Journal of Intelligent and Robotic Systems **4**: 303-319, 1991.
© 1991 *Kluwer Academic Publishers. Printed in the Netherlands.*

Expert Maintenance Systems in Telecommunication Networks

YUVAL LIROV* and ON-CHING YUE
AT&T Bell Laboratories, Holmdel, NJ, U.S.A.

(Received: 5 July 1990; in final form: 8 January 1991)

Abstract. Our survey of some 40 network maintenance expert systems reveals their main shortcoming, which is the difficulty to acquire troubleshooting knowledge both when initializing the expert system and after its deployment. Additionally, the state-of-the-art troubleshooting expert systems do not optimize troubleshooting cost. We present the *AO** algorithm to generate a network troubleshooting expert system which minimizes the expected troubleshooting cost and learns better troubleshooting techniques during its operation.

Key words. Telecommunications networks, troubleshooting, expert systems, learning.

1. Introduction

Telecommunications is a significant expert systems applications domain. (Hannan, 1987) describes three areas of networking in which expert systems can play an important role. These are network design, monitoring, and troubleshooting. It is argued that as the importance of networking increases, the availability of expert systems to assist in handling the responsibilities associated with networks will play a key factor in their overall effectiveness.

The core of an expert system consists of a knowledge base and an inference engine that operates on the knowledge base to develop a solution. Additionally, an expert system contains an interface to the end-user or to an array of sensors and effectors to communicate with the outside world. Expert system construction involves representing and accumulating the problem-solving knowledge in the knowledge base and developing an appropriate inference strategy. In this paper we discuss the expert systems aimed at network maintenance. We survey the design issues of over 40 automatic diagnostic and repair expert systems for telecommunications networks in terms of their structures, knowledge representations, inference mechanisms, and interfaces with developers and end-users. We also present a novel algorithm to generate a learning network troubleshooting expert system which minimizes the expected troubleshooting cost.

First we place the process of maintenance in the broad perspective of other issues in the management of telecommunications networks. While mentioning different network operations activities, we briefly present the main challenge of every endeavor.

* Current affiliation: Salomon Technology Services, Rutherford, New Jersey, USA.

We note that all network operations problems actually assume a form of a general resource management problem. Solving such problems includes methods which work with a model of the system (consisting of both quantitative and qualitative components), and reason with that model using various forms of search. The problem can take on several further dimensions in its full generality, including a requirement to learn from past experience.

1.1. THREE FACETS OF TELECOMMUNICATIONS MANAGEMENT

Telecommunications operations are often subdivided in three broad classes of activities (Wright and Vesonder, 1990): provisioning, maintenance, and administration. Provisioning is the act of providing the network with the necessary services. This includes the forecasting of traffic demand and planning and engineering the required alterations to the network. The difficulty of provisioning is that the networks must often be configured to perform under previously unconsidered environments and scenarios and carry unknown traffic rates.

Maintenance is the process of keeping the network according to the specifications of its operation. Maintenance involves monitoring, testing, troubleshooting, repair, and preventive treatment of all the components of the network. The challenge of maintenance is that the network must be troubleshot on-the-fly. Finally, network administration consists of all the rest of the network operations activities such as message routing, billing, and record keeping. The hardship of the administration is that the traffic needs to be controlled in real time.

1.2. ROAD MAP

The paper proceeds as follows. We first describe the state of the art in the domain of the network maintenance expert systems. This section is subdivided further into the subsections about monitoring expert systems and about the troubleshooting expert systems. The third section is devoted to the new network troubleshooting knowledge acquisition algorithm.

2. Maintenance Expert Systems: State of the Art

Having described the main network operations, we now concentrate on maintenance. First, we digress to mention the cornerstone of the telecommunications networks: the operations systems. Then we discuss the two different aspects of maintenance: monitoring and troubleshooting, and name an all-important maintenance role played by the operations systems.

Successful operation of telecommunications networks critically rely on the support of specialized computer systems, often called operations systems. Such systems, e.g., switching machines, include network monitoring as part of their regular work activities. Network monitoring produces messages to the human operator of the switching machine. These messages report about every telephone call failure.

2.1. MONITORING

As a switching machine processes hundreds of thousands of calls a day, a correspondingly huge number of calls are not completed for a variety of reasons. Sorting out the messages indicating genuine troubles from false alarm messages is a difficult combinatorial problem. Automated network monitoring removes the manual burden of maintaining a consistent description of the network, as well as operating the network in a more efficient manner. Monitoring expert systems, however, are neither intended to correct the malfunction nor to suggest a remedy – they merely alarm the operator about a problem situation. Below we describe several such expert systems.

Ferrara *et al.*, (1989) describe an expert system for continuous monitoring and diagnosing of the Italian ITELPAC network. The system directly receives the alarm messages from the network and, in real-time, deduces the possible anomalies which caused these indications. Knowledge for alarm analysis expresses empirical association between messages and possible anomalies, and allows the anomaly severity to be determined.

Corn *et al.*, (1988) decribe TOPAS-ES (Trunk Operations, Provisioning, and Administration System), an expert system that diagnoses transmission and signalling problems arising on telephone trunks in the AT&T long-distance network. It runs autonomously, issuing operations system commands to retrieve trouble reports and perform mesurements on the problematic trunks. It is also a distributed system, since difference copies of TOPAS-ES must communicate with one another in order to coordinate the diagnosis process from both ends of a trunk.

