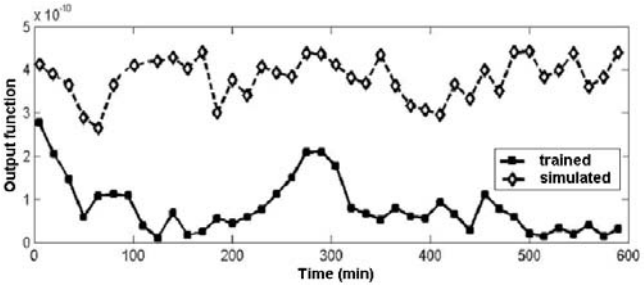

Figure 15 - The tests with two different interfaces using RNN with 5-inputs and 5-hidden layers

6 Future work

According to Haykin [8], for the pattern classification, the first and more important step is the selection of characteristics using not supervised mode. The first mentioned step was reached through the described

experiments presented in this work. This issue was reached by using Artificial Recurrent Neural Networks. The set of data obtained with the training of the RNNs will be used for creating a classification for the graph outlines, producing a hybrid network.

The challenge at this moment is how to classify what it is normal and what is abnormal using the output already generated from neural networks. For this stage it is intended to supply the data of exit of the nets as input in a self-organized map associate to a project of supervised learning of form if to get one better performance as mentioned by Haykin. Figure 16 illustrates the scheme to create a pattern classification that may be used to identify the normal and abnormal behavior from these baseline results. We will employ a self-organized map to group the similar points and in third moment we will develop a feedforward network to classify these points.

