3003

N90-22319

A KNOWLEDGE-BASED SYSTEM WITH LEARNING FOR COMPUTER COMMUNICATION NETWORK DESIGN

by

Samuel Pierre
Télé-université, Université du Québec
4835 Christophe-Colomb, Montréal, Qué. H2J 4C2
Tel: (514) 522-3540   Fax: (514) 522-3540

Hai Hoc Hoang
École Polytechnique de Montréal
Campus de l'Université de Montréal
C.P. 6079 - Succ. "A", Montréal, Qué. H3C 3A7

Evelyne Hausen-Tropper
Département de Mathématiques et Informatique, UQAM
C.P. 8888 - Succ. "A", Montréal, Qué. H3C 3P8

ABSTRACT

Computer communication network design is well-known as complex and hard. For that reason, the most effective methods used to solve it are heuristic. In this paper, we list weaknesses of these techniques, and present a new approach based on artificial intelligence for solving this problem. This approach is particularly recommended for large packet-switched communication networks, in the sense that it permits to ensure high degree of reliability, and offers a very flexible environment dealing with many relevant design parameters as link cost, link capacity and message delay.

KEYWORDS: knowledge-based system, communication network design, inductive learning.

1. INTRODUCTION

A computer communication network is generally modelled as a valued graph whose nodes represent computers and arcs communication links [2, 3]. Before implementing protocols allowing the operation of a network, we must determine the manner whose nodes are linked between them and the capacity of each link. Such a problem is known in the literature as the topological design of computer communication networks [2, 8, 13].

This paper proposes a knowledge-based system with inductive learning for solving this problem. It is organized as follows: section 2 sets up background for the topological design problem and underlines some weaknesses of conventional methods; section 3 puts forward the architecture and the running of the knowledge-based system; section 4 deals with the knowledge organization within the system; section 5 conceptualizes the inductive learning module and states the learning algorithm; section 6 summarizes some results and makes concluding remarks.

2. THE TOPOLOGICAL DESIGN PROBLEM

In this section, we first present prerequisite definitions and notations, a formulation of the topological design problem, and finally the conventional methods used to solve it.

2.1 DEFINITIONS AND NOTATIONS

Let us consider a set of nodes N and a set of edges A connecting these nodes. Let n be the cardinality of N and m the cardinality of A. A "topology" is an undirected graph G=(N,A), where each edge represents a full duplex link with a given capacity, expressed in bits per second (bps).

There are [n(n-1)/2] possible links between all pairs of nodes. This number is denoted by m_{max}.

So, the basic characteristics of a topology are its topological configuration materialized by A, which can be represented by a binary characteristic vector $t=(t_k)$, $k=1,2,\ldots,m_{max}$, and its capacity assignment. For convenience, we shall use i to denote the i-th node and k=(i,j) the edge joining node i and node j, with $i,j = 1,2,\ldots,n$, $i\neq j$, and $k = 1,2,\ldots,m_{max}$. Such a numbering scheme can easily be devised. Note that:

$$\sum_k t_k = m \ .$$

It follows that various topological configurations can be obtained by varying the set of links.

For a given topology, each link k of the topological configuration $t=(t_k)$ is assigned a capacity C_k, such that $t_k=0$ implies $C_k=0$. $C=(C_k)$ denotes the capacity vector associated with the topology. Consequently, a topology will denoted by (t,C). Each link k of t is associated with a cost D_k which is a function of its capacity C_k:

$$D_k = d_k(C_k) \qquad (1)$$

In reference to the running network, all information or message to be transmitted is first broken in small parts called "packets". Independently passing from one node to another, these packets are reassembled at the destination: this is the packet-switching principle [11].

Let $1/\mu$ be the average packet length expressed in bits/packet and γ_{ij} the required traffic in packets/second from source i to destination j. The traffic requirement r_{ij}, expressed in bits/second, can be defined as follows:

$$r_{ij} = \gamma_{ij}/\mu \qquad (2)$$

Then the traffic matrix is $R=(r_{ij})$, i, j = 1,2,...,n, with $i\neq j$.