Kaye and Pagurek, (1989) discuss the role of an expert system for monitoring troubles in a wide-area network of network-switching, private, automatic computer exchanges embedded in a wide-area computer network (WAN). They present a taxonomy of diagnostic problems based on a the breadth of view of the network necessary to detect and diagnose them and a further taxonomy of the types of patterns of events which must be detected. The architecture and experimental development of an expert monitor for a single exchange within such a network is presented with some emphasis on a design approach for eventual real-time implementation.

The GEMS (Generalized Expert Maintenance System) trunk trouble analyzer (Khan and Dube, 1987) is a knowledge-based expert system prototype designed to diagnose and interpret faults on trunks, indicate repair priorities, and suggest appropriate test or performance monitoring stragies for trunk maintenance. A prototype of the expert system has been built using the KEE knowledge engineering environment, an expert system-building software tool, on a Symbolics 3670 Lisp machine.

A prototype expert system for maintenance of Northern Telecom's DMS-family digital switches is presented in Peacocke and Rabie, 1988. The prototype is based on using product documentation and experience obtained from craftspeople and field technical support to create the knowledge base for an expert system which diagnoses and chooses corrective actions for DMS-100 problems.

In military applications, efficient utilization of communication links for tactical data exchange within the battle force requires real-time knowledge of the connectivity between units and information on the architecture and state of the network that supports this connectivity (Lewis, 1987). This information is necessary to diagnose faults or reconfigure a net to achieve improved exchange of tactical data. Real-time information describing the state of a communications net is not usually available, comes at rates too fast to be assimilated by a human operator, and requires the opertor to be extremely knowledgeable in the protocol used by the network and in RF communications in general.

The Lockheed Expert System (LES) (Perkins and Laffey, 1984) has been designed to help knowledge engineers quickly solve problems in diagnosing, monitoring, planning, designing, checking, guiding, and interpreting. In its first application, LES was used to guide less-experienced maintenance personnel in the fault diagnosis of a large signal-switching network. LES uses not only the knowledge of the expert diagnostician (captured in the familiar form of 'IF-THEN' rules), but also knowledge about the structure, function, and casual relations of the device under study to perform rapid isolation of the module causing the failure. In addition to aiding the engineer in troubleshooting an electronic device, LES can also explain its reasoning and actions to the user, and can provide extensive database retrieval and graphics capabilities.

The list of other monitoring expert systems extended to ACE (Vesonder, *et al.*, 1983) which uses a database of maintenance records and is now installed in over 100 sites in the Regional Bell Operating Companies in the United States; NET/ADVISOR (Mantelman, 1986), for local area networks; CENTAURE (Benicourt and Crommelynck, 1989), for the telecommunications network of the French National Railways; DPN Monitor (Baird and White, 1989) for Bell Canada's packet switching network; British Telecom's AMF (Thandasseri, 1986) for TXE4 switches which are used by broadcast studios to route audio signals; RTS (Mantelman, 1986) for the diagnostic messages of the No. 1ESS switching system in the United States; and FIESTA (Miksell, *et al.*, 1987) for the satellite communications for the U.S. federal government. Other monitoring expert systems include TERESA (Callahan, 1988); COMPASS (Prerau, *et al.*, 1985) NEMESYS (Macleish *et al.*, 1987) and (Guattery and Villareal, 1985); PROPHET (Prerau *et al.*, 1986).

All the above systems are built according to the classical knowledge acquisition paradigm, where a knowledge engineer acquires the monitoring knowledge by interrogating the human expert and representing the knowledge in the form of rules. Such a representation allows for efficient chaining of the rules to simulate human expert reasoning but at the same time makes the management of the expert system itself very difficult. In particular, since the new rules may contradict the old rules or their consequences, it becomes very important to verify the consistency of the rule base after every update. Verification of the expert system, however, is not usually included in the expert system design. Additionally, the rule-based knowledge representation prevents taking the cost considerations into account when considering different reasoning paths. Thus, the optimization of the monitoring process is not being addressed by these systems.

2.2. TROUBLESHOOTING

The goal of troubleshooting is to rectify the malfunctioning system by replacing an unknown faulty component. Indicating a faulty component requires correct interpretation of all the available data. The trouble is explained by a malfunction model which can justify the observed data and suggest a replacement component. Malfunction modeling is often referred to as diagnosis. Diagnosis, in other words, is the process of interpreting the available data.

Resolution of the model bears important economic consequences as it is desired to execute the least expensive repair. Usually, however, the initial model resolution is unsatisfactory and it is necessary to seek out additional data to break down the malfunction model into several sub-models. Such data acquisition is performed by interacting with the malfunctioning system. The cost of the interaction contributes to the total troubleshooting task. The decision mechanism about which data to acquire, and how to do the acquisition constitutes the heart of any troubleshooting expert system.

Early troubleshooting expert systems were engineered to mimic the human troubleshooter by 'copying' the diagnostic and decision rules of the human domain expert. A prototype expert system (MAD) for maintenance of Northern Telecom's DMS-family digital switches is presented in Peacocke and Rabie (1988). MAD is based on using product documentation and experience obtained from craftspeople and field technical support to create the knowledge base for an expert system which diagnoses and chooses corrective actions for DMS-100 problems. Another example of incorporating the basic principles of diagnostic expert systems is NTC (Network Troubleshooting Consultant) (Hannan, 1987) system producing an effective network diagnostic tool.

