1

Neural Machine Learning Approaches: Q-Learning and Complexity Estimation Based Information Processing System

Abdennasser Chebira, Abdelhamid Mellouk,
Kurosh Madani and Said Hoceini
LISSI laboratory, University Paris 12-Val de Marne
France

1. Introduction

Real world dilemmas, and especially industry related ones, are set apart from academic ones from several basic points of views. The difference appears since definition of the "problem's solution" notion. In fact, academic (called also sometime theoretical) approach often begins by problem's constraints simplification in order to obtain a "solvable" model (here, solvable model means a set of mathematically solvable relations or equations describing a behavior, phenomena, etc…) (Madani, 2008). If the theoretical consideration is a mandatory step to study a given problem's solvability, for a very large number of real world dilemmas, it doesn't lead to a solvable or realistic solution. Difficulty could be related to several issues among which:

- large number of parameters to be taken into account (influencing the behavior) making conventional mathematical tools inefficient,
- strong nonlinearity of the system (or behavior), leading to unsolvable equations,
- partial or total inaccessibility of system's relevant features, making the model insignificant,
- subjective nature of relevant features, parameters or data, making the processing of such data or parameters difficult in the frame of conventional quantification,
- necessity of expert's knowledge, or heuristic information consideration,
- imprecise information or data leakage.

Examples illustrating the above-mentioned difficulties are numerous and may concern various areas of real world or industrial applications. As first example, one can emphasize difficulties related to economical and financial modeling and prediction, where the large number of parameters, on the one hand, and human related factors, on the other hand, make related real world problems among the most difficult to solve. Another illustrative example concerns the delicate class of dilemmas dealing with complex data's and multifaceted information's processing, especially when processed information (representing patterns, signals, images, etc.) are strongly noisy or involve deficient data. In fact, real world and industrial applications, comprising system identification, industrial processes control, systems and plants safety, manufacturing regulation and optimization, pattern recognition, communication networks (complex routing, large communication networks management

and optimization, etc.) (Mellouk, 2008a), are often those belonging to such class of dilemmas.

If much is still to discover about how the animal's brain trains and self-organizes itself in order to process so various and so complex information, a number of recent advances in "neurobiology" allow already highlighting some of key mechanisms of this marvels machine. Among them one can emphasizes brain's "modular" structure and its "self-organizing" capabilities. In fact, if our simple and inappropriate binary technology remains too primitive to achieve the processing ability of these marvels mechanisms, a number of those highlighted points could already be sources of inspiration for designing new machine learning approaches leading to higher levels of artificial systems' intelligence (Madani, 2007).

In this chapter, we deal with machine learning based modular approaches which could offer powerful solutions to overcome processing difficulties in the aforementioned frame. If the machine learning capability provides processing system's adaptability and offers an appealing alternative for fashioning the processing technique adequacy, the modularity may result on a substantial reduction of treatment's complexity. In fact, the modularity issued complexity reduction may be obtained from several instances: it may result from distribution of computational effort on several modules; it can emerge from cooperative or concurrent contribution of several processing modules in handling a same task; it may drop from the modules' complementary contribution (e.g. specialization of a module on treating a given task to be performed).

A number of works dealing with modular computing and issued architectures have been proposed since 1990. Most of them associate a set of Artificial Neural Networks (ANN) in a modular structure in order to process a complex task by dividing it into several simpler sub-tasks. One can mention active learning approaches (Fahlman & Lebiere, 1990), neural networks ensemble concept proposed by (Hanibal, 1993), intelligent hybrid systems (Krogh & Vedelsby, 1995), Mixture of experts concept proposed by (Bruske & Sommer, 1995) and (Sung & Niyogi, 1995) or structures based on dynamic cells (Lang & Witbrock, 1998). In the same years, a number of authors proposed multi-modeling concept for nonlinear systems modeling, where a set of simple models is used to sculpt a complex behaviour (Goonnatilake & Khebbal, 1996), (Mayoubi et al., 1995), (Murray-Smith & Johansen, 1997), (Ernst, 1998)) in order to avoid difficulties (modeling complexity). However, it is important to remind that the most of proposed works (except those described in the four latest references) remain essentially theoretical and if a relatively consequent number of different structures have been proposed, a very few of them have been applied to real-world dilemmas solution.

The present chapter focuses those machine learning based modular approaches which take advantage either from modules' independence (multi-agent approach) or from self-organizing multi-modeling ("divide and conquer" paradigm). In other words, we will expound online and self-organizing approaches which are used when no a priori learning information is available. Within this frame, we will present, detail and discuss two challenging applicative aspects: the first one dealing with routing optimization in high speed communication networks and the other with complex information processing. Concerning the network routing optimization problem, we will describe and evaluate an adaptive online machine learning based approach, combining multi-agent based modularity and neural network based reinforcement learning ((Mellouk, 2007), (Mellouk, 2008b)). On the side of complex information processing, we will describe and evaluate a self-organizing

modular machine learning approach, combining "divide and conquer" paradigm and "complexity estimation" techniques that we called self-organizing "Tree-like Divide To Simplify" (T-DTS) approach ((Madani et al., 2003), (Madani et al., 2005), (Bouyoucef et al., 2005), (Chebira et al., 2006)).
This chapter is composed by four sections. The second section presents the state of the art of modular approaches over three modular paradigms: "divide and conquer" paradigm, Committee Machines and Multi Agent systems. In section 3, a neural network based reinforcement learning approach dealing with adaptive routing in communication networks is presented. In the last section, dealing with complex information processing, we will detail the self-organizing Tree divide to simplify approach, including methods and strategies for building the modular structure, decomposition of databases and finally processing. A sub-section will present a number of aspects relating "complexity estimation" that is used in T-DTS in order to self-organize such modular structure. Evaluating the universality of T-DTS approach, by showing its applicability to different classes of problems will concern other sub-sections of this fourth section. Global conclusions end this chapter and give further perspectives for the future development of proposed approaches.

2. Modular approaches

Apart from specialized "one-piece" algorithm as explicit solution of a problem, there exist a number of alternative solutions, which promote modular structure. In modular structure, units (computational unit or model) could either have some defined and regularized connectivity or be more or less randomly linked, ending up at completely independent and individual units. The units can communicate with each others. The units' communication may take various forms. It may consist of data exchange. It may consist of orders exchange, resulting either on module's features modification or on its structure. Units may espouse cooperative or competitive interaction. A modular structure composed of Artificial Neural Networks is called Multi Neural Network (MNN).
We will present here three modular paradigms that are of particular interest: "Divide and Conquer" paradigm, Committee Machines and Multi Agent Systems. "Divide and conquer" paradigm is certainly a leading idea for the tree structure described in this section. Committee machines are in large part incorporation of this paradigm. For multi-agent approach the stress is put on the modules independence.

2.1 "Divide and Conquer" paradigms

This approach is based on the principle "Divide et Impera" (Julius Caesar). The main frame of the principle can be expressed as:

- Break up problem into two (or more) smaller sub-problems;
- Solve sub-problems;
- Combine results to produce a solution to original problem.

The ways in which the original problem is split differ as well as the algorithms of solving sub-problems and combining the partial solutions. The splitting of the problem can be done in recursive way. Very known algorithm using this paradigm is Quicksort (Hoare, 1962), which splits recursively data in order to sort them in a defined order. In the Artificial Neural Networks area the most known algorithm of similar structure is Mixture of Experts (Bruske & Sommer, 1995).

Algorithmic paradigms evaluation could be made on the basis of running time. This is useful in that it allows computational effort comparisons between the performances of two algorithms to be made. For Divide-and-Conquer algorithms the running time is mainly affected by:

- The number of sub-instances into which a problem is split;
- The ratio of initial problem size to sub-problem size;
- The number of steps required to divide the initial instance and to combine sub-solutions;
- Task complexity;
- Database size.

2.2 Committee machines

The committee machines are based on engineering principle divide and conquer. According to that rule, a complex computational task is solved by dividing it into a number of computationally simple sub-tasks and then combining the solutions of these sub-tasks. In supervised learning, the task is distributed among a number of experts. The combination of experts is called committee machine. Committee machine fuses knowledge of experts to achieve an overall task, which may be more efficient than that achieved by any of the experts alone (Tresp, 2001).

The taxonomy of committee machines could be as follows:

- Static structures: Ensemble Averaging and Boosting;
- Dynamic structures: Mixture of Experts and Hierarchical Mixture of Experts.

Next several subsections will present the types of committee machines in detail.

2.2.1 Ensemble averaging

In ensemble averaging technique (Haykin, 1999), (Arbib, 1989), a number of differently trained experts (i.e. neural networks) share a common input and their outputs are combined to produce an overall output value y.

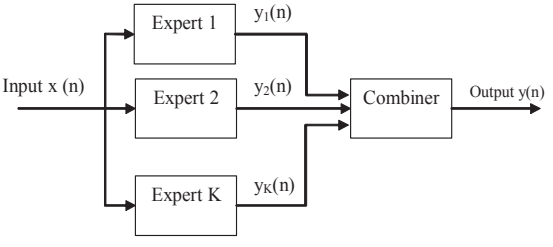

Fig. 1. Ensemble averaging structure

The advantage of such structure over a single expert is that the variance of the average function is smaller than the variance of single expert. Simultaneously both average functions have the same bias. These two facts lead to a training strategy for reducing the overall error produced by a committee machine due to varying initial conditions (Naftaly et al., 1997): the experts are purposely over-trained, what results in reducing the bias at the variance cost. The variance is subsequently reduced by averaging the experts, leaving the bias unchanged.

2.2.2 Boosting

In boosting approach (Schapire, 1999) the experts are trained on data sets with entirely different distributions; it is a general method which can improve the performance of any learning algorithm. Boosting can be implemented in three different ways: Boosting by filtering, Boosting by sub-sampling and Boosting by re-weighing. A well known example is AdaBoost (Schapire, 1999) algorithm, which runs a given weak learner several times on slightly altered training data, and combining the hypotheses to one final hypothesis, in order to achieve higher accuracy than the weak learner's hypothesis would have.

2.2.3 Mixture of experts

Mixture of experts consists of *K* supervised models called expert networks and a gating network, which performs a function of mediator among expert networks. The output is a weighted sum of experts' outputs (Jordan & Jacobs, 2002).
A typical Mixture of Experts structure is presented by figure 2. One can notice the *K* experts and a gating network that filters the solutions of experts. Finally the weighted outputs are combined to produce overall structure output. The gating network consists of *K* neurons, each one is assigned to a specific expert.
The neurons in gating network are nonlinear with activation function that is a differentiable version of "winner-takes-all" operation of picking the maximum value. It is referred as "softmax" transfer function (Bridle, 1990). The mixture of experts is an associative Gaussian mixture model, which is a generalization of traditional Gaussian mixture model (Titterington et al., 1985), (MacLachlan & Basford, 1988).

2.2.4 Hierarchical mixture of experts

Hierarchical mixture of experts (Jordan & Jacobs, 1993) works similarly to ordinary mixture of experts, except that multiple levels of gating networks exist. So the outputs of mixture of experts are gated in order to produce combined output of several mixtures of expert structures. In figure 3 one can see two separate mixture of experts blocks (marked with dashed rectangles). The additional gating network is gating the outputs of these two blocks in order to produce the global structure output.