In order to satisfy the traffic requirements, it is first necessary to choose a routing strategy. The choice is generally motivated by computational considerations and should make the link flow computation relatively easy. If we denote by $f_k^{(p,q)}$ the flow in bps on link k produced by packets travelling from source p to destination q, the total flow f_k in link k is given by:

$$f_k = \sum_{\substack{p=1 \\ (p \neq q)}}^{n} \sum_{q=1}^{n} f_k(p,q) \qquad (3)$$

Consequently, the overall network flow can be represented by a flow vector:

$$f = (f_k) \qquad (4)$$

For a given topological configuration, f is uniquely determined by the routing strategy. Note that $C_k=0$ implies $f_k=0$. Thus, $t_k=0$ implies $f_k=0$.

The routing problem concerns the choice of the best path, according to a given criterion, for traffics from a source to a destination, provided that there exist multiple routes between all pairs of nodes. Such a situation materializes the concept of K-connectivity often used as a network reliability metric.

There are two types of connectivity: the edge-connectivity C_e, and the node-connectivity C_n. The edge-connectivity between two nodes i and j can be defined as the minimum number of edges whose removal will disconnect these two nodes. If we call edge-disjoint paths the paths which have no edges in common, then such an edge-connectivity is equivalent to the number of edge-disjoint paths between the two considered nodes. So, the edge-connectivity of a network is the minimum of the edge-connectivities amongst all pairs of nodes, that is, the number of edge-disjoint paths connecting the most critically connected pair of nodes.

Similarly, the node-connectivity between two nodes i and j is the minimum number of nodes which must be removed from the network to disconnect these two nodes. If we take the minimum node-connectivity over all pairs of nodes, we obtain the node-connectivity of the network, C_n.

If we denote by d the degree of a network, that is, the minimum degree of all nodes, it can be shown that $C_n \leq C_e \leq d$. So, for design purposes and for a given degree of connectivity, the node-connectivity C_n is more demending than the edge-connectivity.

Packets take time for travelling from source i to destination j. The average packet delay from i to j, is denoted by Z_{ij}. The overall average delay T can be generally expressed as follows:

$$T = \frac{1}{\gamma} \sum_{\substack{i=1 \\ (i \neq j)}}^{n} \sum_{j=1}^{n} \gamma_{ij} Z_{ij} \qquad (5)$$

where γ is the total traffic in the network which can be obtained by summing the γ_{ij}'s.

Based on a set of simplifying assumptions, a useful and easily computable expression for the overall average delay has been derived [2]:

$$T = \frac{1}{\gamma} \sum_{k \in A} \frac{f_k}{C_k - f_k} \qquad (6)$$

So, the overall average delay T appears as a function of link capacities C_k and link flows f_k, for all links included in the topological configuration which is considered.

2.2 PROBLEM FORMULATION

The network topological design problem can be formulated as follows [2, 4, 7, 8]:

Given:
- Switching node locations
- Traffic requirements $R=(r_{ij})$
- Capacity options and associated costs for all potential links
- Maximum overall average delay allowed T_{max}

$$\text{Min } D = \sum_k d_k(C_k) \qquad (7)$$

Over:
- Topological configuration t
- Capacity vector C
- Flow vector f

Subject to:
- $f \leq C$ (component wise)
- t is a K-connected topological configuration, $2 \leq K \leq n-1$
- $$T = \frac{1}{\gamma} \sum_{k \in A} \frac{f_k}{C_k - f_k} \leq T_{max} \qquad (8)$$

This problem is known to be NP-hard [5, 6]. The first difficulty arises from the combinatorial character of link selection which involves some explosion risk.

Another major difficulty is the nonlinearity of relevant functions such as communication link costs D, and the average packet delay T. For that reason, only local optima are guaranteed by Kuhn-Tucker conditions [2].

Finally, link capacities are only available in some discrete values as 2400, 4800,9600, 19200, 50000 bps, etc. That constitutes a nontrivial problem which cannot be efficiently solved by discrete programming techniques, because of the size of the problem [2].

