Fusing Hard and Soft Computing for Fault Management in Telecommunications Systems

Roy Sterritt, *Member, IEEE*, and David W. Bustard, *Member, IEEE*

***Abstract*—Global telecommunication systems are at the heart of the Internet revolution. To support Internet traffic they have built-in redundancy to ensure robustness and quality of service. This requires complex fault management. The traditional *hard* approach is to reduce the number of alarm events (symptoms) presented to the operating engineer through monitoring, filtering and masking. The goal of the *soft* approach is to automate the analysis fully so that the underlying fault is determined from the evidence available and presented to the engineer. This paper describes progress toward automated fault identification through a fusion between these soft and hard computing approaches.**

***Index Terms*—Fault identification, fault management, genetic algorithms, hard computing, probabilistic reasoning/networks, soft computing, telecommunication systems.**

I. Introduction

SOFT computing differs from *hard* (conventional) *computing* in that it is tolerant of imprecision, uncertainty and partial truth (Lotfi Zadeh) [1]. In general, soft computing aims to exploit these properties to achieve tractability, robustness and low cost in computing systems.

Currently, soft computing primarily comprises *neural network theory*, *fuzzy logic*, *probabilistic reasoning* (incorporating belief networks), *genetic algorithms*, *chaos theory*, and sections of *learning theory*. It is an expanding field however and a consensus as to the exact scope of the subject has not been reached [2].

Soft computing offers a toolbox of techniques that can be used in combination to address problems in the computing domain [3]. With this perspective, the task is to find the complementary techniques that best fit each specific application [4].

Hard computing techniques [5] are typically easier to apply. Also the stability of resulting solutions is highly predictable and the computational burden of practical algorithms is typically low [6]. Just as soft computing brought together related techniques for greater benefit [7], taking an even broader perspective can bring further gain [8]. The purpose of this paper is to consider the combined use of hard and soft computing techniques in the domain of fault management in telecommunication systems. Definitions of the necessary technical terms and acronyms used in the paper are summarized in Table I. Alarm events used in illustrative examples are summarized in Table IV.

Manuscript received March 14, 2001; revised June 18, 2002. This work was supported by the Centre for Software Process Technologies (CSPT), funded by Invest NI through the Centres of Excellence Programme, and from collaborative work with Nortel Networks on the Jigsaw project, funded though the IRTU Start Programme (Start-ITS187). This work was presented in part at the International Symposium on Soft Computing and Intelligent Systems for Industry, Paisley, U.K., June 26–29, 2001.

The authors are with the School of Information and Software Engineering, Faculty of Informatics, University of Ulster, U.K. (e-mail: r.sterritt@ulster.ac.uk; dw.bustard@ulster.ac.uk).

Digital Object Identifier 10.1109/TSMCC.2002.801725.

TABLE I
FAULT MANAGEMENT NOMENCLATURE

Term	Explanation
Alarm consequences	The consequence of an alarm may be to trigger one or more other alarms, which can lead to a cascade of error reports.
Alarm correlation	This is the conceptual interpretation of multiple alarms, giving them a meaning and notionally replacing them by a single composite alarm.
Element Controller (EC)	This is the network element manager. A network element can know about its own health, internal connectivity, and state of signals on its interfaces but not the network elements to which its connected.
Fault	This is a malfunction that has occurred in either the hardware or software of the network.
Event	An occurrence or incident on the network.
(Alarm) Filtering	This prevents alarms being reported as active unless they exist for a defined period of time.
ITU-T	International Telecommunications Union - Telecommunications standardization body, previously CCITT (Consultative Committee on International Telegraphy and Telephony).
(Alarm) Masking	This inhibits reports from alarms if their superior alarms are active. The masking hierarchy defines this relationship.
(Alarm) Monitoring	Alarms may be switched on or off. If they are on they are being monitored.
Network Element (NE)	A piece of telecommunications equipment that provides support or services to the user.
PDH	Plesiochronous Digital Hierarchy: the first adaptation of analog voice information to the digital model of transport. This legacy approach is catered for within the current SDH standard.
SDH / SONET	SONET and SDH are a set of related standards for synchronous data transmission over fiber optic networks. SONET is short for Synchronous Optical NETwork and SDH is an acronym for Synchronous Digital Hierarchy. SONET is the United States version of the standard published by the American National Standards Institutue (ANSI). SDH is the international version of the standard published by the International Telecommunications Union (ITU).
STM	Synchronous Transfer Mode, rate of transfer.
Switch protection	In a protected network, traffic can be switched to another route if parts of the network fail.
TN-1X	Nortel Networks' product name for one of its SDH multiplexers.
User actions	A user initiated command to a network element.
User logins	A user initiated connection to a network element.