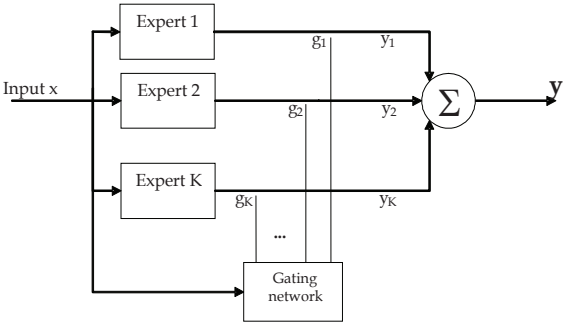

Fig. 2. Mixture of Experts

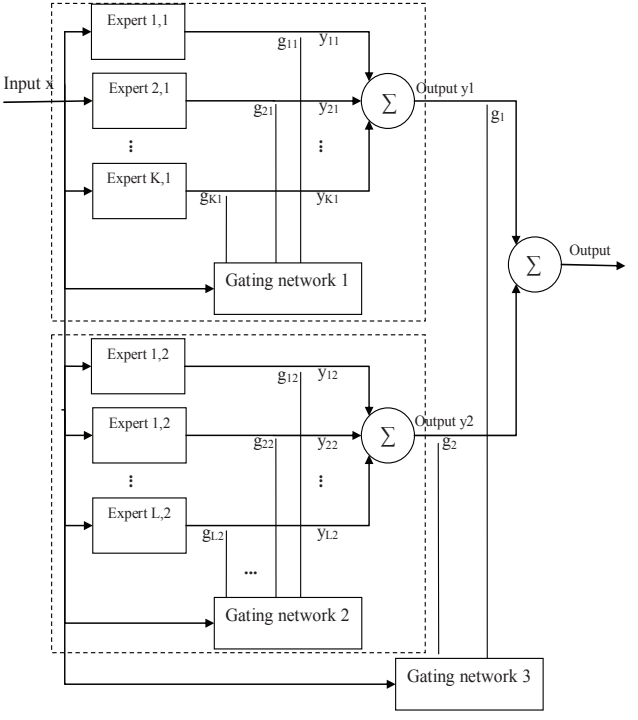

Fig. 3. Example of hierarchical mixture of experts

2.3 Multi agent systems

Multi agent system is a system that compounds of independent modules called "agents". There is no single control structure (designer) which controls all agents. Each of these agents can work on different goals, sometimes in cooperative and sometimes in competitive modes. Both cooperation and competition modes are possible among agents (Decker et al., 1997). There is a great variety of intelligent software agents and structures. The characteristics of Multi Agent Systems (Ferber, 1998) are:

- Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;
- There is no system global control;
- Data are decentralized;
- Computation is asynchronous.

In Multi Agent Systems many intelligent agents interact with each other. The agents can share a common goal (e.g. an ant colony), or they can pursue their own interests (as in the free market economy). Figure 4 gives the classification of intelligent artificial agents considering their origin.

Agents may also be classified according to the tasks they perform:

- **Interface Agents** - Computer programs using artificial intelligence techniques in order to provide assistance to a user dealing with a particular application. The metaphor is that of a personal assistant who is collaborating with the user in the same work environment (Maes, 1994).
- **Information Agents** - An information agent is an agent that has access to at least one, and potentially many information sources, and is able to collect and manipulate information obtained from these sources to answer to users and other information agent's queries (Wooldridge & Jennings, 1995).

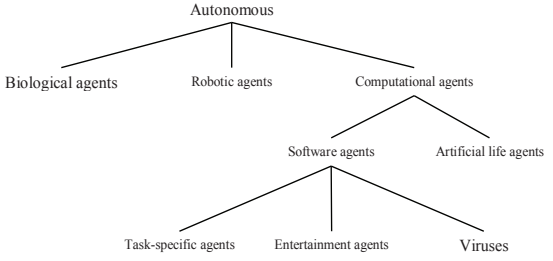

Fig. 4. Classification of intelligent artificial agents considering origin

- **Commerce Agents**- A commerce agent is an agent that provides commercial services (e.g., selling, buying and prices' advice) for a human user or for another agent.
- **Entertainment Agents** - Artistically interesting, highly interactive, simulated worlds to give users the experience of living in (not merely watching) dramatically rich worlds that include moderately competent, emotional agents (Bate et al., 1992).

Agents can communicate, cooperate and negotiate with other agents. The basic idea behind Multi Agent systems is to build many agents with small areas of action and link them together to create a structure which is much more powerful than the single agent itself.

2.4 Discussion

If over past decade wide studies have been devoted to theoretical aspects of modular structures (and algorithms), very few works have concerned their effective implementation and their application to real-world dilemmas. Presenting appealing potential advantages over single structures, this kind of processing systems may avoid difficulties inherent to large and complicated processing systems by splitting the initial complex task into a set of simpler task requiring simpler processing algorithms. The other main advantage is the customized nature of the modular design regarding the task under hand. Among the above-presented structures, the "Divide and Conquer" class of algorithms presents engaging faultlessness. Three variants could be distinguished:

- Each module works with full database aiming a "global" processing. This variant uses a combination of the results issued from individual modules to construct the final system's response.
- Modules work with a part of database (sub-database) aiming a "local" but "not exclusive" processing. In this variant, some of the processing data could be shared by several modules. However, depending on the amount of shared data this variant could be more or less similar to the two others cases.

- Modules work with a part of database (sub-database) aiming a “local” and “exclusive” processing. In this option, sub-databases are exclusive by meaning that no data is shared by modules. The final system’s result could either be a set of responses corresponding to different parts of the initial treated problem or be the output of the most appropriated module among the available ones.

Tree-like Divide To Simplify Approach (described later in this chapter) could be classified as belonging to "Divide and Conquer" class of algorithms as it breaks up an initially complex problem into a set of sub-problems. However, regarding the three aforementioned variants, its actually implemented version solves the sub-problems issued from the decomposition process according to the last variant. In the next section, we present a first modular algorithms which hybridize multi-agents techniques and Q-Neural learning.

3. Multi-agents approach and Q-neural reinforcement learning hybridization: application to QoS complex routing problem

This section present in detail a Q-routing algorithm optimizing the average packet delivery time, based on Neural Network (NN) ensuring the prediction of parameters depending on traffic variations. Compared to the approaches based on Q-tables, the Q-value is approximated by a reinforcement learning based neural network of a fixed size, allowing the learner to incorporate various parameters such as local queue size and time of day, into its distance estimation. Indeed, a Neural Network allows the modeling of complex functions with a good precision along with a discriminating training and network context consideration. Moreover, it can be used to predict non-stationary or irregular traffics. The Q-Neural Routing algorithm is presented in detail in section 3.2. The performance of Q-Routing and Q-Neural Routing algorithms are evaluated experimentally in section 3.3 and compared to the standard shortest path routing algorithms.

3.1 Routing problem in communication networks

Network, such as Internet, has become the most important communication infrastructure of today's human society. It enables the world-wide users (individual, group and organizational) to access and exchange remote information scattered over the world. Currently, due to the growing needs in telecommunications (VoD, Video-Conference, VoIP, etc.) and the diversity of transported flows, Internet network does not meet the requirements of the future integrated-service networks that carry multimedia data traffic with a high Quality of Service (QoS). The main drivers of this evolution are the continuous growth of the bandwidth requests, the promise of cost improvements and finally the possibility of increasing profits by offering new services. First, it does not support resource reservation which is primordial to guarantee an end-to-end Qos (bounded delay, bounded delay jitter, and/or bounded loss ratio). Second, data packets may be subjected to unpredictable delays and thus may arrive at their destination after the expiration time, which is undesirable for continuous real-time media. In this Context, for optimizing the financial investment on their networks, operators must use the same support for transporting all the flows. Therefore, it is necessary to develop a high quality control mechanism to check the network traffic load and ensure QoS requirements.

A lot of different definitions and parameters for this concept of quality of service can be found. For ITU-T E.800 recommendation, QoS is described as “the collective effect of service performance which determines the degree of satisfaction of a user of the service”. This

definition is completed by the I.350 ITU-T recommendation which defines more precisely the differences between QoS and Network Performance. Relating QoS concepts in the Internet are focused on a packet-based end-to-end, edge-to-edge or end-to-edge communication. QoS parameters which refer to this packet transport at different layers are: availability, bandwidth, delay, jitter and loss ratio. It's clear that the integration of these QoS parameters increases the complexity of the used algorithms. Anyway, there will be QoS relevant technological challenges in the emerging hybrid networks which mixes several networks topologies and technologies (wireless, broadcast, mobile, fixed, etc.).
In the literature, we can find the usage of QoS in three ways:

- *Deterministic QoS* consists in sufficiently resources reserved for a particular flow in order to respect the strict temporal constraints for all the packages of flow. No loss of package or going beyond of expiries is considered in this type of guarantee. This model makes it possible to provide an absolute terminal on the time according to the reserved resources.
- *Probabilistic QoS* consists in providing a long-term guarantee of the level of service required by a flow. For time-reality applications tolerating the loss of a few packages or the going beyond of some expiries, the temporal requirements as well as the rates of loss are evaluated on average. The probabilistic guarantee makes it possible to provide a temporal terminal with a certain probability which is given according to the conditions of load of the network.
- *Stochastic QoS* which is fixed before by a stochastic distribution.

Various techniques have been proposed to take into account QoS requirements (Strassner, 2003). By using in-band or out-band specific control protocols, these techniques may be classified as follows: the congestion control (Slow Start (Welzl, 2003), Weighted Random Early Detection (Jacobson, 1988)), the traffic shaping (Leaky Bucket (Feng et al., 1997), Token Bucket (Turner, 1986)), integrated services architecture, (RSVP (Shenker et al., 1997), (Zhang et al., 1993)), the differentiated services (DiffServ (Zhang et al., 1993), (Bernet, 1998)) and QoS based routing. In this section, we focus on QoS routing policies.
A routing algorithm is based on the hop-by-hop shortest-path paradigm. The source of a packet specifies the address of the destination, and each router along the route forwards the packet to a neighbour located "closest" to the destination. The best optimal path is selected according to given criteria. When the network is heavily loaded, some of the routers introduce an excessive delay while others are ignored (not expoited). In some cases, this non-optimized usage of the network resources may introduce not only excessive delays but also high packet loss rate. Among routing algorithms extensively employed in routers, one can note: distance vector algorithm such as RIP (Malkin, 1993) and the link state algorithm such as OSPF (Moy, 1998). These kinds of algorithms take into account variations of load leading to limited performances.
A lot of study has been conducted in a search for an alternative routing paradigm that would address the integration of dynamic criteria. The most popular formulation of the optimal distributed routing problem in a data network is based on a multi-commodity flow optimization whereby a separable objective function is minimized with respect to the types of flow subject to multi-commodity flow constraints (Gallager, 1977), (Ozdalgar et al., 2003). However, due their complexity, increased processing burden, a few proposed routing schemes could be accepted for the internet. We listed here some QoS based routing algorithms proposed in the literature: QOSPF (Quality Of Service Path First) (Crawley et al.,

1998), MPLS (Multiprotocol label switching) (Rosen et al., 1999), (Stallings, 2001), (Partridge, 1992), Traffic Engineering (Strasnner, 2003), (Welzl, 2003), Wang-Crowcroft algorithm (Wang & Crowcroft, 1996), Ants routing approach (Subramanian et al., 1997), Cognitive Packet Networks based on random neural networks (Gelenbe et al., 2002).
For a network node to be able to make an optimal routing decision, according to relevant performance criteria, it requires not only up-to-date and complete knowledge of the state of the entire network but also an accurate prediction of the network dynamics during propagation of the message through the network. This, however, is impossible unless the routing algorithm is capable of adapting to network state changes in almost real time. So, it is necessary to develop a new intelligent and adaptive optimizing routing algorithm. This problem is naturally formulated as a dynamic programming problem, which, however, is too complex to be solved exactly.
In our approach, we use the methodology of reinforcement learning (RL) introduced by Sutton (Sutton & Barto, 1997) to approximate the value function of dynamic programming. One of pioneering works related to this kind of approaches concerns Q-Routing algorithm (Boyan & Littman, 1994) based on Q-learning technique (Watkins & Dayan, 1989). In this approach, each node makes its routing decision based on the local routing information, represented as a table of Q values which estimate the quality of the alternative routes. These values are updated each time the node sends a packet to one of its neighbors. However, when a Q value is not updated for a long time, it does not necessarily reflect the current state of the network and hence a routing decision based on such an unreliable Q value will not be accurate. The update rule in Q-Routing does not take into account the reliability of the estimated or updated Q value because it's depending on the traffic pattern, and load levels, only a few Q values are current while most of the Q values in the network are unreliable. For this purpose, other algorithms have been proposed like Confidence based Q-Routing (CQ-Routing) (Kumar & Miikkualainen, 1998) or Dual Reinforcement Q-Routing (DRQ-Routing) (Kumar & Miikkualainen, 1997), (Goetz et al., 1996). All these routing algorithms use a table to estimate Q values. However, the size of the table depends on the number of destination nodes existing in the network. Thus, this approach is not well suited when we are concerned with a state-space of high dimensionality.

3.2 Q-neural routing approach

In this section, we present an adaptive routing algorithm based on the Q-learning approach, the Q-function is approximated by a reinforcement learning based neural network. First, we formulate the reinforcement learning process.