2.3 CONVENTIONAL METHODS

Taking into account the previous considerations, it is not suitable to search for an exact solution. Only approximate methods are recommended for finding realistic and suboptimal solutions. In fact, the combinatorial nature of this problem suggests the use of heuristics for attempting to reduce the search space of candidate topologies.

Most of conventional procedures use heuristics, and produce suboptimal solutions. They essentially correspond to search procedures which optimize network structure by sequentially changing small parts of a larger network [2, 9, 10, 12, 14].

In the case of small size networks (about 30 nodes), the most popular solution methods are Branch Exchange (BXC), Concave Branch Elimination (CBE) and Cut Saturation (CS) [2, 12]. Lavia and Manning [10] have proposed perturbation techniques under connectivity and diameter constraint. Moreover, for large computer networks (more than 100 nodes), Kleinrock and Kamoun [9] have elaborated optimal clustering structures for hierarchical topological design, while Chen et al. [1] proposed an extended model and a solution method for network topological design, taking into account the selection of switching node locations.

These methods present two major disadvantages:

- they cannot deal with high degree of reliability (connectivity greater than 2) which is required by the large computer networks;

- they require human intervention for obtaining alternate solutions, by minor modifications on a given solution.

3. A KNOWLEDGE-BASED APPROACH

This approach consists in generating an initial topology well characterized, on which some perturbations are applied by an knowledge-based system in order to obtain a good suboptimal solution, lower-cost topology satisfying all constraints of the specified problem [4, 7, 8]. An inductive learning module is also available for the evaluation of rules already stored in the rule base and the generation of new rules from knowledge contained in the system. In this section, we explain the proposed approach and present the architecture of the system.

3.1 GENERAL ORGANIZATION

From data specified by a user, a good starting topology is first generated. Rules are applied on this topology for providing positive examples (good topologies satisfying all constraints, particularly the delay constraint) and negative examples (good topologies violating the delay constraint). All positive examples determine a set of feasible good topologies, and a solution corresponds to the least cost topology of this set. Furthermore, the generated examples are submitted to an inductive learning module, whose the role is to improve the rules for generating examples. More precisely, this module deals with:

- the detection and correction of rule inconsistencies;
- the elimination of rule redundancies;
- the addition of new knowledge;
- the rule updates;
- etc..

The system is decomposed into four major functional modules, as follows:

- the initial topology generator which produces a starting topology satisfying the K-connectivity constraint;

- the example generator playing the role of an rule-based system or expert system, and using heuristic perturbations for generating positive and negative examples from the starting topology;

- the inductive learning module which receives a set of nondeterministic rules and a collection of representative examples, and improves the rule base; and

- the user interface module which permits interactions between (expert and nonexpert) users and the system, particularly in order to specify data and parameters characterizing the network to design.

3.2 ARCHITECTURE OF THE SYSTEM

Figure 1 gives a detailed representation of the problem-solving system. In order to understand it, we first present some basic definitions, then a summary of used notations and finally the general algorithm.

3.2.1 Basic Definitions

The rules can be deterministic or nondeterministic. Deterministic rules generally express the analytic properties of generated initial topologies. They serve to describe absolute truth contexts, and are consequently accompanied by likelihood factors equal to one. On the other hand, a rule is nondeterministic when it refers to an uncertainty situation, expressed by a likelihood factor less than one. These rules are inspired either by conventional heuristics or experimental methods of machine learning from examples. Obviously, the likelihood factors are nonnegative real numbers not greater than one.

When a starting topology is submitted to the example generator, all applicable rules are applied to it, in order to generate new derived topologies, called examples, which are stored in the knowledge base. This is called a "perturbation cycle". For the first perturbation cycle, the starting topology is generated by the initial topology generator and is consequently called an "initial topology". For the subsequent perturbation cycles, the starting topology is somehow selected among these derived exemples and is renamed a "reference topology". So, for a given design task, it can exist many reference topologies, but only one related initial topology. Similarly, we can define a "learning cycle" as the process allowing to modify the base of nondeterministic rules, on user requests.