TABLE II
AN EXAMPLE SET OF ALARM EVENTS

Type	Path	Event_Type	EC_Time	EC_Date	Alarm	Sev	NE_ID	Alarm_ID
TN-1X	/bireh706/TN-1X/Mux_03	Comms fail alarm	11:27:12	26 March 1999	present	Critical	5002	25001
TN-1X	/bireh706/TN-1X/Mux_03	Comms fail alarm	11:33:41	26 March 1999	clear	Critical	5002	25001

TABLE III
EXAMPLE OF OTHER ALARMS GENERATED FROM COMMS FAIL ALARM

Type	Path	Event_Type	EC_Time	EC_Date	Alarm	Sev	NE_ID	Alarm_ID
TN-1X	/bireh706/TN-1X/Mux_02/S6	Qecc-Comms_Fail	11:31:48	26 March 1999	present	Critical	5001	668
TN-1X	/bireh706/TN-1X/Mux_04/S7	Qecc-Comms_Fail	11:32:38	26 March 1999	present	Critical	5003	349
TN-1X	/bireh706/TN-1X/Mux_04/S7	Qecc-Comms_Fail	11:33:05	26 March 1999	clear	Critical	5003	349
TN-1X	/bireh706/TN-1X/Mux_02/S6	Qecc-Comms_Fail	11:33:31	26 March 1999	clear	Critical	5001	668

II. OVERVIEW OF TELECOMMUNICATIONS SYSTEMS AND FAULT MANAGEMENT

High-speed broadband telecommunication systems are built with complex management systems and extensive redundancy to ensure robustness. The presence of a fault may be detected by the offending network component, or its parent, and the consequences of that fault may also be reported by other network elements. This often results in a large number of alarm events being raised and cascaded to the network Element Controller.

The behavior of the alarms is so complex it appears nondeterministic [9], making it very difficult to isolate the true cause of the fault [10]. Failures in the network are unavoidable but quick detection and identification of faults is essential to ensure robustness. To this end the ability to *correlate* alarm events, identifying those that are related, is very important [11].

The major telecommunication equipment manufacturers deal with alarm correlation through alarm monitoring, filtering and masking as specified by ITU-T [12] and other standards bodies. Resulting rule-based diagnostic systems provide assistance to the operator whose expertise is then required to determine the underlying fault (or faults) from the correlated set of alarms reported.

There are two real world concerns in handling faults:

1) coping with the sheer volume of alarm event traffic resulting from a fault;
2) locating the cause of the fault quickly and efficiently from these symptoms [13].

Alarms monitoring, filtering and masking reduce the number of events to help address the first concern. The skill of the operator is central to addressing the second.

This description implies that fault detection is a two-stage process but that need not be the case. Artificial intelligence (AI) techniques can be applied directly to fault symptoms to help locate the source of faults. Telecommunication manufacturers have been reluctant to make full use of AI techniques, however, especially those with an "uncertainty" element. Rule-based approaches dominate, largely because the conclusions drawn are deterministic, and thus can be traced and understood by domain experts. Rule-based approaches, however, cannot easily handle uncertainty and pose substantial maintenance problems [14]. Techniques to assist in the discovery and development of rules in heterogeneous network environments are also essential.