3.2.1 Reinforcement learning

Algorithms for reinforcement learning face the same issues as traditional distributed algorithms, with some additional peculiarities. First, the environment is modelled as stochastic (especially links, link costs, traffic, and congestion), so routing algorithms can take into account the dynamics of the network. However no model of dynamics is assumed to be given. This means that RL algorithms have to sample, estimate, and perhaps build models of pertinent aspect of the environment. Second, RL algorithms, unlike other machine learning algorithms, do not have an explicit learning phase followed by evaluation. Since there is no training signal for a direct evaluation of the policy's performance before the packet has reached its final destination, it is difficult to apply supervised learning techniques to this

problem (Haykin, 1998). In addition, it is difficult to determine to what extent a routing decision that has been made on a single node may influence the network's overall performance. This fact fits into the temporal credit assignment problem (Watkins, 1989).
The RL algorithm, called reactive approach, consists of endowing an autonomous agent with a correctness behavior guaranteeing the fulfillment of the desired task in the dynamics environment. The behavior must be specified in terms of Perception - Decision – Action loop (Fig. 5). Each variation of the environment induces stimuli received by the agent, leading to the determination of the appropriate action. The reaction is then considered as a punishment or a performance function, also called, reinforcement signal.

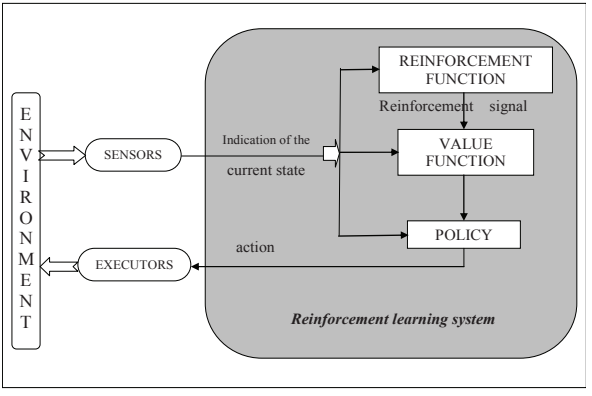

Fig. 5. Reinforcement learning model

Thus, the agent must integrate this function to modify its future actions in order to reach an optimal performance. In other words, a RL Algorithm is a finite-state machine that interacts with a stochastic environment, trying to learn the optimal action the environment offers through a learning process. At any iteration the automaton's agent chooses an action, according to a probability vector, using an output function. This function stimulates the environment, which responds with an answer (reward or penalty). The automaton's agent takes into account this answer and jumps, if necessary, to a new state using a transition function.

Network Elements	**RL System**	
Network	Environment	-
Each network node	Agent	-
Delay in links and nodes	Reinforcement	T
Estimate of total delay	Function	$Q(s, y, d)$
Action of sending a packet	Value Function	$a(s_t)$
Node through which the packet passes in time t	Action	s_t
	State in time t	
Local routing decision	Policy	π

Table. 1. Correspondences between a RL system and network elements

It is necessary for the agent to gather useful experience about the possible system states, actions, transitions and rewards actively to act optimally. Another difference from

supervised learning is that on-line performance is important: the evaluation of the system is often concurrent with learning.
A Reinforcement Learning system thus involves the following elements: an Agent, an Environment, a Reinforcement Function, an Action, a State, a Value Function, which is obtained from the reinforcement function, and a Policy. In order to obtain a network routing useful model, it is possible to associate the network's elements to the basic elements of a RL system, as shown in Table 1.

3.2.2 Q-learning algorithm for routing

In our routing algorithm (Mellouk, 2006), the objective is to minimize the average packet delivery time. Consequently, the reinforcement signal which is chosen corresponds to the estimated time to transfer a packet to its destination. Typically, the packet delivery time includes three variables: The packet transmission time, the packet treatment time in the router and the latency in the waiting queue. In our case, the packet transmission time is not taken into account. In fact, this parameter can be neglected in comparison to the other ones and has no effect on the routing process.
The reinforcement signal T employed in the Q-learning algorithm can be defined as the minimum of the sum of the estimated $Q\ (y, x, d)$ sent by the router x neighbor of router y and the latency in waiting queue q_y corresponding to router y.

$$T = \min_{x \in \text{neighbor of } y} \left\{ q_y + Q(y,x,d) \right\} \quad (1)$$

$Q(s, y, d)$ denote the estimated time by the router s so that the packet p reaches its destination d through the router y. This parameter does not include the latency in the waiting queue of the router s. The packet is sent to the router y which determines the optimal path to send this packet (Watkins, 1989).

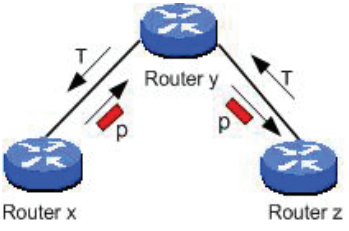

Fig. 6. Updating the reinforcement signal

Once the choice of the next router made, the router y puts the packet in the waiting queue, and sends back the value T as a reinforcement signal to the router s. It can therefore update its reinforcement function as:

$$\Delta Q(s,y,d) = \eta(\alpha + T - Q(s,y,d)) \quad (2)$$

So, the new estimation $Q'(s,y,d)$ can be written as follows (fig.6):

$$Q'(s,y,d) = Q(s,y,d)\left(1-\eta\right) + \eta(T+\alpha) \quad (3)$$

α and η are respectively, the packet transmission time between s and y, and the learning rate.

3.2.2 Q-learning neural net architecture

The neural network proposed in our study is a Recurrent Multi-Layers Perceptron (MLP) with an input, one hidden and an output layer.

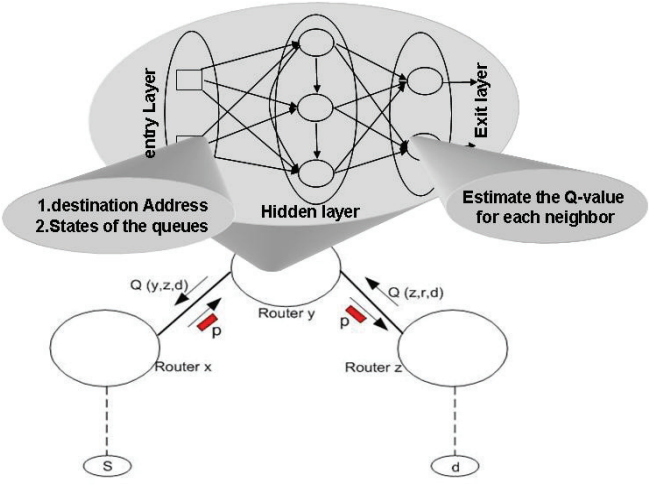

Fig. 7. Artificial Neural Network Architecture

The input cells correspond to the destination addresses d and the waiting queue states. The outputs are the estimated packet transfer times passing through the neighbors of the considered router. The algorithm derived from this architecture is called Q-Neural Routing and can be described according to the following algorithm:

Etiq1 :
{While (not packet receive)
    Begin
    End
}
If (packet = "packet of reinforcement")
    Begin
        1. Neural Network updating using a retro-propagation algorithm based on gradient method,
        2. Destroy the reinforcement packet.
    End
Else
    Begin
        1. Calculate Neural Network outputs,
        2. Select the smallest output value and get an IP address of the associated router,
        3. Send the packet to this router,
        4. Get an IP address of the precedent router,
        5. Create and send the packet as a reinforcement signal.
    End
End
Goto ***Etiq1***

3.3 Implementation and simulation results

To show the efficiency and evaluate the performances of our approach, an implementation has been performed on OPNET software of MIL3 Company. The proposed approach has been compared to that based on standard Q-routing (Boyan & Littman, 1994) and shortest path routing policy. OPNET constitutes for telecommunication networks an appropriate modeling, scheduling and simulation tool. It allows the visualization of a physical topology of a local, metropolitan, distant or on board network. The protocol specification language is based on a formal description of a finite state automaton.

The proposed approaches have been compared to that based on standard Q-routing and shortest paths routing policies (such as Routing Internet Protocol RIP). The topology of the network used for simulations purpuse, which used in many papers, includes 33 interconnected nodes, as shown in figure 8. Two kinds of traffic have been studied: low load and high load of the network. In the first case, a low rate flow is sent to node destination-1, from nodes source-1 and source-4. From the previous case, we have created conditions of congestion of the network. Thus, a high rate flow is generated by nodes source-2 and source-3. Two possible ways R-1 (router-29 and router-30) and R-2 (router-21 and router-22) to route the packets between the left part and the right part of the network.

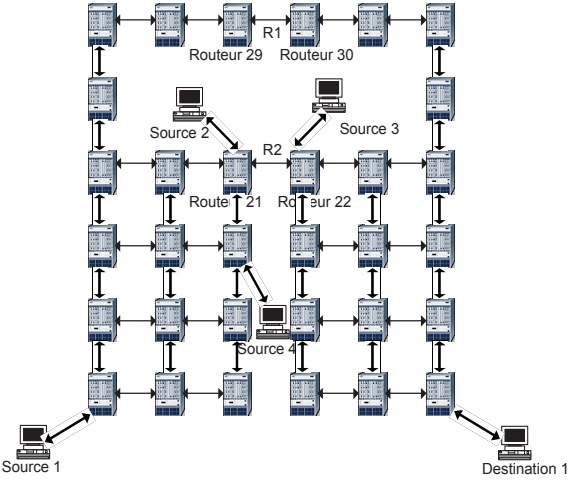

Fig. 8. Network topology for simulation

Performances of algorithms are evaluated in terms of average packet delivery time. Figure 9 and figure 10 illustrates the obtained results when source-2 and source-3 send information packets during 10 minutes. From figure 10, one can see clearly, that after an initialization period, the Q-routing and Q-Neural routing algorithms, exhibit better performances than RIP. Thus, packet average delivery time obtained by Q-routing algorithm and Q-Neural routing algorithm is reduced of respectively 23.6% and 27.3% compared to RIP routing policy (table 2). These results confirm that classical shortest path routing algorithm like RIP lead to weak performances due to packets delayed in the waiting queues of the routers. Moreover, this policy does not take into account the load of the network. On the other hand, when a way of destination is saturated, Q-routing and Q-Neural routing algorithms allow

the selection of a new one to avoid this congestion. In the case of a low load (figure 10), one can note that after a period of initialization, performances of these algorithms are approximately the same as those obtained with RIP routing policy.

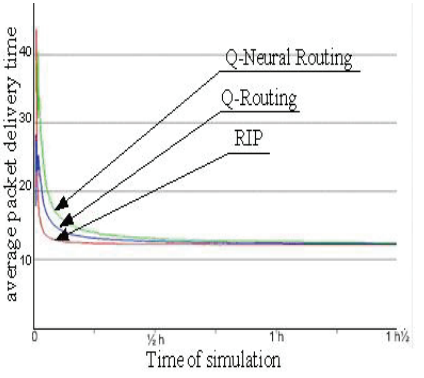

Fig. 9. Network with a low load

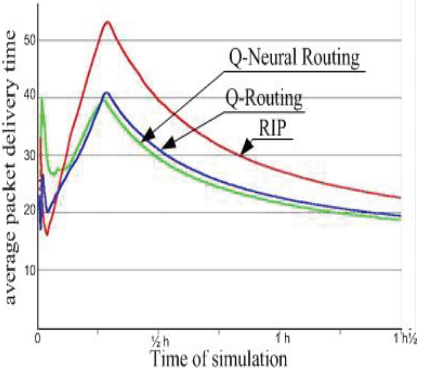

Fig. 10. Network with a high load

Computed Algorithms	*MAPTT*
Q-routing	42
Q-neural routing	40
RIP	55

Table 2. Maximum average packet delivery time

Figure 11 illustrates the average packet delivery time obtained when a congestion of the network is generated during 60 minutes. Thus, in the case where the number of packets is more important, the Q-Neural routing algorithm gives better results compared to Q-routing algorithm. For example, after 2 hours of simulation, Q-Neural routing exhibits a performance of 20% higher than that of Q-routing. Indeed, the use of waiting queue state of the neighboring routers in the routing decision, allows anticipation of routers congestion.