3.2.2 Summary of Notations

The meanings of notations used in figure 1 are as follows:

F :	an information vector submitted by the user interface module to the initial topology generator; it contains the specifications which are necessary to start the system.
q :	a question/answer vector exchanged between the user interface module and the system; according to the nature of the dialogue, the example generator appears as the unit which interprets, formulates and fulfils user requests.
X :	an example base acting as input to the inductive learning module, which is accumulated during the life time of the system.
E :	an example base accumulated during the solution of the current design problem.

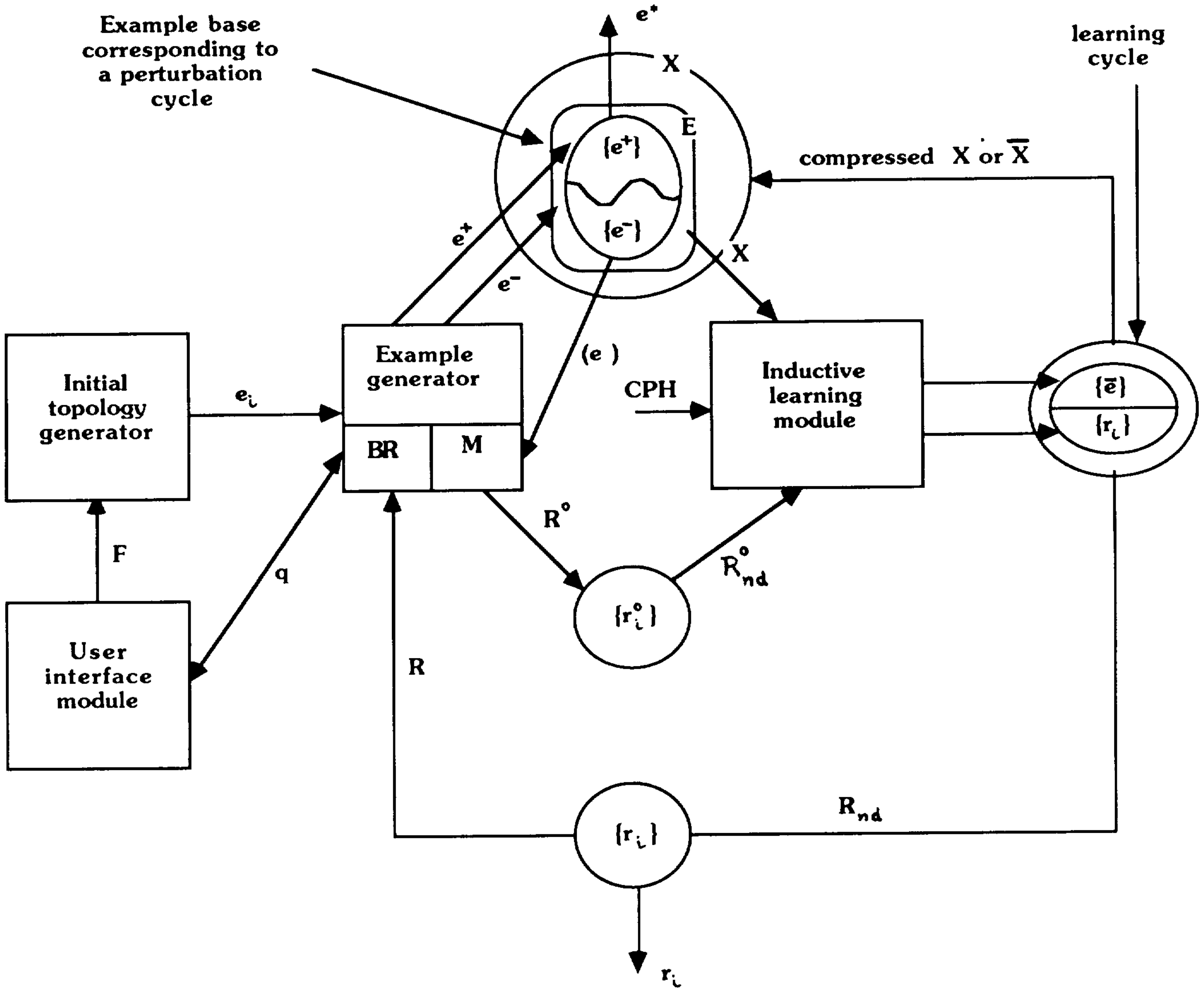

Fig. 1 - Detailed architecture of the system

e_i: an initial topology.