This paper looks at techniques across soft and hard computing that might be used beneficially in fault handling. What matters most is improving effectiveness and at the very least the softer techniques can be offered as additional support to more traditional approaches to fault detection.

III. INDUSTRIAL APPLICATION OF FAULT MANAGEMENT

A *fault* is a malfunction that has occurred in either the hardware or software of the network. This can be due to some external event such as, for example, a fiber cable being severed, or an internal fault, such as a component failure.

An *event* is an occurrence on the network. Events that relate to the management of the network are recorded by the Element Controller. In older releases each recorded event was an alarm. This is no longer the case; other examples of events include *user logins* and *user actions* such as *switch protection*.

There are numerous types of alarm event that may be generated within a network element. For instance, a Nortel Networks FiberWorld multiplexer (TN-1X, release 8) has 88 possible alarm events. Other releases and other products have different possibilities. An example of a critical alarm is *Comms fail*—an alarm raised by the Element Controller if it cannot maintain a communications channel to another network element.

An alarm exists for a period of time so under normal circumstances an *alarm present* event will be accompanied by an *alarm clear*. In the event records shown in Table II, for example, the *Comms fail* alarm exists for six minutes and 29 seconds.

Each alarm type is assigned a severity level of *critical*, *major* or *minor* by the network management system, depending on the severity of the event indicated by the alarm type. In the example, the alarm type *Comms fail* has a critical severity level.

The instance of a fault can cause several alarm events to be raised from an individual network element. Also a fault may trigger numerous secondary alarms in other network elements across the network. For example, the *Comms fail* alarm may trigger an embedded control channel communications alarm (*Qecc-Comms_fail*), or even laser alarms, depending on the fault and configuration. In the *Comms fail* alarm example (Table II),

TABLE IV
ALARM EXPLANATIONS

Alarm Type	Explanation
AIS	Alarm Indication Signal
AU-AIS	Administrative Unit - Alarm Indication Signal An AIS has been detected in the H1 and H2 bytes indicating a failure in the upstream path.
Comms fail	The comms fail alarm is raised by the TN-MS management system if the TN-MS can not maintain a communication channel to the indicated NE.
INT-AU-AIS	Internal-Administrative Unit-Alarm Indication Signal An internal AIS has been detected in the H1 and H2 pointer bytes of the AU for three consecutive frames.
INT-TU-AIS	Internal-Tributary Unit-Alarm Indication Signal An AIS has been detected internally in the pointer bytes of the TU.
LP-PLM	Low-order Path - Path Label Mismatch The value of the signal label bits in the V5 byte does not correspond with expectations.
MS-AIS	Multiplex Section - Alarm Indication Signal MS-AIS indicates an AIS has been detected in the K2 byte in the section overhead, indicating a failure at the far multiplexer
MS-FERF	Multiplex Section - Far End Receive Failure A Far End Receive Failure indication has been detectedin the K2 byte of the section overhead.
NE-Card_Fail	The subrack controller has failed to receive three consecutive 'heart beat' messages on the internal control bus from a traffic card.
PPI-AIS	PDH Physical Interface - Alarm Indication Signal (AIS) has been detected in the incoming 2Mbit/s traffic
PPI-Unexpl_Signal	PDH Physical Interface - Unexplained Signal A 2Mbit/s or 34 Mbit/s signal has been detected on a tributary which is configured not to expect traffic (no connection)
Qecc-Comms_fail	The NE cannot communicate via the Embedded Control Channel (ECC) of the indicated STM-N card with the neighbouring NE
TU-AIS	Tributary Unit-Alarm Indication Signal An AIS has been detected in the pointer bytes of the TU.

other *Qecc-Comms_fail* alarms were raised (Table III), indicating that the network element cannot communicate with its neighbors.