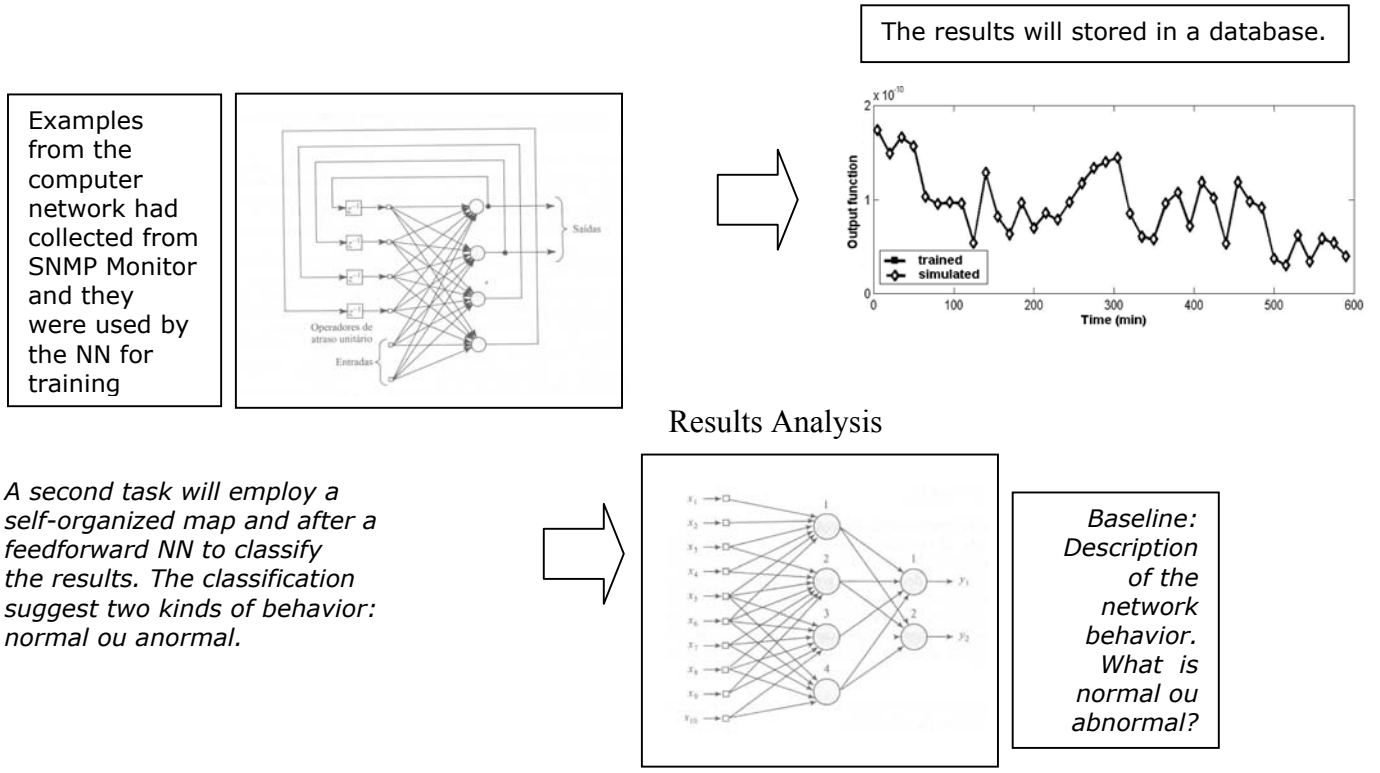

Figure 16 – The scheme for the work development

7 Conclusions

The network management claims skilled personnel, able to detect, diagnose and correct problems quickly and accurately, preferably before they affect the user community [12]. With the correct employment of the AI techniques, if the input and output of neural network were known, we may use a feedforward neural network. This network may be trained estimating the unknown state changes. If there are only input samples, the recurrent neural network must be able to provide the output function using the adaptative learning. The experiments were performed applying the functions and objects from MatLab.

This work outlines the application of the recurrent neural networks in the baseline development of the computer network utilization rate. The different experiment through three different topologies of neural networks allow to conclude some issues about their usage. In our experiment the obtained results are very similar. Since the training until the simulation tests with data set collected in the same interface or the results obtained with the data set of the different data-link. These results allow to conclude that any topology of recurrent neural networks is capable to learn from examples. The baseline was shaped using some examples. The next step will be to develop the network to classify the results.

The future works will research other applications for the network management. New tests and topologies have been developed in different functional areas such as security management. Another possible approach to improve this work is the development of a new intelligent protocol, such as an SNMP with artificial intelligence resources.

References

[1] A. C. Vieira Junior and M. L. Anido, "The architecture of a novel tool for network management using GSM/GPRS mobile devices," presented at Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, 2004.

[2] G. Goldzsmith, Y. Yemini. “Evaluation Management Decisions via Delegation.” *In: Proceedings of*

IFIP/ISINM'93 – International Symposium on Integrated Network Management, Apr., 1993.

[3] J.L.de C. e Silva, A.N. da Silva, J. N. de Souza. "Applying Cooperative Remote Objects and Mobile Agents Concepts for Management by Delegation in Heterogeneous Network Systems". *In: Proceedings of International Conference on Telecommunications – ICT'99*", Vol. II, Cheju, Korea, 1999.

[4] J.P. Bigus, J. Bigus. *"Constructing Intelligent Agents Using Java*". Second Edition. John Wiley and Sons, Inc.: USA, 2001.

[5] C.C. Hayes. "Agents in A Nutshell – A very Brief Introduction". *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, Jan./Fev., 1999. Pp.127-132.

[6] N. R. Jennings, "On agent-based software engineering". Artificial Intelligence, no. 117, p. 277-296, Elsevier Science, 2000.

[7] V.R. Lesser. "Cooperative Multiagent Systems: A Personal View of the State of the Art". *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, Jan./Fev., 1999. Pp.133-142.

[8] HAYKIN, S. *Neural Networks*, 2001.

[9] M.L. Minsky, S.A. Papert. *Perceptrons: an introduction to computational geometry*, MIT Press, 1988

[10] J. M. Barreto. "*Conexionismo e Resolução de Problemas*". Titular professor contest dissertation, Universidade Federal de Santa Catarina, Informatics and Statistics Department, Florianopolis, SC, 1996.

[11] A.S.M. De Franceschi, J.M. Barreto. "Distributed Problem Solving Based on Recurrent Neural Networks Applied to Computer Network Management". *In: Proceedings of International Conference on Telecommunications – ICT'99*", Vol. II, Cheju, Korea, 1999

[12] A. Leiwand, K.F. *Conroy Network Management – A Practical Perspective*. Addison-Wesley, 2nd ed., 1996.

[13] R. A. Maxion, F. E. Feather. A Case Study of Ethernet Anomalies in a Distributed Computing Environment, *IEEE Transactions on Reliability*, vol.39, no.4, Oct, 1990.

[14] A. S. M. De Franceschi, M. A. da Rocha, H. L. Weber, C. B. Westphall. "Employing Remote Monitoring and Artificial Intelligence Techniques to Develop the Proactive Network Management". *In: Proceedings of IEEE International Workshop on Application of Neural Networks in Telecommunications*, Melbourne, Australia, 1997. pp.116-123.

[15] M. Roisenberg, J. M. Barreto, F. M. de Azevedo, L. M. Brazil. "On a Formal Concept of Autonomous Agents". *In: 16th IASTED International Conference Applied Informatics*, Germany, February, 1998.

[16] H. Demuth, M. Beale. *Neural Network Toolbox For Use with MATLAB® - Users Guide Version 4*. (ISBN 0-534-95049-3). USA : The Math Works, 2001.