BR: a rule base allowing to generate examples or facts which constitute E.

M : an inference engine, allowing to apply the rules in BR to the examples in E, which are considered as facts.

e^+: a positive example provided by the example generator.

e^-: a negative example provided by the example generator.

e^*: the best feasible solution so far obtained during one or more perturbation cycles already performed.

$\{e^+\}$: a set of positive examples accumulated.

$\{e^-\}$: a set of negative examples accumulated.

(e): the least cost example in E given to the example generator to start a new perturbation cycle.

$\bar{e}$: a representative example selected by the inductive learning module.

CPH: a hypothesis preference criterion, allowing to discriminate plausible assumptions in the learning process.

R^0: a base of initial rules, including both deterministic and nondeterministic rules.

r_i^0: a nondeterministic initial rule.

$\{r_i^0\}$: a subset of nondeterministic initial rules.

R^0_{nd}: a base of nondeterministic initial rules.

r_i: a nondeterministic rule resulting from a learning cycle.

$\{r_i\}$: a subset of nondeterministic rules accumulated during a learning cycle.

R_{nd}: a base of nondeterministic rules resulting form a learning cycle.

R: a new rule base, obtained by an union of the subset R_d of deterministic rules and the base R_{nd} of nondeterministic rules resulting from a learning cycle.

3.2.3 General Algorithm

The general algorithm is defined by the following steps:

Step 1 : The user interface module transmits to the initial topology generator the information vector F specifying the context of the design.

Step 2 : The initial topology generator produces a starting example or initial topology e_i satisfying the specification vector F. The example base E is empty.

Step 3 : The example generator applies the rule base to the starting topology to generate positive examples e^+ and negative examples e^- satisfying the specification vector F.

All generated examples are included in E.

Step 4 : At the end of a perturbation cycle, the system proposes the best feasible feasible solution obtained, that is,

$$e^* = \min_D \{e^+\}$$

Step 5 : The user interface module, via the vector q, possibly asks for explanations about the proposed solution, generation of a new solution, learning new rules, and so on.

Step 6 : If a new solution is required, the example generator applies again the rule base to a new reference topology or example (e), which is the least cost example in E, that is, go back to step 3.

Step 7 : If a learning cycle is required, the inductive learning module receives the example base X, the hypothesis preference criterion CPH and the base of nondeterministic rules R^0_{nd}, induces new nondeterministic rules $\{r_i\}$, and constructs a new abstract and compressed representation of X called $\bar{X}$.

Step 8 : At the end of a learning cycle, the example base X is updated by the assignment $X := X \cup X$, the base of nondeterministic rules R^0_{nd} is then replaced by the new base R_{nd}. The result of that is a new rule base defined by the assignment

$R := R_d \cup R_{nd}$.

Step 9 : The user interface module, via the vector q, possibly asks to display the new induced rules, to modify the rule base, to submit a new vector q, to stop the running of the system.

Step 10 : Stop.

4. KNOWLEDGE ORGANIZATION

The example generator which is represented in figure 1 acts as a knowledge-based module. It essentially consists of a rule base and an inference engine. In this section, we explain the operating of the example generator and deal with the rule base organization.

4.1 THE EXAMPLE GENERATOR

When the initial topology generator provides a particular initial topology considered as a starting topology (t^0,c^0) from the problem specifications, the example generator receives this topology and applies its knowledge base to transform (t^0,c^0) into (t^i,c^i), i= 1,2,...