A. Alarm Monitoring, Filtering and Masking

Alarms can grow exponentially in the network under certain fault conditions; the larger the network the greater the number of alarms produced. It is therefore essential that the network elements provide some correlation of these alarms so that the Element Controller, or a higher level network manager, is not overwhelmed and that only those with high priorities are transmitted.

This is handled in three sequential transformations: *alarm monitoring*, *alarm filtering* and *alarm masking*, as summarized schematically in Fig. 1.

Alarm monitoring takes the raw state of an alarm and produces a *monitored* state. Alarm monitoring is enabled/disabled for each alarm instance. If monitoring is enabled, then the monitored state is the same as the raw state. If monitoring is disabled then the monitored state is *clear*—that is the alarm is suppressed.

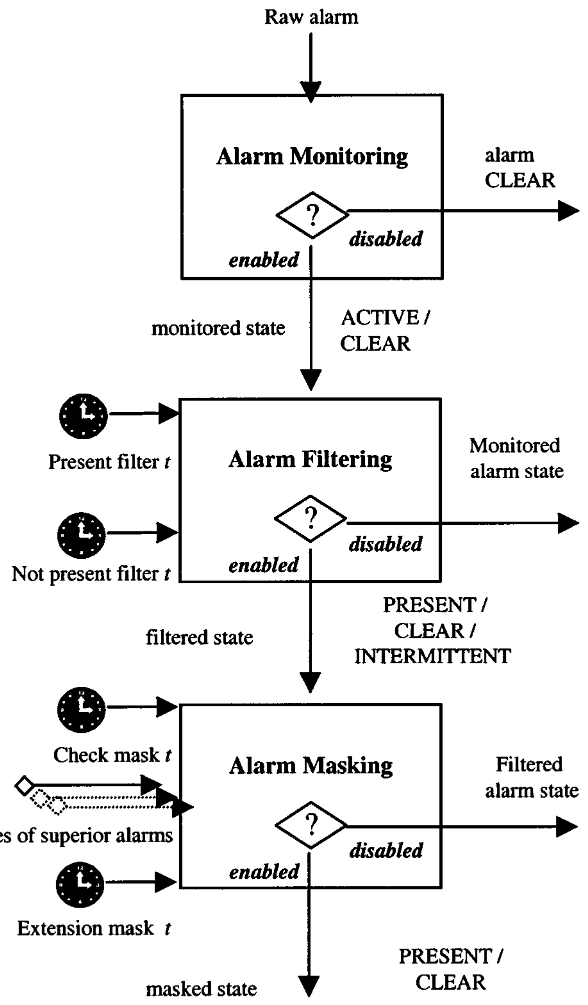


Fig. 1. Alarm monitoring, filtering and masking sequential transformations.

Alarm filtering is also enabled/disabled for each alarm instance. An alarm may exist in any one of three states: *present*, *intermittent* or *clear*, depending on how long the alarm is raised. Assigning these states, by checking for the presence of an alarm within certain filtering periods, determines the alarm filtering imposed.

Alarm masking is designed to prevent the unnecessary reporting of alarms. The masked alarm is inhibited from generating reports if an instance of its superior alarm is active and fits the masking periods. A *masking hierarchy* determines the priority of each alarm type. Alarm masking is also enabled/disabled individually for each alarm. If an alarm changes state at any time the network management system is informed.

The simple example above and its associated processing, illustrate that fault determination is not a straightforward process. There are numerous combinations of possible alarm events, with differing arrival orders at the Element Controller. Added to this complexity is the fact that within the same physical network configuration, individual alarms can be configured in different states such as *masking disabled* or *masking enabled* and the network itself may be switched between different levels of protection, resulting in different symptoms being presented under the same fault conditions.