In general, the topology of the neural network must be fixed before the training process. The only variables being able to be modified are the values of the weights of connections. The specification of this architecture, the number of cells of each layer and of connections, remains a crucial problem. If this number is insufficient, the model will not be able to take into account all data. A contrario, if it is too significant, the training will be perfect but the network generalization ability will be poor (overfitting problem). However, we are concerned here by online training, for which the number of examples is not defined a priori. For that, we propose an empirical study based on pruning technique to find a compromise between a satisfactory estimate of the function Q and an acceptable computing time.

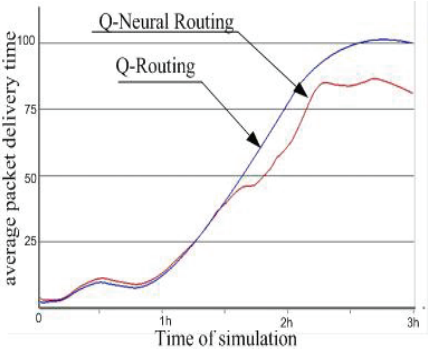

Fig. 11. Very High load Network

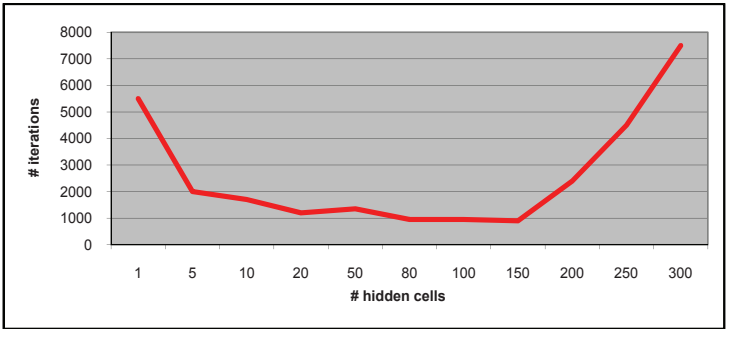

Fig. 12. Empirical pruning study for choosing the number of hidden cells over time

The Neural network structure has been fixed using an empirical pruning strategy (figure 12). A self-organizing approach is useful for automatic adjustment of the neural network parameters as the number of neuron per layer and the hidden layers numbers for example. Next section introduces such a concept and present complexity estimation based self organizing structure.

4. Self-organizing modular information processing through the Tree-like Divide To Simplify approach

This section presents in detail the "Tree-like-Divide To Simplify" (T-DTS) approach, define its structure, and describe the types of modules that are used in the structure. T-DTS is based on modular tree-like decomposition structure, which is used amongst others for task decomposition. This section will present also in detail procedures and algorithms that are used for the creation, execution and modification of the modules. It will discuss also advantages and disadvantages of T-DTS approach and compare it with other approaches.

T-DTS is a self-organizing modular structure including two types of modules: Decomposition Unit (DU) and Processing Unit (PU). The purpose is based on the use of a set of specialized mapping neural networks (PU), supervised by a set of DU. DU could be a prototype based neural network, Markovian decision process, etc. The T-DTS paradigm

allows us to build a modular tree structure. In such structure, DU could be seen as "nodes" and PU as leaves. At the nodes level, the input space is decomposed into a set of subspaces of smaller sizes. At the leaves level, the aim is to learn the relations between inputs and outputs of sub-spaces, obtained from splitting. As the modules are based on Artificial Neural Networks, they inherit the ANN's approximation universality as well as their learning and generalization abilities.

4.1 Hybrid Multiple Neural Networks framework - T-DTS

As it has been mentioned above, in essence, T-DTS is a self-organizing modular structure (Madani et Al., 2003). T-DTS paradigm builds a tree-like structure of models (DU and PU). Decomposition Units are prototypes based ANNs and Processing Units are specialized mapping ANNs. However, in a general frame, PU could be any kind of processing model (conventional algorithm or model, ANN based model, etc…). At the nodes level(s) - the input space is decomposed into a set of optimal sub-spaces of the smaller size. At the leaves level(s) - the aim is to learn the relation between inputs and outputs of sub-spaces obtained from splitting. T-DTS acts in two main operational phases:

Learning: recursive decomposition under DU supervision of the database into sub-sets: tree structure building phase;

Operational: Activation of the tree structure to compute system output (provided by PU at tree leaf's level).

General block diagram of T-DTS is described on Figure 13. The proposed schema is open software architecture. It can be adapted to specific problem using the appropriate modeling paradigm at PU level: we use mainly Artificial Neural Network computing model in this work. In our case the tree structure construction is based on a complexity estimation module. This module introduces a feedback in the learning process and control the tree building process. The reliability of tree model to sculpt the problem behavior is associated to the complexity estimation module. The whole decomposing process is built on the paradigm "*splitting database into sub-databases - decreasing task complexity*". It means that the decomposition process is activated until a low satisfactory complexity ratio is reached. T- DTS

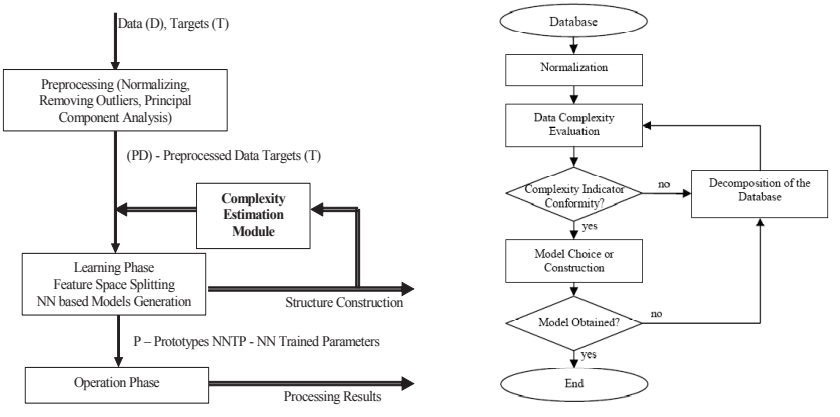

Fig. 13. Bloc scheme of T-DTS: *Left* – Modular concept, *Right* – Algorithmic concept

software architecture is depicted on Figure 14. T-DTS software incorporates three databases: decomposition methods, ANN models and complexity estimation modules databases.

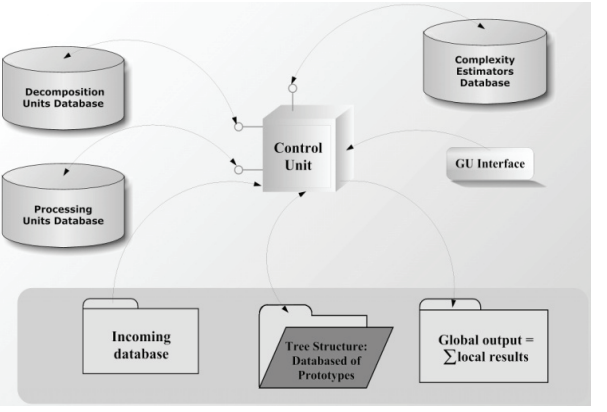

Fig. 14. T-DTS software architecture

T-DTS software engine is the *Control Unit.* This core-module controls and activates several software packages: normalization of incoming database (if it's required), splitting and building a tree of prototypes using selected decomposition method, sculpting the set of local results and generating global result (learning and generalization rates). T-DTS software can be seen as a *Lego system* of decomposition methods, processing methods powered by a control engine an accessible by operator thought Graphic User Interface.

The three databases can be independently developed out of the main frame and more important, they can be easily incorporated into T-DTS framework.

For example, SOM-LSVMDT (Mehmet et al., 2003) algorithm; which is based on the same idea of decomposition, can be implement by T-DTS by mean of LSVMDT (Chi & Ersoy, 2002) (Linear Support Vector Machine Decision Tree) processing method incorporation into PU database.

- The current T-DTS software (version 2.02) includes the following units and methods:
  - ✓ Decomposition Units:
  - ✓ CN (Competitive Network)
  - ✓ SOM (Self Organized Map)
  - ✓ LVQ (Learning Vector Quantization)
- Processing Units:
  - ✓ LVQ (Learning Vector Quantization)
  - ✓ Perceptrons
  - ✓ MLP (Multilayer Perceptron)
  - ✓ GRNN (General Regression Neural Network)
  - ✓ RBF (Radial basis function network)
  - ✓ PNN (Probabilistic Neural Network)
  - ✓ LN
- Complexity estimators (Bouyoucef, 2007), presented in sub-section 4.2.5, are based on the following criteria:
  - ✓ MaxStd (Sum of the maximal standard deviations)

- ✓ Fisher measure.
- ✓ Purity measure
- ✓ Normalized mean distance
- ✓ Divergence measure
- ✓ Jeffries-Matusita distance
- ✓ Bhattacharyya bound
- ✓ Mahalanobis distance
- ✓ Scattered-matrix method based on inter-intra matrix-criteria (Fukunaga, 1972).
- ✓ ZISC© IBM ® based complexity indicator (Budnyk & al. 2007).

4.2 T-DTS learning and decomposition mechanism

The decomposition mechanism in T-DTS approach builds a tree structure. The creation of decomposition tree is data-driven. It means that the decision to-split-or-not and how-to-split is made depending on the properties of the current sub-database. For each database the decision to-split-or-not should be made. After a positive decision a Decomposition Unit (DU) is created which divides the data and distributes the resulting sub-databases creating children in the tree. If the decision is negative the decomposition of this sub-database (and tree branch) is over and a Processing Unit should be built for the sub-database. The type of the new tree module depends on the result of decomposition decision made for the current sub-database (and in some cases also on other parameters, as described later). The tree is built beginning from the root which achieves the complete learning database. The process results in a tree which has DUs at nodes and Processing Unit models in tree leaves.
Figure 15 shows decomposition tree structure (in case of binary tree) and its recurrent construction in time, while question marks mean decomposition decisions.
For any database B (including the initial) a splitting decision (if to split and how to split) is taken. When the decision is positive then a Decomposition Unit is created, and the database is decomposed (clustered) by the new Decomposition Unit. When the decomposition decision is negative, a Processing Unit is created in order to process the database (for example to create a model).
The database B incoming to some Decomposition Unit will be split into several sub-databases $b_1,b_2...b_k$, depending on the properties of the database B and parameters τ obtained from controlling structure. The function $S(\psi_i,\tau)$ assigns any vector ψ_i from database B to an appropriate sub-database *j*. The procedure is repeated in recursive way i.e. for each resulting sub-database a decomposition decision is taken and the process repeats. One chain of the process is depicted in figure 16.

$$S(\Psi_i,\tau,\xi) = (s_1 \ldots s_k \ldots s_M)^T \quad \textbf{with} \quad \begin{array}{ll} s_k = 1 & \text{if } \tau = \tau_k \text{ and } \xi = \xi_k \\ s_k = 0 & \text{else} \end{array} \tag{4}$$

The scheduling vector $S(\psi_i,\tau_k)$ will activate (select) the *K*-th Processing Unit, and so the processing of an unlearned input data conform to parameter τ_k and condition ξ_k will be given by the output of the selected Processing Unit:

$$Y(\Psi_i) = Y_k(i) = F_k(\Psi_i) \tag{5}$$

Complexity indicators are used in our approach in order to reach one of the following goals:

- Global decomposition control - estimator which evaluates the difficulty of classification of the whole dataset and chooses decomposition strategy and parameters before any decomposition has started,
- Local decomposition control - estimator which evaluates the difficulty of classification of the current sub-database during decomposition of dataset, in particular:
  - ✓ Estimator which evaluates the difficulty of classification of the current sub-database, to produce decomposition decision (if to divide the current sub-database or not);
  - ✓ Estimator which can be used to determine the type of used classifier or its properties and parameters.
- Mixed approach - use of many techniques mentioned above at once, for example: usage of Global decomposition control to determine the parameters of local decomposition control.

One should mention also that estimation of sub-database complexity occurs for each sub-database dividing decision thus computational complexity of the algorithm should rather be small. Thus it doesn't require advanced complexity estimation methods. Considering these features, the second option - estimation during decomposition - has been chosen in our experiments in order to achieve self adaptation feature of T-DTS structure.