The initial topology is characterized by the following attributes: a number of nodes, a number of links, a link flow vector, a link capacity vector, a link utilization vector, a degree of connectivity, an average delay, a total communication design and other secondary attributes mainly used by the learning process. These

[illegible] constraint while

are the features of the concept of example. Moreover, an example whose the average delay is greater than the maximum allowed delay is classified as a

rule R_i generates an example e_i which is stored in the short-term example base. When all the rules were considered with regard to that starting

applying link deletion rules. They are based on the set of propositions expressing the analytical properties of initial topologies. The following is an example:

If 1) the related initial topology contains more than 4 nodes
2) the related initial topology has a degree of connectivity equal to 2
Then at most (n- 3) links can be deleted from the reference topology to obtain a derived topology.

5. Inductive Learning

Inductive learning is defined as the acquisition of knowledge by means inductive inferences which are effectuated from facts provided by a teacher or an environment (Mitchell, Carbonell and Michalski 1986). The related module aims at improving the rule base in order to achieve more refined inferences. In this section, we first formulate our inductive learning problem, then we present an appropriate algorithm.

5.1 LEARNING CHARACTERIZATION

The implemented learning is incremental, with partial-memory of past examples [15]. It can be formulated in the following terms:

Given:
- a nondeterministic rule base, R_{nd}
- an example base, E
- with each rule r_i of R_{nd} is associated a candidate hypothesis space H
- an hypothesis preference criterion CPH which permits to select amongst a set of plausible hypotheses.

Objective:
- Find - by generalization, specialization or reformulation- a new nondeterministic rule base R_{nd} such as the description $R = R_d \cup R_{nd}$ consistently covers the near total of good examples stored in E.

5.2 Learning Algorithm

The proposed inductive learning algorithm is defined by the following steps:

Step 1 : Receive the set R^0_{nd} of nondeterministic initial rules and do $R_{nd} := R_{nd}$;

Step 2 : Receive from the example base E an example e, then build the subset $R_{nd}(e)$ of nondeterministic rules which has generated e, where $e \in E$ and $R_{nd}(e) \subset R_{nd}$;

Step 3 : If e is a positive example, then the nondeterministic rules $R_{nd}(e)$ which have generated it are checked:
. update the likelihood factors, if necessary;
. make a list of discriminating properties of e which could imply the generality of rules in the subset $R_{nd}(e)$;
. generalize, if necessary, the rules of R_{nd} taking into account the related hypothesis spaces; if there is conflict in the

selection of hypotheses, use the given hypothesis preference criterion to solve it;

Step 4 : If e is a negative example, then at least one of nondeterministic rules which have generated it is not confirmed:
. update the likelihood factors;
. make a list of involved discriminating properties of e;
. specialize or reformulate the rules of $R_{nd}(e)$ taking into account the related hypothesis spaces; if there is conflict in the selection of hypotheses, use the given hypothesis criterion to solve it;

Step 5 : If at least one example of the rule base E is not yet considered, then go to step 2;

Step 6 : Stop.

6. Computational Experience and Concluding Remarks

In order to evaluate the efficiency of our method, now implemented on a typical IBM PC AT, we have considered a set of fifty network problems, randomly generated, which have been also solved by the cut saturation method. For a convenient comparison with the cut saturation method, our experience is based on the following choices:

- the number of nodes is always kept equal to 25;
- traffic is constant between each pair of nodes;
- the degree of connectivity is always equal to 2;
- the maximum delay is T_{max} = 200 msec;
- the average size of data packets is equal to 1000 bits/packet.

For a given problem, a solution is characterized by a topological configuration t, a capacity vector C, a flow vector f, an average delay T, a transmission links cost D, and CPU time. In 80 % of cases, solution provided by our method gives a lower cost than the cut saturation solution. Furthermore, in 90 % of cases, the CPU time required to provide a solution is lower in the case of SIDROGT than cut saturation.

In this paper, we have presented an artificial intelligence approach for solving the network design problem. The heart of this approach is constituted by an expert module which receives an starting topology and operates on it local transformations by means heuristic perturbations. An inductive learning module is used for improving the efficiency of those transformations.