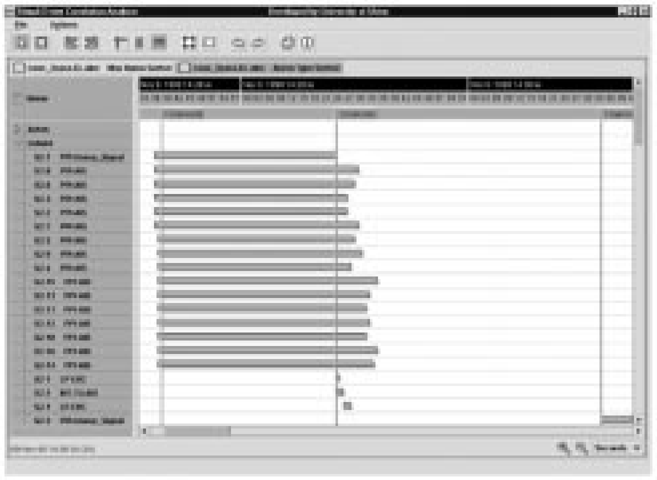


Fig. 2. Output from NxGantt, an event correlation analyzer.

B. Alarm Correlation

At the heart of fault management is *alarm correlation* [9], [11]. The alarm events may be the first indication that a fault has occurred. Since the primary concern is to avoid network traffic interruption, a quick diagnosis is essential.

Alarm correlation is a conceptual interpretation of multiple alarms, giving them a new meaning [11] and from that, potentially, creating derived higher order alarms [15]. Jakobson and Weissman proposed correlation as a generic process involving six operations: *compression*, *suppression*, *count*, *Boolean patterns*, *generalization*, and *specialization*.

Some of the definitions of these operations that follow use the set of alarms shown in Fig. 2. This is part of a screenshot of an alarm analysis tool [16] that displays an event log's alarm events (horizontal bars) against user action events (vertical lines) over time. It shows a small time window on a multiplexer (Enfield) that has alarms active on all 16 ports on the tributary card in slot 2, on November 9 1998; for example, the alarm on slot 2 port 8 was active from 2:28:36 pm to 2:29:30 pm.

1) Compression $[A, A, \ldots .A]$ $=>$ A*: Multiple occurrences of an alarm can be compressed into a single alarm*. In the example, PPI-AIS, an *Alarm Indication Signal* (AIS) has been detected in the incoming traffic. In the test configuration used it is common practice to connect the tributary ports together. Thus the PPI-AIS alarms are the same and can be compressed to one PPI-AIS alarm.

2) Suppression $[A, B, p(A) < p(B)]$ $=>$ $\emptyset$*: A low-priority alarm may be inhibited in the presence of a higher alarm*. This is generally referred to as *masking*. A consequence of the PPI-AIS in Fig. 2 is that an INT-TU-AIS alarm is raised briefly. This is of higher priority so the PPI-AIS alarm can be suppressed.

3) Count $[n \times A]$ $=>$ B*: A specified number of occurrences of an alarm can be substituted with a new alarm*. If the alarms in Fig. 2 were not from a test configuration the 15 occurrences of PPI-AIS could indicate a fault with the tributary card. If so, an NE-Card_Fail alarm could be substituted for the 15 PPI-AIS alarms.

4) Boolean Pattern $[A, B, \ldots .T, \wedge, \vee, \neg]$ $=>$ C*: A new alarm can be substituted for a set of alarms satisfying a Boolean pattern*. An MS-AIS alarm in a network indicates that an AIS alarm has been detected in the traffic address, implying a failure at the far multiplexer. If AU-AIS injection is enabled for the MS-AIS alarm then MS-FERF will always be raised. Thus, MS-AIS $\wedge$ MS-FERF $=>$ AU-AIS.

5) Generalization $[A, A \subset B]$ $=>$ B*: An alarm can be generalized to its super class*. PPI-AIS is the lowest of the AIS alarms. It may be superseded by other higher priority AIS alarms (INT-TU-AIS, TU-AIS, INT-AU-AIS, AU-AIS).