Troubleshooting expert systems share the same shortcomings with the monitoring expert systems, i.e., the inability to optimize the troubleshooting process, and the inability to learn. Such expert systems generally require a knowledge engineer to maintain their knowledge bases during system operation.

Troubleshooter (Marques, 1988) is one of the first expert systems addressing the second shortcoming. Troubleshooter is a system for diagnosing problems in the Datakit Virtual Circuit Switch networking environment. Troubleshooter uses software tools from the academic laboratories, most notably OPS4 from Carnegie Mellon University and Franz LISP from the University of California at Berkeley. The Datakit Troubleshooter uses the same command interface as the human troubleshooters are and frequently resolves the trouble on its own initiative. To make the troubleshooting decisions, it keeps a historical database of component failures along with the evidence which was available during each troubleshooting session. New hypothesis are risen by applying Bayes' Rule to the frequency of failure records in its historical database. The historical database is updated immediately after the repair session, so that the future diagnosis may use the new evidence.

The Automatic Network Troubleshooter (ANT) (Zinky and Flechon, 1989) project's goal is to reduce federal network maintenance expense by reducing the mean-time-to-repair for performance problems. ANT uses a model-based troubleshooting technique: first creating an executable model of network misbehavior, then exploring it to identify the cause of performance problems. ANT uses pattern matching and the protocol hierachy to speed up the search for the bottleneck. Finally, ANT uses sensitivity analysis to determine the appropriate corrective action. The ANT prototype detects throughput bottlenecks caused by misconfigured parameters and lack of bandwidth.

Maintaining a large, private communications network is also a complex, expensive operation (Tieman, 1987). American Airlines is in the process of upgrading its communications network, AADN (American Airlines Data Network), which currently has over 110000 devices at over 11000 sites. One component of AADN is the Network Diagnostic Knowledge System (NDKS), an expert system which isolates and resolves problems detected on the network. Two logical components of NDKS cooperate to improve service to users on the network and to reduce operational and personnel costs related to maintaining the network. One of these components runs and interprets diagnostic tests to isolate faulty equipment when an anomaly is detected on the network. The other component directs the diagnostic procedures of the help desks to analyze problems reported by equipment users on the network. Both components share information which improves the overall performance of AADN.

Fault location in a malfunctioning data-link-protocol implementation is considered also in Barchanski, (1987) where the notion of protocol diagnosis is introduced. Another system (Brossier *et al.*, 1986) was developed to diagnose failures on private packet-switching networks, including rather simple problems of connection but also more difficult problems at the bit-frames and packet levels. A diagnosis expert system for the NEC NEAX61-series digital switching system is described in Koseki *et al.*, (1987). The system is founded on a multiple-knowledge-based approach, utilizing design knowledge represented by an abstract signal-flow model of the switching system, and domain-specific symptom knowledge and test knowledge. All kinds of knowledge are uniformly described in a network-based representation.

The list of troubleshooting expert systems further includes ARTEX (Fleischanderl *et al.*, 1989); Tektronix' network troubleshooter (Sayles and Thomas, 1988); DANTES (Van Cotthem *et al.*, 1987); ExT (Yudkin, 1987), (Yudkin, 1988); AUTOTEST-2 (Ackroff *et al.*, 1988; Fujimoto and Minato, 1988; Tarouco, 1987).

3. Troubleshooting Knowledge Acquisition

In this section, we develop a network maintenance expert system design aid. Such an aid automatically constructs the troubleshooting computer programs which can be later used by the repair technicians. The system is able to develop a strategy of its own given the description of the network and its behavior. The interested reader is referred to Tzafestas (1988) for a special chapter on knowledge-based communication network diagnosis.

Let a network $P = P(S, B)$ be modeled by its structure S and its behavior B. Structure $S = S(F, \Gamma)$ is defined by the set F of components and topology Γ on the set F. Examples of components in a network are voice/data multiplexors, terminal interfaces, switching modules and power subsystems. These components are connected in sequential and parallel manners in forming a network. Behavior B is modeled by the *a priori* probability p_x of failure associated with every component $x \in F$. We want to design a procedure which is responsible for solving the following:

Problem 1 (*Diagnostic Problem*): Given a multicomponent plant P and a symptom indicating an anomaly on the expected input/output behavior, identify the subset $F_f \subset F$ of faulty components.

However, the identification part of the above problem is itself a complex process. In particular it involves a repetitive generation of hypotheses and their support or rejection. Proving or rejecting a hypothesis requires in turn performing sequences of measurements and subsequent sequences of symbolic computations. Addtionally, each measurement t_i involves, certain costs c_i, which can be the use of an expensive piece of test equipment, the duration of the measurement causing disruption of network services, and/or possible degradation of network reliability due to stressful tests. One can compute the total cost of a measurement sequence by adding the respective costs. Therefore, the identification requirement of the Problem 1 can be satisfied by recursively solving the following two problems.

Problem 2 (*Classification Problem*): Given a system and the results of previously performed measurements, obtain an estimate of the faulty subset F_f (generate hypothesis).

Problem 3 (*Planning Problem*): Given an estimate of F_f, obtain the minimum expected cost plan (sequence) of measurements, so that when performed, the initial hypothesis will be either proved or rejected. The recursion is continued until a provable hypothesis is generated.