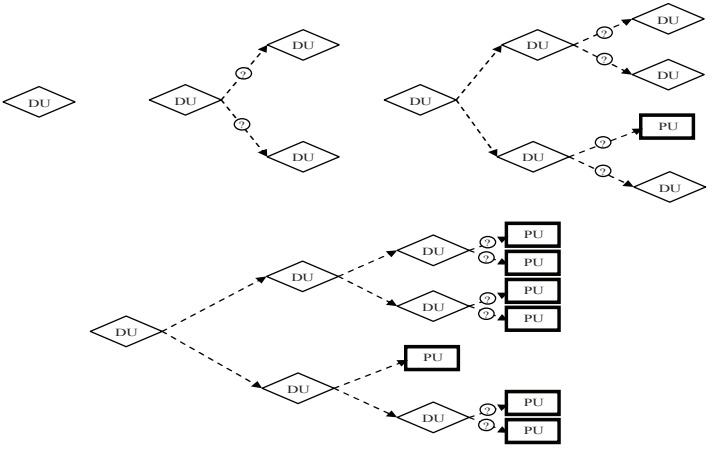

Fig. 15. T-DTS decomposition tree creation in time

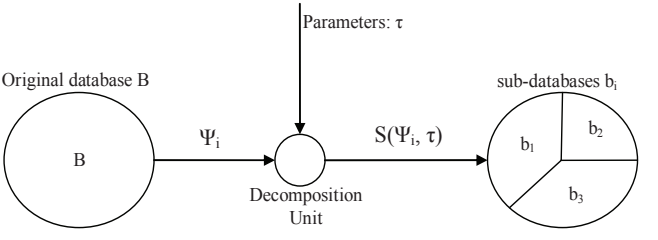

Fig. 16. Decomposition Unit activities

4.2.1 Decomposition Unit (DU)

The purpose of Decomposition Unit is to divide the database into several sub-databases. This task is referred in the literature as clustering (Hartgan, 1975). To accomplish this task a plenty of methods are known. We are using Vector Quantization unsupervised methods, in particular: competitive Neural Networks and Kohonen Self-Organizing Maps (Kohonen, 1984). These methods are based on prototype, that represent the centre of cluster (cluster = group of vectors). In our approach cluster is referred to as sub-database.

4.2.2 Decomposition of learning database

The learning database is split into M learning sub-databases by DUs during building of the decomposition tree. The learning database decomposition is equivalent to "following the decomposition tree" decomposition strategy. The resulting learning sub-databases could be used for Processing Unit learning. Each sub-database has then Processing Unit attached. The Processing Unit models are trained using the corresponding learning sub-database.

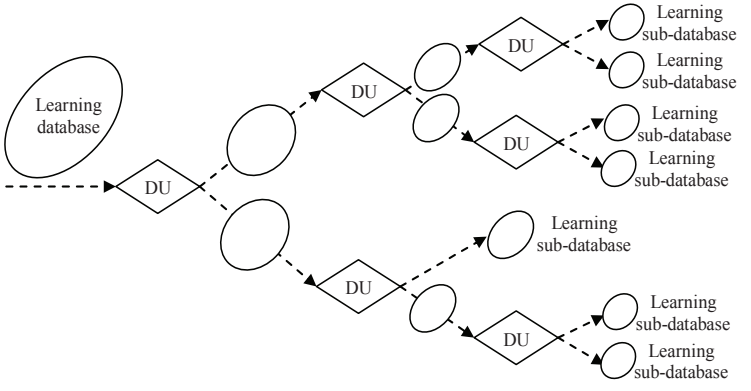

Fig. 17. Decomposition of learning database "following the decomposition tree" strategy

4.2.3 Training of Processing Units (models)

For each sub-database T-DTS constructs a neural based model describing the relations between inputs and outputs. Training of Processing Unit models is performed using standard supervised training techniques, possibly most appropriate for the learning task required. In this work only Artificial Neural Networks are used, however there should be no difficulty to use other modelling techniques.

Processing Unit is provided with a sub-database and target data. It is expected to model the input/output mapping underlying the subspace as reflected by the sub-database provided. The trained model is used later to process data patterns assigned to the Processing Unit by assignment rules.

4.2.4 Processing Units

Processing Unit models used in our approach can be of any origin. In fact they could be also not based on Artificial Neural Networks at all. The structure used depend on the type of learning task, we use:

- for classification - MLP, LVQ, Probabilistic Networks (Haykin, 1999), RBF, Linear Networks;
- for regression - MLP, RBF;
- for model identification - MLP.

Processing Unit models are created and trained in the learning phase of T-DTS algorithm, using learning sub-databases assigned by decomposition structure. In the generalization phase, they are provided with generalization vectors assigned to them by pattern assignment rules. The vectors form generalization sub-databases are processed by Processing Unit models. Each Processing Unit produce some set of approximated output vectors, and the ensemble of them will compose whole generalization database.

4.2.5 Complexity estimation techniques

The goal of complexity estimation techniques is to estimate the processing task's difficulty. The information provided by these techniques is mainly used in a splitting process according to a divide and conquer approach. It act's at three levels:

- The task decomposition process up to some degree dependant on task or data complexity.
- The choice of appropriate processing structure (i.e. appropriated model) for each subset of decomposed data.
- The choice of processing architecture (i.e. models parameters).

The techniques usually used for complexity estimation are sorted out in three main categories: those based on Bayes error estimation, those based on space partitioning methods and others based on intuitive paradigms. Bayes error estimation may involve two classes of approaches, known as: *indirect* and *non-parametric* Bayes error estimation methods, respectively. This sub-section of the chapter will present a detailed summery of these main complexity estimation methods which are used in the T-DTS self-organizing system core, focusing mainly on measurements supporting task decomposition aspect.

4.2.5.1 Indirect Bayes error estimation

To avoid the difficulties related to direct estimation of the Bayes error, an alternative approach is to estimate a measure directly related to the Bayes error, but easier to compute. Usually one assumes that the data distribution is normal (Gaussian). Statistical methods grounded in the estimation of probability distributions are most frequently used. The drawback of these is that they assume data normality. A number of limitations have been documented in literature (Vapnik, 1998):

- model construction could be time consuming;
- model checking could be difficult;
- as data dimensionality increases, a much larger number of samples is needed to estimate accurately class conditional probabilities;
- if sample does not sufficiently represent the problem, the probability distribution function cannot be reliably approximated;
- with a large number of classes, estimating a priori probabilities is quite difficult. This can be only partially overcome by assuming equal class probabilities (Fukunaga, 1990), (Ho & Basu, 2002).
- we normally do not know the density form (distribution function);
- most distributions in practice are multimodal, while models are unimodal;
- approximating a multimodal distributions as a product of univariate distributions do not work well in practice.

4.2.5.1.1 Normalized mean distance

Normalized mean distance is a very simple complexity measure for Gaussian unimodal distribution. It raises when the distributions are distant and not overlapping.

$$d_{norm} = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{6}$$

The main drawback of that estimator is that it is inadequate (as a measure of separability) when both classes have the same mean values.

4.2.5.1.2 Chernoff bound

The Bayes error for the two class case can be expressed as:

$$\varepsilon = \int \min_i \left[P(c_k)\, p\left(x \mid c_k\right) \right] dx \tag{7}$$

Through modifications, we can obtain a Chernoff bound ε_u, which is an upper bound on ε for the two class case:

$$\varepsilon_u = P(c_1)^s P(c_2)^{1-s} \int p(x^s \mid c_1)\, p(x^{1-s} \mid c_2) dx \quad \text{for } 0 \le s \le 1 \tag{8}$$

The tightness of bound varies with s.

4.2.5.1.3 Bhattacharyya bound

The Bhattacharyya bound is a special case of Chernoff bound for s = 1/2. Empirical evidence indicates that optimal value for Chernoff bound is close to 1/2 when the majority of separation comes from the difference in class means. Under a Gaussian assumption, the expression of the Bhattacharyya bound is:

$$\varepsilon_b = \sqrt{P(c_1)P(c_2)}\, e^{-\mu(1/2)} \tag{9}$$

where:

$$\mu(1/2) = \frac{1}{8}(\mu_2 - \mu_1)^T \left[\frac{\Sigma_1 + \Sigma_2}{2} \right]^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \ln \frac{\left| \frac{\Sigma_1 + \Sigma_2}{2} \right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \tag{10}$$

and μ_i and Σ_i are respectively the means and classes covariance's (i in {1,2}).

4.2.5.1.4 Mahalanobis distance

Mahalanobis distance (Takeshita et al., 1987) is defined as follows:

$$M_D = \left(\mu_2 - \mu_1\right)^T \Sigma^{-1} \left(\mu_2 - \mu_1\right) \tag{11}$$

M_D is the Mahalanobis distance between two classes. The classes' means are μ_1 and μ_2 and Σ is the covariance matrix. Mahalanobis distance is used in statistics to measure the similarity of two data distributions. It is sensitive to distribution of points in both samples. The Mahalanobis distance is measured in units of standard deviation, so it is possible to assign statistical probabilities (that the data comes from the same class) to the specific measure

values. Mahalanobis distance greater than 3 is considered as a signal that data are not homogenous (does not come from the same distribution).

4.2.5.1.5 Jeffries-Matusita distance

Jeffries-Matusita (Matusita ,1967) distance between class's c_1 and c_2 is defined as:

$$JM_D = \int_x \left\{ \sqrt{p(X \mid c_2)} - \sqrt{p(X \mid c_1)} \right\}^2 dx \tag{12}$$

If class's distributions are normal Jeffries-Matusita distance reduces to:

$$JM_D = 2\left(1 - e^{-\alpha}\right), \text{ where} \tag{13}$$

$$\alpha = \frac{1}{8}(\mu_2 - \mu_1)^T \left(\frac{\Sigma_2 + \Sigma_1}{2} \right)^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \log_e \left(\frac{\det \Sigma}{\det \Sigma_1 - \det \Sigma_2} \right) \tag{14}$$

Matusita distance is bounded within range [0, 2] where high values signify high separation between c_1 and c_2 classes.

4.2.5.2 Non-Parametric Bayes error estimation and bounds

Non-parametric Bayes error estimation methods make no assumptions about the specific distributions involved. They use some intuitive methods and then prove the relation to Bayes error. Non-parametric techniques do not suffer from problems with parametric techniques.

4.2.5.2.1 Error of the classifier itself

This is the most intuitive measure. However it varies much depending on the type of classifier used and, as such, it is not very reliable unless one uses many classification methods and averages the results. The last solution is certainly not computationally efficient.

4.2.5.2.2 k-Nearest Neighbours, (k-NN)

K- Nearest Neighbours (Cove & Hart, 1967) technique relays on the concept of setting a local region *Γ(x)* around each sample *x* and examining the ratio of the number of samples enclosed *k* to the total number of samples *N*, normalized with respect to region volume *v*:

$$\hat{p}(x) = \frac{k(x)}{vN} \tag{15}$$

K-NN technique fixes the number of samples enclosed by the local region (*k* becomes constant). The density estimation Equation for k-NN becomes:

$$\hat{p}(x) = \frac{k\text{-}1}{v(x)N} \tag{16}$$

where *p(x)* represent probability of specific class appearance and *v(x)* represent local region volume. K-NN is used to estimate Bayes error by either providing an asymptotic bound or through direct estimation. K-NN estimation is computationally complex.

4.2.5.2.3 Paren Estimation

Parzen techniques relay on the same concept as k-NN: setting a local region $\Gamma(x)$ around each sample x and examining the ratio of the samples enclosed k, to the total number of samples N, normalized with respect to region volume v:

$$\hat{p}(x) = \frac{k}{vN} \tag{17}$$

The difference according to k-NN is that Parzen fixes the volume of local region v. Then the density estimation equation becomes:

$$\hat{p}(x) = \frac{k(x)}{vN} \tag{18}$$

where $p(x)$ represents density and $k(x)$ represents number of samples enclosed in volume. Estimating the Bayes error using the Parzen estimate is done by forming the log likelihood ratio functions based upon the Parzen density estimates and then using resubstitution and leave-one-out methodologies to find an optimistic and pessimistic value for error estimate. Parzen estimates are however not known to bound the Bayes error. Parzen estimation is computationally complex.