6) Specialization $[A, A \supset B]$ $=>$ B*: An alarm can be specialized to a subclass*. Alarm specialization is the opposite of alarm generalization and provides for substitution of an alarm by a more specific subclass of the alarm. In telecommunications, alarms are usually thought of in terms of generalizations—that is, higher order alarms mask lower order alarms. When looking for the root cause of an alarm, however, consideration of the low-level alarm can be helpful.

This section has considered how alarm correlation can reduce the number of alarms presented under fault conditions to the operator. This reduced set of symptoms may also be used by soft computing techniques to predict the underlying fault(s).

IV. Fusing Hard and Soft Approaches to Fault Management

Ideally a fault management system should be able to facilitate fault prediction. There are two aspects to prediction [18].

- *Fault identification/diagnosis*—prediction of the fault(s) that have occurred from the alarms present.
- *Behavior prediction*—warn the operator in advance of possible severe faults inferred from the alarms that are appearing.

Soft computing techniques can be used in both cases.

A. Soft Computing in Fault Management

Artificial neural networks is one of the main techniques in soft computing. Neural networks have given good predictive performance in several application areas, including pattern recognition, signal processing, time series processing, unsupervised clustering, visualization of complex data, data compression, control problems and image processing.

Their possible use in fault management has been considered in [19]–[22]. Specifically, neural networks can be used to

- identify fault-prone modules in software;
- learn index rules and adaptation functions for communication network fault resolution;
- identify faults in switching systems;
- act as a preprocessor for a fault diagnosis expert system.

Comprehensibility, however, is an issue [23] and telecommunication companies are reluctant to use such "black box" techniques in their fault management systems [24].

B. Probabilistic Networks for Fault Management

An alternative to neural networks is Bayesian belief networks (BBNs). These use probabilistic reasoning. BBNs meet the requirements of comprehensibility and transparency through their graphical structure and mathematical foundations. The graphical representation, which can be easily interpreted, is particularly attractive for appreciating and communicating fault symptom relationships (Fig. 3).

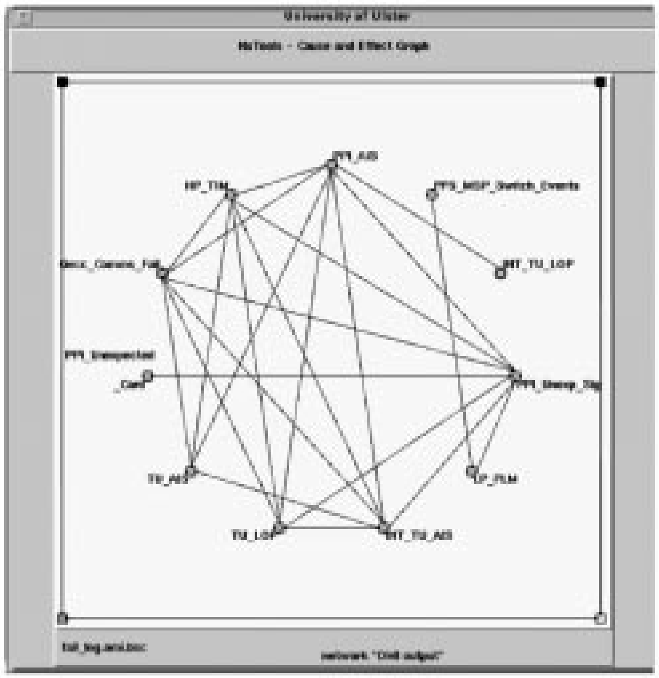


Fig. 3. Example induced probabilistic network (Bayesian belief network) from alarm data for fault management.

Capturing the alarm data and extending it with fault nodes in a BBN provides the basis of an expert system that can deal with imprecision and uncertainty. The technique can infer possible faults even when the available information is incomplete or inaccurate.

BBNs consist of a set of propositional variables represented by nodes in a directed acyclic graph. Each variable has a specified value taken from a set of possibilities. Directed arcs represent the probabilistic relationships between nodes. The absence of an arc between nodes indicates their independence.