Solution provided by such a system is obviously suboptimal. But, it is made up by an computationally efficient and flexible process which allows to attempt a new solution by initiating a new perturbation cycle, or to improve the rule base by initiating a new learning cycle. Furthermore, another advantage of that system is the high degree of connectivity which it permits. The initial topology generator provides topologies which are 1, 2,..., (n-1) connected (where n denotes the number of nodes), satisfying by the way the reliability constraint. That is truly innovative in comparison with the other methods, which are limited to the 2-connectivity and generally start with an unrefined starting topology. The degree of connectivity is preserved by both the knowledge-based module and the inductive learning module. So, it is not necessary to run a time-consuming connectivity-restoring algorithm. For those reasons, such a system is suitable for designing large scale computer networks, where a high level of reliability is required.

References

[1] K.J. Chen, J.F. Stach, and T.H. Wu, "A New Method for Topological Design in Large, Traffic Laden Packet Switched Networks,"Proc. Ninth Data Communications Symposium, ACM SIGCOMM Computer Communication Review, Vol.15, No.4, 1985, pp.115-121.

[2] M. Gerla and L.Kleinrock, "On the Topological Design of Distributed Computer Networks," IEEE Trans. on Communications, Vol.Com-25, No.1, 1977, pp.48-60.

[3] R.R. Boorstyn and H. Frank, "Large-Scale Network Topological Optimization," IEEE Trans. on Communications, Com-25, No.1, 1977, pp.29-47.

[4] E. Hausen-Tropper, H.H. Hoang, S.Pierre, "Approche Système Expert pour la Conception de Réseaux Téléinformatiques de Grande Taille", Neuvièmes Journées Internationales sur les Systèmes Experts et leurs Applications, Avignon, France, in Intelligence Artificielle et Télécommunications, 1989, pp. 134-148.

[5] R.M. Karp, "Combinatorics, Complexity, and Randomness," CACM, Vol.29, No.2, 1986, pp.97-109.

[6] D.S. Johnson, J.K. Lenstra and A.H.G. Rinnooy, "The Complexity of the Network Design Problem," Networks, Vol.8, No.4, 1978, pp.279-285.

[7] S. Pierre and E. Hausen-Tropper, "Design Topologique de Réseaux Téléinformatiques de Grande Taille: une Nouvelle Approche," IEEE Montech'86, Montréal, 1986, pp.370-372.

[8] S. Pierre, H.H. Hoang, E. Tropper, "An Expert Systems Application in Computer Network Topological Design," IASTED International Conference on Expert Systems, Theory & Applications, Zurich, Switzerland, 1989, pp. 139-142.

[9] L. Kleinrock and F. Kamoun, "Optimal Clustering Structures for Hierarchical Topological Design of Large Computer Networks," Networks, Vol.10, 1980, pp.221-248.

[10] A. Lavia and E.G. Manning, "Perturbations Techniques for Topological Optimization of Computer Networks," Proceedings of Fourth Data Communications Symposium, 7-9 oct. 1975, Quebec City, pp.4-16, 4-24.

[11] E.A. Sykes and C.C. White, "Specifications of a Knowledge System for Packet-Switched Data Network Topological Design," Expert Systems in Government Symposium, McLean, VA, Oct. 1985, pp. 102-110.

[12] M. GERLA, H. FRANK, W. CHOU, and J. ECKL, "A Cut Saturation Algorithm for Topological Design of Packet-Switched Communication Networks," Proc. Nat. Telecom. Conf., 1974, pp. 1074-1085.

[13] H.H HOANG, "Topological Optimization of Networks: A Nonlinear Mixed Integer Model Employing Generalized Benders Decomposition," IEEE Transactions on Automatic Control, Vol. Ac-27, No. 1, 1982, pp. 165-169.

[14] R. KRONZ, S. LEE, and M. SUN, "Practical Design Tools for Large Packet-Switched Networks," Proc. IEEE INFOCOM, 1983, pp. 591-599.

[15] T.M. MITCHELL, J.G. CARBONELL, and R.S. MICHALSKI, MachineLearning: A Guide to Current Research, Kluwer Academic Publishers, 1986.