4.2.5.2.4 Boundary methods

The boundary methods are described in the work of Pierson (Pierson, 1998). Data from each class is enclosed within a boundary of specified shape according to some criteria. The boundaries can be generated using general shapes like: ellipses, convex hulls, splines and others. The boundary method often uses ellipsoidal boundaries for Gaussian data, since it is a natural representation of those. The boundaries may be made compact by excluding outlying observations. Since most decision boundaries pass through overlap regions, a count of these samples may give a measure related to misclassification rate. Collapsing boundaries iteratively in a structured manner and counting the measure again lead to a series of decreasing values related to misclassification error. The rate of overlap region decay provides information about the separability of classes. Pierson discuses in his work a way in which the process from two classes in two dimensions can be expanded to higher dimension with more classes. Pierson has demonstrated that the measure of separability called the Overlap Sum is directly related to Bayes error with a much more simple computational complexity. It does not require any exact knowledge of the *a posteriori* distributions. Overlap Sum is the arithmetical mean of overlapped points with respect to progressive collapsing iterations:

$$O_S(mt_0) = \frac{1}{N}\sum_{k=1}^{m}(kt_0)\Delta s(kt_0) \tag{19}$$

where t_o is the step size, m is the maximum number of iteration (collapsing boundaries), N is the number of data points in whole dataset and $\Delta s(kt_0)$ is the number of points in the differential overlap.

4.2.5.3 Measures related to space partitioning

Measures related to space partitioning are connected to space partitioning algorithms. Space partitioning algorithms divide the feature space into sub-spaces. That allows obtaining some

advantages, like information about the distribution of class instances in the sub-spaces. Then the local information is globalized in some manner to obtain information about the whole database, not only the parts of it.

4.2.5.3.1 Class Discriminability Measures

Class Discriminability Measure (CDM) (Kohn et al., 1996) is based on the idea of inhomogeneous buckets. The idea here is to divide the feature space into a number of hypercuboids. Each of those hypercuboids is called a "box". The dividing process stops when any of following criteria is fulfilled:

- box contains data from only one class;
- box is non-homogenous but linearly separable;
- number of samples in a box is lower that $N^{0.375}$, where N is the total number of samples in dataset.

If the stopping criteria are not satisfied, the box is partitioned into two boxes along the axis that has the highest range in terms of samples, as a point of division using among others median of the data.

Final result will be a number of boxes which can be:

- homogenous terminal boxes (HTB);
- non-linearly separable terminal boxes (NLSTB);
- non-homogenous non-linearly separable terminal boxes (NNLSTB).

In order to measure complexity, CDM uses only Not Linearly Separable Terminal Boxes, as, according to author (Kohn et al., 1996), only these contribute to Bayes error. That is however not true, because Bayes error of the set of boxes can be greater than the sum of Bayes errors of the boxes - partitioning (and in fact nothing) cannot by itself diminish the Bayes error of the whole dataset; however it can help classifiers in approaching the Bayes error optimum. So given enough partitions we arrive to have only homogenous terminal boxes, so the Bayes error is supposed to be zero, that is not true.

The formula for CDM is:

$$CDM = \frac{1}{N}\sum_{i=1}^{M}\left\{k(i) - \max[k(j \mid i)\right\} \tag{20}$$

where k(i) is the total number of samples in the *i*-th NNLSTB, k(j | i) is the number of samples from class j in the *i*-th NNLSTB and N is the total number of samples. For task that lead to only non-homogenous but linearly separable boxes, this measure equals zero.

4.2.5.3.2 Purity measure

Purity measure (Sing, 2003) is developed by Singh and it is presented with connection to his idea based on feature space partitioning called PRISM (Pattern Recognition using Information Slicing Method). PRISM divides the space into cells within defined resolution B. Then for each cell probability of class *i* in cell *l* is:

$$p_{il} = \frac{n_{il}}{\sum_{j=1}^{K_l} n_{jl}} \tag{21}$$

where n_{jl} is the number of points of class *j* in cell *l*, n_{il} is the number of points of class *i* in cell *l* and K_l is the total number of classes.

Degree of separability in cell *l* is given by:

$$S_{H(l)} = \sqrt{\left(\frac{k}{k\ \ 1}\right)\sum_{i=1}^{k}\left(p_{il} \quad 1/k\right)^2} \tag{22}$$

These values are averaged for all classes, obtaining overall degree of separability:

$$S_H = \sum_{l=1}^{H_{total}} S_{H(l)} \frac{N^l}{N} \tag{23}$$

where N^l signifies the number of points in the *l*-th cell, and *N* signifies total number of points. If this value was computed at resolution B then it is weighted by factor $w = \frac{1}{2^B}$ for B=(0,1,...31). Considering the curve (S_H versus normalized resolution) as a closed polygon with vertices (x_i,y_i), the area under the curve called **purity** for a total of *n* vertices is given as:

$$AS_H = \frac{1}{2}\left(\sum_{i=1}^{n-1}\left(x_i y_{i+1} - y_i x_{i+1}\right)\right) \tag{24}$$

The *x* axis is scaled to achieve values bounded within range [0, 1]. After the weighing process maximum possible value is 0.702, thus the value is rescaled once again to be between [0, 1] range.

The main drawback of purity measure is that if in a given cell, the number of points from each class is equal, then the purity measure is zero despite that in fact the distribution may be linearly separable. Purity measure does not depend on the distribution of data in space of single cell, but the distribution of data into the cells is obviously associated with data distribution.

4.2.5.3.3 Neighborhood Separability

Neighborhood Separability (Singh, 2003) measure is developed by Singh. Similarly to purity, it also depends on the PRISM partitioning results. In each cell, up to *k* nearest neighbors are found. Then one measure a proportion p_k of nearest neighbors that come from the same class to total number of nearest neighbors. For each number of neighbors *k, 1<=k<=*λ_{il} calculate the area under the curve that plots p_k against *k* as φ_j. Then compute the average proportion for cell H_l as:

$$p_l = \frac{1}{N^l}\sum_{j=1}^{N_l}\phi_j \tag{25}$$

Overall separability of data is given as:

$$S_{NN} = \sum_{l=1}^{H_{total}} p_l \frac{N^l}{N} \tag{26}$$

One compute the S_{NN} measure for each resolution B=(0, 1, … ,31). Finally, the area AS_{NN} under the curve S_{NN} versus resolution gives the measure of **neighborhood separability** for a given data set.

4.2.5.3.4 Collective entropy

Collective entropy (Singh & Galton, 2002), (Singh, 2003) measure the degree of uncertainty. High values of entropy represent disordered systems. The measure is connected to data partitioning algorithm called PRISM.
Calculate the entropy measure for each cell H_l:

$$E_l = \sum_{i=1}^{K_l} \left(\; p_{il} \;\; \log(p_{il}) \right) \tag{27}$$

Estimate overall entropy of data as weighted by the number of data in each cell:

$$E = \sum_{l=1}^{H_{total}} E_l \cdot \frac{N^l}{N} \tag{28}$$

Collective entropy for data at given partition resolution is defined as:

$$E_C = 1 - E \tag{29}$$

This is to keep consistency with other measures: maximal value of 1 signifies complete certainty and minimum value of 0 uncertainty and disorder.
Collective entropy is measured at multiple partition resolutions B=(0,…31) and scaled by factor $w = 1/2^B$ to promote lower resolution. Area under the curve of Collective Entropy versus resolution gives a measure of uncertainty for a given data set. That measure should be scaled as $AS_E = AS_E \big/ 0.702$ to keep the values in [0,1] range.

4.2.5.4 Other Measures

The measures described here are difficult to classify as they are very different in idea and it's difficult to distinguish common properties.

4.2.5.4.1 Correlation-based approach

Correlation-based approach (Rahman & Fairhurst, 1998) is described by Rahman and Fairhust. In their work, databases are ranked by the complexity of images within them. The degree of similarity in database is measured as the correlation between a given image and the remaining images in database. It indicates how homogenous the database is. For separable data, the correlation between data of different classes should be low.

4.2.5.4.2 Fisher discriminant ratio

Fisher discriminant ratio (Fisher, 2000) originates from Linear Discriminant Analysis (LDA). The idea of linear discriminant approach is to seek a linear combination of the variables which separates two classes in best way. The Fisher discriminant ratio is given as:

$$f_1 = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{30}$$

where μ_1, μ_2, σ_1, σ_2 are the means and variances of two classes respectively. The measure is calculated in each dimension separately and afterwards the maximum of the values is taken. It takes values from $[0,+\infty]$; high value signifies high class separability. To use it for multi class problem it is necessary however to compute Fisher discriminant ratios for each two-element combination of classes and later average the values.

Important feature of the measurement is that it is strongly related to data structure. The main drawback is that it acts more like a detector of linearly separable classes than complexity measure. The advantage is very low computational complexity.

4.2.5.4.3 Interclass distance measures

The ***interclass distance measures*** (Fukunaga, 1990) are founded upon the idea that class separability increases as class means separate and class covariance's become tighter. We define:
Within-class scatter matrix:

$$S_w = \sum_{i=1}^{L} P(\omega_i)\Sigma_i \tag{31}$$

Between-class scatter matrix:

$$S_b = \sum_{i=1}^{L} P(\omega_i)(\mu_i - \mu_0)(\mu_i - \mu_0)^T \tag{32}$$

Mixture (total) scatter matrix:

$$S_m = S_w + S_b \tag{33}$$

where μ_i are class means, $P(c_i)$ are the class probabilities, Σ_i are class covariance matrices, and $\mu_0 = \sum_{i=1}^{L} P(\omega_i)\mu_i$ is the mean of all classes.
Many intuitive measures of class separability come from manipulating these matrices which are formulated to capture the separation of class means and class covariance compactness. Some of the popular measures are:

$$J_1 = tr(S_2^{-1}S_1),\ J_2 = \ln\left|S_2^{-1}S_1\right|,\ J_3 = \frac{tr(S_1)}{tr(S_2)} \tag{34}$$

where S_1, S_2 are a tuple from among { S_b, S_w, S_m}, and *tr* signifies matrix trace. Frequently many of these combinations and criteria result in the same optimal features.

4.2.5.4.4 Volume of the overlap region

We can find volume of the overlap region (Ho & Baird, 1998) by finding the lengths of overlapping of two classes' combination across all dimensions. The lengths are then divided by overall range of values in the dimension (normalized), where d_o represents length of overlapping region, d_{max} and d_{min} represent consequently maximum and minimum feature values in specified dimension:

$$r_d = \frac{d_o}{d_{\max} - d_{\min}} \tag{35}$$

Resulting ratios are multiplied across all dimensions *dim* to achieve volume of overlapping ratio for the 2-class case (normalized with respect to feature space)

$$v_o = \prod_{i=1}^{\dim} r_d \tag{36}$$

It should be noted that the value is zero as long as there is at least one dimension in which the classes don't overlap.

Technique	Relation to Bayes error	Computing cost	Probability density functions	Number of classes
Chernoff bound	Yes	High	needed	2
Bhattacharyya bound	Yes	Medium	needed	2
Divergence	Yes	High	needed	2
Mahalanobis distance	Yes	Medium	not needed	2
Matusita distance	Yes	High	needed	2
Entropy measures	No	High	needed	>2
Classifier error	Potential	Depends on the classifier used		
k-Nearest Neighbours	Yes	High	not needed	>2
Parzen estimation	No	High	not needed	>2
Boundary methods	Yes	Medium	not needed	2
Class Discriminability Measures	No	High	not needed	2
Purity	No	High	not needed	>2
Neighbourhood separability	No	High	not needed	>2
Collective entropy	No	High	not needed	2
Correlation based approach	No	High	not needed	>2
Fisher discriminant ratio	No	very low	not needed	2
Interclass distance measures	No	Low	not needed	>2
Volume of the overlap region	No	Low	not needed	2
Feature efficiency	No	Medium	not needed	2
Minimum Spanning Tree	No	High	not needed	>2
Inter-intra cluster distance	No	High	not needed	2
Space covered by epsilon neighbourhoods	No	High	not needed	>2
Ensemble of estimators	Potential	High	depends	Depends

Table 3. Comparison of Classification Complexity Techniques

4.2.6 Discussion

Classification complexity estimation methods present great variability. The methods which are derived from Bayes error are most reliable in terms of performance, as they are theoretically stated. The most obvious drawback is that they have to do assumptions about a priori probability distributions. If the advantage of the methods designed using experimental (empirical) basis is that they are based uniquely on experimental data and do not need probability density estimates of distributions, these methods are as various as those relating the Bayes error's estimation and their performance are difficult to predict. Some methods are designed only for two-class problems, and as such they need special procedures to accommodate them to multi-class problem (like counting the average of complexities of all two-class combinations). The table 3, comparing complexity estimation methods, is aimed at several specific aspects which are:

- **Relation with Bayes error** which could be seen as a proof of estimator's accuracy up to some point;
- **Computational Cost**, this is especially important when the measurements are taken many times during the processing of problem, as in T-DTS case;

- **Need for probability density function estimates**;
- **Number of classes** in classification problem for which the method can be applied directly.