Problem 2 is solved trivially by applying a single *Classification Rule*:

> The ambiguity set (the set of suspicious components) is the largest subnetwork having all good inputs and all bad outputs. This rule is sometimes called 'the upstream indictment rule'.

The test planning problem on the other hand is known to be NP-hard (Hyafil and Rivest, 1976). Usually the planning problem is solved by applying an enumerative method. Enumerative methods solve a discrete optimization problem by recursively splitting down the feasible set into smaller subsets, computing bounds on the value of the objective function, and eliminating some subsets from further consideration on the basis of these bounds. The splitting procedure stops when every subset has been shown to contain a less valuable solution than the one already in hand.

Our approach for the test planning problem solution is a variation of the enumerative method. The key idea of our approach is to solve the test sequencing problem always for the purely sequential or parallel case. Such an approach allows for direct computation of the expected cost of the test sequence, thus allowing for its explicit minimization. Analysis of such idealized cases is eased in the telecommunication networks domain because of the obvious causality relation imposed on the network components in the sequential case and because of the explicit lack of such a relation in the parallel case. Knowledge-based post processing based on the causality relation of the idealized sequential and parallel cases enables us to construct a good test plan for the general case. The reader interested in a test sequencing procedure, which does not rely on such a causality assumption, is referred to Pattipati and Dontamsetty (1989).

3.1. PURELY SEQUENTIAL CONNECTIONS

In the sequel, we assume that each component is a single-input single-output (SISO) Least Replaceable Unit (LRU). This constraint is not a general restriction, since any component having more inputs or outputs can be logically substituted with the corresponding number of SISO logical LRU's and the indictment of the original component can be implied by indictment of any one of its SISO subcomponents.

DEFINITION 1. A multicomponent network $P(S,B)$ is called *completely 0-1 observable* if there exists a test $t_i \in T$ for every link $j_i \in \Gamma$ with only two possible outcomes, PASS or FAIL.

DEFINITION 2. System $P(S, B)$ is said to be *fauly* if there exists a link j_i for which test t_i has observed the outcome FAIL. The corresponding test t_i is called *symptom.*

We assume that every component $x \in F$ has an associated *a priori* failure probability p_x. Also we assume that every test $t \in T$ has an associated cost $c(t)$ which must be paid in order to observe the outcome of the test. And finally, we assume that there is only one defective component (single-fault assumption).

Problem 3 (*Test Planning Problem*): Given a faulty, completely 0-1 observable multicomponent network P with the test set T, find the test sequence $t_1 \ldots t_n$ of minimal expect cost which is able to discriminate the defective component.

Let us associate a circular node and two outcoming double links with every test t and its two possible outcomes. Note that each outcome defines a partition in the

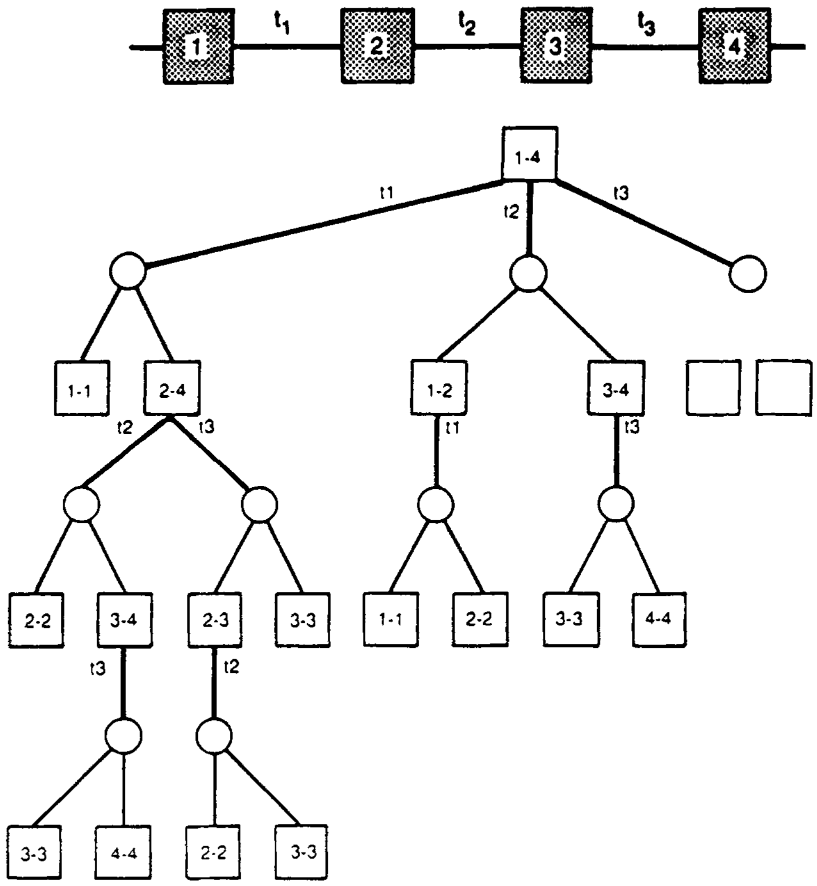

Fig. 1. Test tree for the sequential network.

dynamic system P of the components which are under suspicion of being defective (*ambiguity set*) and the rest. Let us associated a square node with each ambiguity set.

A choice set is defined by the set of all tests that may be performed to the given ambiguity set.

Let us associate a single link with every element in the choice set. Note that when the cardinality of the ambiguity set is 1 then the cardinality of the corresponding choice set is 0.

DEFINITION 3. A test tree is obtained by attaching the circular nodes to the single links (a tree of ambiguity sets and choice sets) (Figure 1).

DEFINITIION 4. A test path is defined by the starting and the terminal ambiguity sets s and t and it consists of all the tests t_i that are required to reach t from s. Cost of test path is defined by

$$c(s, t) = \sum_{\tau \in \Gamma^{-1} F_i} c(\tau),$$

where

$$\hat{\Gamma}^{-1} = \Gamma^{-1}t \cup \Gamma^{-2}t \cup \ldots,$$

$\Gamma^{-1}t$ is the set of tests leading to t,

$X^{-1}(\Gamma^{-1}t)$ is the ambiguity set that generated $\Gamma^{-1}t$,

$\Gamma^{-2}t = \Gamma^{-1}(X^{-1}(\Gamma^{-1}t))$, and so on.

DEFINITION 5. A spanning test tree is the subtree of the test tree which has all the components X of the plant as its leaves appearing only one. Cost of the spanning test tree is defined by

$$c(\Gamma) = \sum_{x \in F} c(s, x) p_x, \qquad (1)$$

where s is the initial ambiguity set.

Note: Given a symptom, a spanning test tree defines a test sequence which is able to discriminate the defective component causing the symptom. The cost of the spanning test tree is the expected cost of the test sequence. Thus we may restate the Test Planning Problem in the following way.

Problem 3″ (*Test Planning Problem*): Given a test tree find its minimal spanning test tree.

3.1.1. *Main Equation*

Let the test tree Γ consist of the root (initial state), set of initially possible tests $t_1, \ldots, t_n$ and set of resulting subtrees $\Gamma_{i_1}, \Gamma_{i_2}$, $i = 1, \ldots, n$, corresponding to the two possible outcomes of each of the tests.

Let $c(\Gamma)$ stand for the cost of traversing the test tree Γ in a particular path, $c^*(\Gamma)$ be the minimal cost, $c(t_i)$ represent the cost of performing test t_i, and P_i and Q_i be the corresponding probabilities of obtaining the test trees Γ_{i_1} and Γ_{i_2} according to the result of test t_i (FAIL or PASS).

Then, using the principle of optimality (i.e. the optimal test tree consists of optimal test subtrees), we obtain

$$c^*(\Gamma) = \min_{i=1 \cdots n} \{c(t_i) + P_i c^*(\Gamma_{i_1}) + Q_i c^*(\Gamma_{i_2})\} \qquad (2)$$

and

$$P_i = \sum_{j \in X^{-1}(\Gamma_{i_1})} p_{ij}, \quad Q_i = 1 - P_i,$$

$$\sum_{i=1}^{n} p_{ij} = 1,$$

where p_{ij} is the probability of failure of component j belong to subtree Γ_{i_1}.

Such a tree allows for natural representation in the terms of AND/OR tree. Every node in the AND/OR tree is either an AND node or an OR node. An AND node has as its cost a sum of the costs of its subtrees and therefore corresponds to the sum

$$c_i = c(t_i) + P_i c^*(\Gamma_{i_1}) + Q_i c^*(\Gamma_{i_2}).$$

An OR node has its cost computed from the relation

$$c^*(\Gamma) = \min_{i=1\cdots n} c_i,$$

i.e., it is the minimal of the costs of its subtrees. Standard artificial intelligence algorithms to search AND/OR trees can be now applied (Bagchi and Mahanti, 1983). The most powerful AND/OR tree search algorithm is the AO^* algorithm which uses a heuristic cost to speed up the search.

Let us define heuristics h_{i_1} and h_{i_2} representing the underestimates of the costs of the trees Γ_{i_1} and Γ_{i_2}, respectively. Then the lower bound of the cost of selecting test t_i can be redefined as

$$c(t_i) = c(t_i) + p_i h_{i_1} + q_i h_{i_2}, \qquad (3)$$

where $p_i + q_i = 1$.

Then the estimate of the cost of the minimal test tree is obtained from

$$c_n = \min_{i=1\cdots n} \{c(t_i)\}$$

and the best test is

$$t^* = \arg \min_{i=1\ldots n} \{c(t_i)\}. \qquad (4)$$

We utilize a result from information theory for the heuristic h computation. Let n_x be the number of test steps required to indict component x, and $c_{\min}$ be the minimum of $c(t)$ for all $t \in T$. Then from Equation (1) we have

$$c(\Gamma) \geqslant c_{\min} \sum_{x \in X} n_x p_x, \quad c_{\min} \bar{L} = h, \qquad (5)$$

where $\bar{L}$ is the expected code length for the binary Huffman algorithm which is known to be optimal (Huffman, 1962). By saving the partial cost estimates during the computation, one achieves the dynamic programming advantage of the formula (2). By developing only the nodes of the tree which satisfy (4), and pruning away the nodes where the cost estimate c_n already exceeds the current minimal cost, one achieves the advantage of branch-and-bound approach. And, finally, since the optimal test tree, produced by Equation (5), is computed over a larger set of networks, h is an admissible heuristic, i.e., h produces an underestimate of the cost of the optimal test tree. Thus we obtain an AO^* algorithm for the best test sequence selection.