Recently, a number of investigations pushed forward the idea to combine several complexity estimation methods: for example by using a weighted average of them (Bouyoucef, 2007). It is possible that a single measure of complexity be not suitable for practical applications; instead, a hierarchy of estimators may be more appropriate (Maddox, 1990).

Using complexity estimation techniques based splitting regulation, T-DTS is able to reduce complexity on both data and processing chain levels (Madani et Al., 2003). It constructs a treelike evolutionary architecture of models, where nodes (DU) are decision units and leaves correspond to Neural Network - based Models (Processing Unit). That results in splitting the learning database into set of sub-databases. For each sub-database a separate model is built.

This approach presents numerous advantages among which are:

- simplification of the treated problem - by using a set of simpler local models;
- parallel processing capability - after decomposition, the sub-databases can be processed independently and joined together after processing;
- task decomposition is useful in cases when information about system is distributed locally and the models used are limited in strength in terms of computational difficulty or processing (modeling) power;
- modular structure gives universality: it allows using of specialized processing structures as well as replacing Decomposition Units with another clustering techniques;
- classification complexity estimation and other statistical techniques may influence the parameters to automate processing, i.e., decompose automatically;
- automatic learning.

However, our approach is not free of some disadvantages:

- if the problem doesn't require simplification (problem is solved efficiently with single model) then Task Decomposition may decrease the time performance, as learning or executing of some problems divided into sub-problems is slower than learning or executing of not split problem; especially if using sequential processing (in opposition to parallel processing);
- some problems may be naturally suited to solve by one-piece model - in this case splitting process should detect that and do not divide the problem;
- too much decomposition leads to very small learning sub-databases. Then they may loss of generalization properties. In extreme case leading to problem solution based only on distance to learning examples, so equal to nearest-neighbor classification method.

In the following section, we study the efficiency of T-DTS approach when dealing with classification problems.

4.2.7 Implementation and validation results

In order to validate the T-DTS self-organizing approach, we present in this section the application of such a paradigm to three complex problems. The first one concerns a pattern recognition problem. The second and third one are picked from the well know UCI repository: a toy problem (Tic-Tac-Toe) for validation purpose and a DNA classification one.

4.2.7.1 Application to UCI Reprository

Complexity estimating plays key-role in decomposition and tree-building process. In order to evaluate and validate T-DTS approach, we use two benchmarks from the UCI Machine Learning Repository (Bouyoucef, 2007). These two benchmarks are:

1. Tic-tac-toe end-game problem. The problem is to predict whether each of 958 legal endgame boards for tic-tac-toe is won for `x'. The 958 instances encode the complete set of possible board configurations at the end of tic-tac-toe. This problem is hard for the covering family algorithm, because of multi-overlapping.
2. Splice-junction DNA Sequences classification problem. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as EI sites), and recognizing intron/exon boundaries (IE sites). There are 3190 numbers of instances from Genbank 64.1, each of them compound 62 attributes which defines DNA sequences (ftp-site: ftp://ftp.genbank.bio.net) problem.

Next subsections include description of experimental protocol.

4.2.7.2 Experimental protocol

In the first case, Tic-tac-toe end game, we have used 50% of database for learning purpose and 50% for generalization purpose. At the node level (DU), competitive networks perform the decomposition. The following complexity estimation methods have been used: Mahalanobis, ZISC and Normalized mean. At T-DTS leaf level we have applied PU - LVQ.

Method type	*Max G_r (± Std. Dev.) (%)*
IB3-CI	99.1
CN2 standard	98.33 (± 0.08)
IB1	98.1
Decision Tree (DT)+FICUS	96.45 (± 1.68)
3-Nearest neighbor algorithm+FICUS	96.14 (± 2.03)
MBRTalk	88.4
Decision Tree (DT) Learning Concept	85.38 (± 2.18)
T-DTS&Mahalanobis com. est.	**84.551 (± 4.592)**
NewID	84.0
CN2-SD (add. weight.)	83.92 (± 0.39)
T-DTS&ZISC based com. est.	**82.087 (± 2.455)**
IB3	82.0
Back propagation +FICUS	**81.66 (± 14.46)**
T-DTS&Normalized mean com. est.	**81.002 (±1.753)**
7-Nearest neighbor	76.36 (± 1.87)
CN2-WRAcc	70.56 (± 0.42)
3-Nearest neighbor	67.95 (± 1.82)
Back propagation	**62.90 (± 3.88)**
Perceptron+FICUS	37.69 (± 3.98)
Perceptron	34.66 (± 1.84)

Table 4. Tic-tac-toe endgame problem

Method type	*Max G_r (± Std. Dev.) (%)*
3-Nearest neighbor algorithm+**FICUS**	86.30 (± 4.96)
Perceptron+**FICUS**	83.96 (± 6.22)
Decision Tree (DT)+**FICUS**	83.78 (± 4.61)
Back propagation algorithm+**FICUS**	83.42 (± 7.73)
T-DTS&ZISC based com. Est	**80.084 (± 3.176)**
3-Nearest neighbor algorithm	79.18 (± 6.32)
T-DTS&Mahalanobis based com. Est	78.672 (± 4.998)
Perceptron	76.34 (± 6.71)
T-DTS & Jeffries-Matusita based c.e.	75.647 (±8.665)
Decision Tree (DT) Learning Concept	73.55 (± 5.88)

Table 5. Splice-junction DNA sequences classification test

For DNA Benchmark, we have used 20% of database for learning purpose and 80% for generalization purpose. At the node level competitive networks perform the decomposition. The following complexity estimation methods have been used: Mahalanobis, Bhattacharya, ZISC, Purity and Fisher Measure. At T-DTS leaf level we applied PU - MLP.
For DNA Benchmark, we have used 20% of database for learning purpose and 80% for generalization purpose. At the node level competitive networks perform the decomposition. The following complexity estimation methods have been used: Mahalanobis, Bhattacharya, ZISC, Purity and Fisher Measure. At T-DTS leaf level we applied PU - MLP.
For both cases, a manual optimization has been performed. We have selected the decomposition units, the complexity estimation methods and the processing units that allow us to reach the highest performances in terms of generalization rate. In the next subsection, we present the results and compare them to those obtained by other approaches, mainly based on decision tree algorithms.

4.2.7.3 Results presentation and discussion

Various experiments have been conducted according to the experimental protocol described previously. Table 4 and Table 5 consolidate the results of our experiments and the results obtained by other classification approaches (Lavrac et al,. 2002), (Aha, 1991), (Markovitch & Rosenstein, 2002). As it is shown, we have resolved Tic-tac-toe endgame classification task with respectively 84.55%, 82.09% and 81.00% of generalization rates using *Mahalanobis,* ZISC and Normalized mean complexity estimators with a standard deviation of 4.59%, 2.46% and 1.75%. Taking into account standard deviation ratio, we can state that these results are equivalent as they are in the same range.
IB3-CI, CN2, IB1, DT and MBRTalk algorithms are rely on the instances extracting and their extrapolation. So, they are well adapted to board game problems. They also use domain knowledge to reach very high generalization rates (around 95%). Methods associated to FICUS use hypothesis driven construction strategies and especially FICUS algorithms allows to enhance the learning data base size.
In our case, T-DTS uses only data driven strategy. So, as we can see in Table 5, for Splice-junction DNA Sequences benchmark, taking into account the generalization rate standard deviation, the leading algorithms exhibit the same performances (3-Nearest neighbor+**FICUS,** Perceptron+**FICUS,** DT+**FICUS,** Back propagation +**FICUS and T-DTS&ZISC**). So, without using specific domain knowledge, T-DTS reaches a high

generalization rate. The T-DTS strength is its ability to solve hard classification problems without need of domain specific knowledge. In the experiments described in this paper, T-DTS structure optimization has been conducted manually (by the user). This is the main drawback.

5. Conclusion

Due the complexity of the actual systems based on heterogeneous methods, artificial neural networks approaches can reduce this complexity by modeling the environment as stochastic. Algorithms based on Neural Networks can take into account the dynamics of these environments with no model of dynamics to be given. Main idea of the approaches developed in this chapter is to take advantage from distributed processing and task simplification by dividing an initially complex processing task into a set of simpler subtasks using complexity estimation based loop to control the splitting process. An appealing consequence of combining complexity estimation based splitting and artificial neural networks based processing techniques is decreasing of user's intervention in specifying processing parameters. A first modular structure is proposed. We have focused our attention in some special kind of Constrained Based Routing in wired networks which we called QoS self-optimization Routing. In a second part, we study the use of T-DTS self-organizing and task adaptive abilities. Beside complexity estimation based self-organization and adaptation abilities of our approach, the neural nature of generated models leads to additional attractive features which are modularity and some universality of the issued processing system, opening new dimensions in bio-inspired artificial intelligence. Moreover, the distributed nature of T-DTS makes the processing phase potentially realizable using either parallel machine or network of sequential machines. Very promising results, obtained from experimental validation, involving either the presented set of classification benchmarks (problems) or the reported pattern recognition dilemma, show efficiency of such self-organizing multiple models' generator to enhance global and local processing capabilities by reducing complexity on both processing and data levels.

6. Acknowlegments

The present work has been partially supported by French Ministry of High Education and Research. A part of this project has also benefit from the French Eiffel Excellence Program of EGIDE.

7. References

Aha. D. W. (1991). Incremental Constructive Induction: An Instance-Based Approach, *Proceedings of the Eight International Workshop on Machine Learning, Morgan Kaufmann*.

Arbib (1989). *The Metaphorical Brain*, 2nd Edition, New York: Wiley.

Bates. J., Bryan Loyall A. & Scott Reilly W. (1989). Integrating reactivity, goals and emotion in a broad agent. *Technical Report CMU-CS-92-142*, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Bernet Y. (1998). Requirements of Diff-serv Boundary Routers, IETF Internet Draft.

Bouyoucef E. (2007). Contribution à l'étude et la mise en œuvre d'indicateurs quantitatifs et qualitatifs d'estimation de la complexité pour la régulation du processus d'auto organisation d'une structure neuronale modulaire de traitement d'information, *PhD Thesis*, LISSI, University Paris XII.

Bouyoucef E., Chebira A., Rybnik M., Madani K. (2005). Multiple Neural Network Model Generator with Complexity Estimation and self Organization Abilities", *International Scientific Journal of Computing*, vol.4, issue 3, pp.20-29.

Boyan J. A. and Littman M. L., Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach, *Advances in Neural Information Processing Systems 6,* Cowan, Tesauro and Alspector (eds).

Bridle, J.S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, *Neurocomputing: Algorithms, architectures and applications*, F.Foulgelman-Soulie and J Hérault, eds., New York: Springer-Verlag.

Bruske J., Sommer G. (1995). Dynamic Cell Structure, *Advances in Neural Information Processing Systems 7*, The MIT Press, Ed by G. Tesauro, pp.497-504.

Budnyk I., Chebira A., Madani K. (2008). Estimating Complexity of Classification Tasks Using Neurocomputers Technology, *International Scientific Journal of Computing*, under press.

Chebira A., Bouyoucef E., Rybnik M., Madani K. (2006). ATNS: An Adaptive Tree Neural Structure, *International Journal of Information Technology and Intelligent Computing*, IEEE Computational Intelligence Society, Vol. 1, N°3, pp.463-476.

Chi H., Ersoy O.K. (2002). Support Vector Machine Decision Trees with Rare Event Detection, *International Journal for Smart Engineering System Design*, Vol. 4, 225-242.

Cover T. M., Hart P. E. (1967). Nearest neighbour pattern classification. *IEEE Transactions on information theory*, Vol IT-13, pp 21-27.

Crawley E., Nair R., Rajagopalan B., Sandick H. (1998). A Framework for QoS-based Routing in the Internet, *RFC2386*, IETF, August.

Decker, K., Sycara, K., Williamson, M. (1997). Middle-Agents for the Internet ", *Proceedings of the 15th International Joint Conference on Artificial Intelligence,* Nagoya, Japan.

Ernst S. (1998). "Hinging hyper-plane trees for approximation and identification, *37th IEEE Conf. on Decision and Control*, Tampa, Florida, USA.