Table I. Test tree algorithm performance in the purely sequential case

Number of components	4	9	11	13	15	17	19	21	23	25
Branch and bound without heuristic	3	26	36	43	51	144	66	121	88	117
*AO**	6	16	20	30	30	79	63	90	72	95

3.1.2. *Performance in the Purely Sequential Case*

Table I summarizes the number of nodes expanded by the test selection algorithm in the purely sequential connection case.

The failure probabilities and the cost of tests (we assume the output of every component to be observable) are summarized in Table II.

3.2. PURELY PARALLEL CONNECTIONS

Given a completely 0-1 observable sequential network, one may notice that there are two tests which have special significance among the rest of the available tests. These

Table II. Failure probabilities of single components and their associated costs of testing

Component number	Failure probability	Component output measurement cost ($)
1	0.01	2
2	0.02	1
3	0.01	3
4	0.04	2
5	0.02	2
6	0.01	1
7	0.02	3
8	0.01	2
9	0.04	1
10	0.02	4
11	0.01	2
12	0.02	1
13	0.01	3
14	0.04	2
15	0.02	2
16	0.01	1
17	0.02	3
18	0.01	2
19	0.04	1
20	0.02	4
21	0.01	2
22	0.02	1
23	0.01	3
24	0.04	2
25	0.02	8

tests may indict a component or define a new ambiguity set consisting of the entire previous network without that component. These tests are measuring either the output of the first component on the signal path or the input to the last component. The discussed component is then either the first or the last component respectively. We will call these special tests as *terminal tests*.

The main equation (2) will define a terminal test as the last test only in the special cases such as

(1) sequential network consisting only of two or three components;
(2) uneven (skewed) failure rate distribution of the components;
(3) uneven test cost distribution;
(4) combination of the above cases.

In general, however, the best test will always be chosen as nonterminal test. In particular, when all the failure probabilities are equal and the costs of the tests are equal, the main equation (2) will reduce to the familiar bisection method, producing a balanced test tree. So we may claim that a variation of a 'group testing' methodology is implemented via utilization of the main equation (2): a test will either indict or clear of blame an entire subset of the sequential network. This group testing is facilitated in the sequential network because a fault is propagated from the defective component downstream via the network (the signal path).

Unfortunately, such group testing is not applicable in the case of purely parallel networks. The reason is purely economical: it is too time-consuming to construct artificially sequential or parallel subnetwork on the given parallel network. Thus, we are forced to apply the 'terminal test' methodology at every step. Now, however, any component belonging to the parallel network may play the role of the first (or the last) component of the sequential network. The problem of optimal ordering of the test sequence is thus equivalent to the problem of optimally eliminating components one-by-one from the initial ambiguity set consisting of the entire purely parallel network.

In the purely parallel case the main equation (2) is reduced to

$$c^*(\Gamma) = \min_{i=1\cdots n} \{c(t_i) + q_i c^*(\Gamma_i)\} \qquad (6)$$

As in the sequential case, an A^* algorithm can be constructed in order to build the minimal spanning test tree for a given parallel network.

We note, however, that a less than quadratic time performance algorithm can be constructed if we just order increasingly the ratios p_x/c and choose the next test accordingly. This algorithm provides the minimum expected cost diagnostic tree.

3.3. HYBRID SEQUENTIAL AND PARALLEL CONNECTIONS

As mentioned at the beginning of this section, the key idea of our appraoch is to carry out the test planning always at either purely sequential or purely parallel level. A small

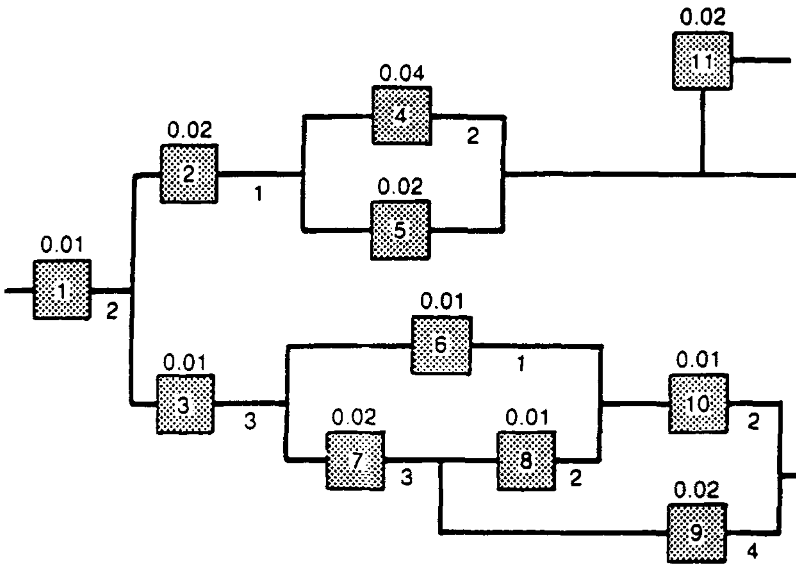

Fig. 2. Hybrid network.

rule-based knowledge system is constructed so as to provide our system with the ability to *make abstractions* of the given network.

Using this knowledge system, a general pattern (either sequential or parallel) is asserted intially. Then the components that do not fit into the chosen general pattern, are aggregated into the 'super component' level. This new abstract network is registered in the memory (asserted) and passed to the test planning algorithm. It is convenient to represent every abstracted network as a node of a variation of the AND/OR graph. We differentiate between *par* nodes representing the parallel abstractions and *seq* nodes, representing the sequential abstractions. The rules governing the abstraction process are constructed in such a way that every component or subsystem is represented only once in the *par/seq* graph. The process of the *par/seq* graph construction is continued until every node consists of no abstractions. The action of the test planning algorithm is deferred, however, until the cost of testing the system component is known. This process of network abstraction and test tree computation is repeated until a test tree for each abstraction level is generated. The total test tree is obtained then jut be merging the computed test trees when exiting the recursion. An example of the test tree (Figure 3) generated by the algorithm for a hybrid parallel and sequential network (Figure 2) follows. The expected cost is $2.79.

4. Conclusions

We survey the designs of over 40 automatic diagnostic and repair expert systems for telecommunications networks in terms of their structures, knowledge representations, inference mechanisms, and interfaces with developers and end-users. The survey reveals that the main shortcoming of the network maintenance expert systems is the

```
measure input to 11;
if passed then check 11
else begin
    measure output from 1;
    if failed then check 1
    else begin (* troubleshooting $S_1$, which consists of $S_2$ and $S_3$ *)
               (*                                               in parallel *)
               (* first is $S_2$ which consists of 2 and $S_4$ in sequence *)
        measure output of 4 and 5;
        if failed then begin
                    measure output of 2;
                    if failed then check 2
                    else begin (* $S_4$ consists of 4 and 5 in parallel *)
                        measure output of 5;
                        if failed then check 5
                        else check 4
                    end
        end
        else begin (* next is $S_3$ *)
          measure output of 7;
          if failed then begin
                      measure output of 3;
                      if failed then check 3
                      else check 7
          end
          else begin
             measure output of 10;
             if failed then begin
                        measure input to 10;
                        if failed then begin
                                    measure output of 6;
                                    if failed then check 6
                                    else check 8
                        end
                        else check 10
             end
             else check 9
          end
        end
    end
end
```

Fig. 3. Test tree.

difficulty to acquire the troubleshooting knowledge both when intializing the expert system and after its deployment. Additionally, the state-of-the-art troubleshooting expert sysems do not optimize troubleshooting cost. We present an AO^* algorithm to generate a network troubleshooting expert system which minimizes the expected troubleshooting cost and learns new troubleshooting techniques during its operation. The algorithm requires three kinds of input: network topology, component failure rate, and relative costs of tests. It uses a modification of Huffman code which we prove to be an admissible heuristic in the AO^* search. We implemented our algorithm in PROLOG. The prototype system demonstrated encouraging results on several problems of industrial significance.

References

Ackroff, J., Surko, P.T. and Wright, J.R., 1988, Auto Test-2: An expert system for special services, *Proc. Fourth Annual Artificial Intelligence and Advanced Computer Technology Conference*, Long Beach, CA, 503–508.

Baird, C. and White, T., 1989, A real time network monitor. *Proc. Conf. Artificial Intelligence, Telecommunications, and Computer Systems*, Avignon, France, pp. 35–41.

Bagchi, A. and Mahanti, A., 1983, Admissible heuristic search in AND/OR graphs, *Theore. Comput. Sci.* **24**, 207–219.

Barchanski, J.A., 1987, *Expert Systems for Data Link Protocol Diagnosis*, IEEE, New York.

Benicourt, A. and Crommelynck, V., 1989, CENTAURE: Le sysems expert de surveillance due resau national de teleinformatique de la SNCF, *Proc. Conf. Artificial Intelligence, Telecommunications, and Computer Systems*, Avignon, France, 17–34.

Bernstein, S., 1987, DesigNet: An intelligent system for network design and modelling, *ICC'87*, Seattle, Washington.

Brossier, J.Y., Faller, B., Fron, A., Garcia, G. and Kirsch, P., 1986, Communication network troubleshooting expert system, *Internat. Conf. Communications*, Washington, D.C. Conference Record IEEE-86CH2314, pp. 1238–1240.

Callahan, P., 1988, Expert Systems for AT&T Switched Network Maintenance. *AT&T Technical J.* **67**(1), 93–103.

Chester, M., 1986, Expert systems help to shape and control data networks, *Electronic Products* (*Garden City, New York*), **29**(12), 15–17.

Corn, P.A., Dube, R., McMichael, F. and Tsay, J.L., 1988, An autonomous distributed expert system for switched network maintenance, *Globecom '88*, IEEE Global Telecommunication Conference and Exhibition Communications for the Information Age.

Cynar, L., Mueller, D. and Paroczai, A., 1986, Computers design networks by imitating the experts. *I, Data Commun.* **15**(4), 137–45.

Ferrara, F., Giovannini, F. and Paschetta, E., 1989, An expert system for on-line monitoring of a data network, *Csel. Tech. Rep.* (*Italy*) **17**(2), 179–83.

Ferrara, F., Giovannini, F. and Paschetta, E. 1989, IAS: An expert system for packet-switched network monitoring and repair assistance. *Proc. Conf. Artificial Intelligence, Telecommunications, and Computer Systems*, Avignon, France, pp. 185–197.

Fleischanderl, G., Friedrich, G. and Retti, J., 1989, Model-driven fault localization in audio routing systems, *Proc. Conf. Artificial Intelligence, Telecommunications, and Computer Systems*, Avignon, France, pp. 173–183.

Fujimoto, K. and Minato, Y., 1988, Intelligent network operation systems for INS network, *IEEE Internat. Conf. Communications '88: Digital Technology Spanning the Universe*, Conference Record (88CH2538).