Fahlman S. E., Lebiere C. (1990). The Cascaded-Correlation Learning Architecture, *Advances in Neural Information Processing Systems 2*, Morgan Kauffman, San Mateo, pp.524-534.

Feng W., Kandlur D., Saha D., Shin K. (1997). Understanding TCP Dynamics in an Integrated Services Internet", *Proceedings of NOSSDAV*.

Ferber J. (1998). Multi-Agent Systems: Towards a Collective Intelligence, Reading, MA: Addison-Wesley.

Fisher A. (2000). The mathematical theory of probabilities, John Wiley ed.

Fukunaga K. (1972). *Introduction to statistical pattern recognition, School of Electrical Engineering*, Purdue University, Lafayette, Indiana, Academic Press, New York and London.

Fukunaga K. (1990). *Introduction to statistical pattern recognition*, Academic Press, New York, 2nd edition.

Gallager R. G.(1997). A minimum delay routing algorithm using distributed computations, *IEEE Transactions on Communications*, Vol. COM-25.

Gelenbe E., Lent R., Xu Z. (2002). Networking with Cognitive Packets, *Proc. ICANN 2002*, Madrid, Spain, August 27-30.

Goetz P., Kumar S., Miikkulainen R. (1996). On-Line Adaptation of a Signal Predistorter through Dual Reinforcement Learning, Proc. of the 13th Annual Conference Machine Learning, Bari, Italy.

Goonatilake S., Khebbal S. (1996). Intelligent Hybrid Systems: Issues, Classification and Future Directions, *Intelligent Hybrid Systems*, John Wiley & Sons Ed., pp.1-20.

Hannibal A. (1993). VLSI Building Block for Neural Networks with on chip Back Learning, *Neurocomputing*, n°5, pp.25-37.

Hartigan J. (1975). *Clustering Algorithms*. John Wiley and Sons Ed., New York.

Haykin S (1988). *Neural Networks– A Comprehensive Foundation*, Mcmillan College Publishing.

Haykin S. (1999). Neural *Networks – a Comprehensive foundation*, Prentice Hall Int.

Ho T.K., Basu M (2000). Measuring the complexity of classification problems, *Proceedings of the 15th Intenational Conference on Pattern Recognition*, Barcelona, Spain, pp. 43-47, September 3-8.

Ho T.K., Baird H.S. (1998). Pattern classification with compact distribution maps, *Computer Vision and Image Understanding*, vol. 70, no.1, pp.101-110.

Ho T.K., Basu M. (2002). Complexity measures of supervised classification problems, *IEEE Transactions on pattern Analysis and Machine Intelligence*, vol. 24, issue 3, March, pp.289-300.

Hoare C.A.R. (1962). Quicksort, *Computer Journal*, 5(1), pp.10-15.

Jacobson V. (1988). Congestion Avoidance of Network Traffic, Computer Communication" *Review*, vol. 18, no. 4, pp.314-329.

Jordan M.I., Jacobs R.A. (1993). Hierarchical mixtures of experts and the EM algorithm, *Technical Report AIM-1440*.

Jordan M.I., Jacobs R.A (2002). Learning in Modular and hierarchical systems, The Handbook of Brain Theory and Neural Networks, 2nd edition. Cambridge, MA: MIT Press, 2002.

Kohonen T. (1984). Self-Organization and Associative Memory, Springer-Verlag.

Kohn A., Nakano L. G., and Mani V. (1996). A class discriminability measure based on feature space partitioning, *Pattern Recognition*, 29(5), pp.873-887.

Krogh A., Vedelsby J. (1995). Neural Network Ensembles, Cross Validation, and Active Learning, *Advances in Neural Information Processing Systems 7*, The MIT Press, Ed by G. Tesauro, pp. 231-238.

Kumar S. and Miikkualainen R. (1998). Confidence-based Q-routing: an on-queue adaptive routing algorithm, *Proceedings of Symp. of Neural Networks in Engineering*.

Kumar S. and Miikkualainen R. (1997). Dual reinforcement Q-routing: an on-queue adaptive routing algorithm, *Proceedings of Symp. of Neural Networks in Engineering*.

Lang K. J. and Witbrock M. J. (1998). Learning to tell two spirals apart, *Proc. of the Connectionist Models Summer School*, Morgan Kauffman Ed., pp. 52-59.

Lavrac N., Flach P., Kavsek B., Todorovski L. (2002). Rule induction for subgroup discovery with CN2-SD, *Proc. Of IEEE ICDM*, pp. 266-273.

McLachlan, G.J., Basford, K.E. (1988). *Mixture Models: Interference and Applications to Clustering*", New York: Marcel Dekker.

Madani K., Chebira A., Rybnik M. (2003). Data Driven Multiple Neural Network Models Generator Based on a Tree-like Scheduler, *Lecture Notes in Computer Science n°2686*, SI on Computational Methods in Neural Modelling, (Jose Mira, Jose R. Alvarez Ed.) - Springer Verlag Berlin Heidelberg, ISBN 3-540-40210-1, pp.382-389.

Madani K., Thiaw L., Malti R., Sow G. (2005). Multi-Modeling: a Different Way to Design Intelligent Predictors, *LNCS 3512*, Ed.: J. Cabestany, A. Prieto, and F. Sandoval, Springer Verlag Berlin Heidelberg, pp.976 – 984.

Madani K. (2007). Toward Higher Level Intelligent Systems, *(Key-Note Paper), proceedings of IEEE- 6th International conference on Computer Information Systems and Industrial Management Applications (IEEE-CISIM'07)*, IEEE Computer Society, Elk, Poland, June, 28-30, pp.31-36.

Madani K. (2008). Artificial Neural Networks Based Image Processing & Pattern Recognition: From Concepts to Real-World Applications, *(Plenary Tutorial Talk, Key-Note Paper), proceedings of IEEE- 1st International workshop on Image Processing Theory, Tools and Applications (IEEE-IPTA'08)*, IEEE Computer Society, Sousse, Tunisia, November 23-26, pp. 19-27.

Maddox J. (1990). Complicated measure of complexity, *Nature*, vol. 344, *pp. 705.*

Maes, P. (1994). Social interface agents: Acquiring competence by learning from users and other agents, *Spring Symposium on Software Agents (Technical Report SS-94-03),* O. Etzioni (ed.), AAAI Press. pp.71-78.

Malkin G. (1998). RIP version2, Carrying Additional Information, *IETF RFC 1388 RFC 1993.*

Moy J. (1998). OSPF Version 2, *IETF RFC2328.*

Markovitch S., Rosenstein D. (2002). Feature Generation Using General Constructor Functions, *Machine Learning*, Springer Ed., Volume 45, N°. 1, pp.59-98.

Matusita K. (1967). On the notion of anity of several distributions and some of its applications. *Annals Inst. Statistical Mathematics*, Vol., 19, pp.181-192.

Mayoubi M., Schafer M., Sinsel S. (1995). Dynamic Neural Units for Non-linear Dynamic Systems Identification, *LNCS Vol. 930*, Springer Verlag, pp.1045-1051.

Mehmet I. S., Bingul Y., Okan K. E. (2003). Classification of Satellite Images by Using Self-organizing Map and Linear Support Vector Machine Decision Tree, *2nd Annual Asian Conference and Exhibition in the field of GIS*.

Mellouk A. (2008a). *End to End Quality of Service Engineering in Next Generation Heteregenous Networks,* ISTE/Wiley Ed.

Mellouk A., Hoceini S., Cheurfa M. (2008b). Reinforcing Probabilistic Selective Quality of service Routes in Dynamic Heterogeneous Networks, *Journal of Computer Communication*, Elsevier Ed., Vol 31, n°11, pp. 2706-2715.

Mellouk A., Lorenz P., Boukerche A., Lee M. H. (2007). Quality of Service Based Routing Algorithms for heterogeneous networks, *IEEE Communication Magazine*, Vol. 45, n°2, pp.65-66.

Mellouk A., Hoceini S., Amirat Y. (2006). Adaptive Quality of Service Based Routing Approaches: Development of a Neuro-Dynamic State-Dependent Reinforcement Learning Algorithm, *International Journal of Communication Systems,* Ed. Wiley InterSciences, Vol 20, n°10, pp.1113-1130.

Murray-Smith R. and Johansen T.A. (1997). *Multiple Model Approaches to Modeling and Control*, ed. Murray-Smith R. and T.A. Johansen, Taylor & Francis Publishers.

Naftaly, U., Intrator, N., Horn, D. (1997). Optimal ensemble averaging of neural networks, Network, vol.8, pp.283-296.

Ozdaglar A.E., Bertsekas D. P. (2003). Optimal Solution of Integer Multicommodity Flow Problem with Application in Optical Networks, *Proc. Of Symposium on Global Optimisation*.

Partridge C. (1992). A proposed flow specification, *IETF RFC1363.*

Pierson W.E. (1998). Using boundary methods for estimating class separability, *PhD thesis, Department of Electrical Engineering, Oho State University*.

Rahman A. F. R., Fairhurst M. (1998). Measuring classification complexity of image databases : a novel approach, *Proceedings of International Conference on Image Analysis and Processing*, pp.893-897.

Rosen E., Viswanathan A., Callon R. (1999). Multiprotocol Label Switching Architecture, IETF *Internet Draft draft-ietf-mpls-arch-06.txt*.

Saeed K., Tabedzki M., Adamski M. (2003). A View-Based Approach for Object Recognition, *Conradi Research Review Finland*, Vol. 2, Issue 1, pp.85-95.

Schapire R. E. (1999). A Brief Introduction to Boosting, *Proc. Of IJCAI*, pp.1401-1406.

Shenker S., Partridge C., Guerin R. (1997). Specification of guaranteed quality of service, *IETF RFC2212*.

Singh S. (2003). Multiresolution Estimates of classification complexity, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 25 , Issue 12, pp 1534 – 1539.

Singh S., A.P. Galton (2002). Pattern Recognition using Information Slicing Model (PRISM), *Proc. 15th International Conference on Pattern Recognition (ICPR2002),* Quebec.

Stallings W. (2001). MPLS , *Internet Protocol Journal*, Vol. 4, n° 3, pp.34-46.

Strassner J. (2003), *Policy-Based Network Management: Solutions for the Next Generation*, Morgan-Kaufmann Ed.

Subramanian D., Druschel P., and Chen J. (1997). Ants and reinforcement learning: A case study in routing in dynamic networks, *Proc. of the Fifteenth International Joint Conference on Artificial Intelligence*, vol. 2, pp.832-839.

Sung K. K., Niyogi P. (1995). Active Learning for Function Approximation, *Advances in Neural Information Processing Systems7*, pp.593-600.

Sutton R. S. and Barto A. G. (1994). *Reinforcement Learning*, MIT Press.

Takeshita, T., Kimura, F., Miyake, Y. (1987). On the Estimation Error of Mahalanobis Distanc, *Trans. IEICE* Journal, 70-D, pp.567-573.

Titterington D. M., Smith A.F., Makov V.E. (1985). *Statistical Analysis of Finite Mixture Distributions*, Wiley New York.

Tresp V. (2001). *Handbook for Neural Network Signal Processing*, CRC Press.

Turner J. (1986). New directions in communications (or which way to the information age), *IEEE Communications Magazine*, vol. 24(10), pp.8-15.

Vapnik V.N. (1998). *Statistical Learning Theory*, New York Wiley Ed.

Wasserman P. D. (1993)., Advanced Methods in Neural Computing, *New York: Van Nostrand Reinhold,* pp.35-55.

Wang Z. and Crowcroft J. (1996). QoS Routing for Supporting Resource Reservation, IEEE Journal on Selected Areas in Communications, 17 (8), pp. 1488-1504.

Watkins C. J., Dayan P. (1989). Q-Learning, *Machine Learning*, Vol.8, pp.279–292.

Watkins C. J. (1989). Learning from Delayed Rewards, *Ph.D. thesis, University of Cambridge, England*.

Welzl M. (2003). *Scalable Performance Signalling and Congestion Avoidance*, Kluwer Academic Publishers.

Wooldridge M., Jennings N. (1995). Intelligent agents: theory and practice, *The Knowledge Engineering Review*, Vol.10:2, pp.115-152.

Zhang L., Deering S., Estrin D. et Zappala D. (1993). RSVP : A New Resource ReSerVation Protocol, IEEE *Network*, vol. 7, No 5, pp.8–18.