Guattery, S. and Villareal, F., 1985, NEMESYS: An expert system for fighting congestion in the long distance network, *IEEE Symp. Expert Systems in Government* (October), Washington, D.C.

Hannan, J.J., 1987, Network solutions employing expert systems, *Sixth Internat. Phoenix Conf. Computers and Communications*, pp. 543–547.

Huffman, D.A., 1962, A method for the construction of minimum redundancy codes, *Proc. IRE*, **40**, 1098–1101.

Hyafil, L. and Rivest, R., 1976, Constructing optimal binary decision trees is NP-complete, *Information Processing Lett.* **51**(1), 15–17.

Karsai, G., Biegl, C., Padalkar, S., Sztipanovits, J., Kawamura, K., Miyasaka, N. and Inui, M., 1988, Knowledge-based approach to real-time supervisory control. *Proc. 1988 Americ. Control Conference*, **1**, pp. 620–625.

Katsuyama, Y., 1987, Expert system for digital transmission network troubleshooting, *ICC'87* (June).

Kaye, A.R. and Pagurek, B., 1989, An expert monitor for a data circuit switch in a wide-area network, IEEE Infocom '89 The Conference on Computer Communications, *Proc. 8th Joint Conf. IEEE Computer and Communications Societies. Technology: Emerging or Converging?*, IEEE, New York.

Khan, N.A. and Dube, R., 1987, The GEMS-trunk trouble analyzer: A knowledge-based expert system for trunk maintenance. IEEE Infocom '87. The Conference on Computer Communications, *Proc. 6th Conf. Global Networks: Concept to Realization*, IEEE, New York.

Koseki, Y., Wada, S.I., Nishida, T. and Mori, H., 1987, *SHOOTX: A Multiple Knowledge Based Diagnosis Expert System for NEAX61 ESS*, IEEE, New York.

Lewis, D.D., 1987, Architecture of a real-time expert system for data link monitoring and management, *Naval Engineers J.*, **99**(3), 90–93.

Macleish, K.J., Thiedke, S. and Vennergrund, D., 1987, Application of artificial intelligence in telecomunications, *IEEE/IEICE Global Telecommunications Conf.* Ohmsha Ltd, Tokyo.

Mantelman, L., 1986, AI carves inroads: Network design, testing and management, *Data Commun.* (July): 106–123.

Marques, T.E., 1988, Symptom-driven expert system for isolating and correcting network faults, *IEEE Commun. Magazine* **26**(3), 6–13.

Miksell, S., Quillin, R. and Lowe, D., 1987, *Network Fault Diagnosis: Knowledge Representation Using Parallel Reasoning*, IEEE, New York.

Pattipati, K. and Dontamsetty, M., 1989, Test Sequencing in Modular Systems, *IEEE Internat. Conf. Systems, Man and Cybernetics*, Cambridge, MA.

Peacocke, D. and Rabie, S., 1988, Knowledge-based maintenance in networks. *IEEE J. Selected Areas in Commun.* **6**(5), 813–818.

Perkins, W.A. and Laffey, T.J., 1984, Les: a general expert system and its applications, *Proc. SPIE The International Society for Optical Engineering*. 485, pp. 46–57.

Prerau, D., Gunderson, S. and Levin, P., 1986, The PROPHET expert system: Pro-active maintenance of telephone company outside plant, *Proc. Conf. Intelligence, Communication, and Computer Systems*, Avignon, France, pp. 185–197.

Prerau, D., Gunderson, S., Reinke, R. and Goyal, S., 1985, The COMPASS expert system: Verification, technology transfer, and expansion, *2nd Conf. Artificial Intelligence Applications*, Washington, D.C. pp. 597–602.

Sayles, W. and Thomas, J., 1988, Finding and fixing network faults with an expert system, *Data Commun.* **17**(6), 149–150.

Tarouco, L., 1987, Network management *IBERICOM '87: 1st Iberian Conf. Data Communications*. 1:2 Vol. p. 657.

Thandasseri, M., 1986, Expert systems for TXE4A exchanges, *Electrical Commun.* **60**(2).

Tieman, L.R., 1987, American Airline's network diagnostic knowledge system, *Proc. 3rd Artificial Intelligence and Advanced Computer Technology Conference*, pp. 702–706.

Tzafestas, G. (ed.), (1988), Knowledge Based System Diagnosis, Supervsion and Control, Plenum Press, New York.

Van Cotthem, H., Mathonet, R. and Van Ryckeghem, L., 1987, DANTES: An expert system shell dedicated to real-time network troubleshooting, *ICC '87*, June, Seattle, Washington.

Vesonder, G., Stolfo, S., Zielinski, J., Miller, F. and Copp, D., 1983, ACE: An expert system for telephone cable maintenance, *IJCAI '83* **8**, 116–120.

Wright, J. and Vesonder, G., 1990, Expert systems in telecommunications, *Expert Systems with Applications* **1**(2), 127–136.

Yudkin, R.O., 1987, On testing communication networks, *GLOBECOM Tokyo 1987 IEEE/IECE Global Telecommunications Conference*.

Yudkin, R.O., 1988, On testing communications networks, *IEEE J. Selected Areas in Communications* **6**(5), 805–812.

Zinky, J.A. and Flechon, J.-L., 1989, An automatic network troubleshooter for throughput bottlenecks in computer networks, *Proc. Annual AI Systems in Government Conference* pp. 296–302.