The BBN can be used for deduction in fault management. For given alarm data, it will determine the most probable cause(s) of the supplied alarms, thus enabling the process to act as a fault identification or prediction system.

The graphical structure of potential causal relationships can be considered a complex form of alarm correlation. The alarms are connected by edges that indicate the probabilistic strength of correlation. It is created by *induction*, from stored alarm event data. Additionally, faults can be incorporated as nodes in the belief network. The belief network is then used to *deduce* the cause of faults from live alarm events.

1) Learning the Graph—Induction: In general, the structure of a Bayesian network is developed from a database of information on the frequency of occurrence of combinations of different variable values—the alarms in this case. The goal is to induce the network structure from the data. Heckerman provides a useful description of the problem [25]. There has been considerable work reported in the literature in this area, including that of Cooper and Herskovits who developed a commonly used algorithm for induction [26]. Unfortunately the general problem is NP-hard [27]. For a given number of variables there are a very large number of potential graphical structures that can be induced. In principle, to determine the best structure the data should be fitted to each possible graphical structure, that structure scored, and the structure with the best score selected. This is expensive, so algorithms for inducing networks from data need to take a more heuristic approach as the number of variables grows.

Specifically, there are $2^{k(k-1)/2}$ possible independence graphs for a k-dimensional random vector. So, for example, there are 64 probabilistic models for $k = 4$, and 32 768 models for $k = 6$, and 268.4 million for $k = 8$. Thus, without applying any prior knowledge, the previously mentioned Nortel multiplexer with 88 alarm types has approximately 9.22 e^{153} possible belief networks.

Fig. 3 is an example of a network induced from alarm data. The occurrence of an alarm changes the probabilities. For example, if alarm type LP-PLM is observed, this increases the probability that alarm PPI-Unexpl_Signal will be observed.

2) Using the Graph—Deduction: Once the BBN graph has been developed [Fig. 4(a)] it can be used as the knowledge structure of an expert system [Fig. 4(b)]. The cause and effect relationships (the correlations between the alarms) may be deduced or inferred from given alarm data to determine the underlying fault or fault possibilities.

Once an alarm has been reported the effects of that observation are propagated throughout the network and the other marginal probabilities are updated. In simple or special case belief networks the marginal probabilities (likelihoods) of each state can be calculated from the knowledge of the joint distribution using the product rule and Bayes' theorem. Some algorithms calculate the marginals exactly but the calculation on graphical structures is NP-hard [28]. Therefore many researchers have developed algorithms that produce an approximate answer, sacrificing accuracy for a lower computational overhead. These approximate algorithms have resulted in reasonable performance for inference on belief networks that contain hundreds or even thousands of nodes.

In fault management the BBN should be seen as an addition to existing rule-based approaches rather than a replacement for them. In effect, the BBN sits above the current system manager using the reduced set of alarms that is normally presented to the operator [Fig. 4(b)], and offering advice on possible faults.

This fusion means that when learning (inducing) the graphical structure and parameters of a BBN, the volume of alarm data may be reduced by using first-stage alarm correlation (in this case the rule base system). The expert system fault identification approach—in this case the deduction from the belief network—then operates on the reduced set of alarms. As a result, the BBN is much less complex which has the additional advantage of being more likely to gain user acceptance.

C. Genetic Algorithms for Fault Management

Section IV-B1 discussed the induction or learning of belief networks. A soft computing algorithm to induce cause and effect networks (probabilistic reasoning and specifically BBNs) has been developed using a genetic algorithm [29]. The Cause and Effect Genetic Algorithm (CAEGA) manipulates BBNs to produce (breed) new, potentially better, belief networks. It has given good results when applied to fault management telecommunications data [30]. This is used on large amounts of historical fault management data rather than a set of data specifically collected from a known fault episode on the network.

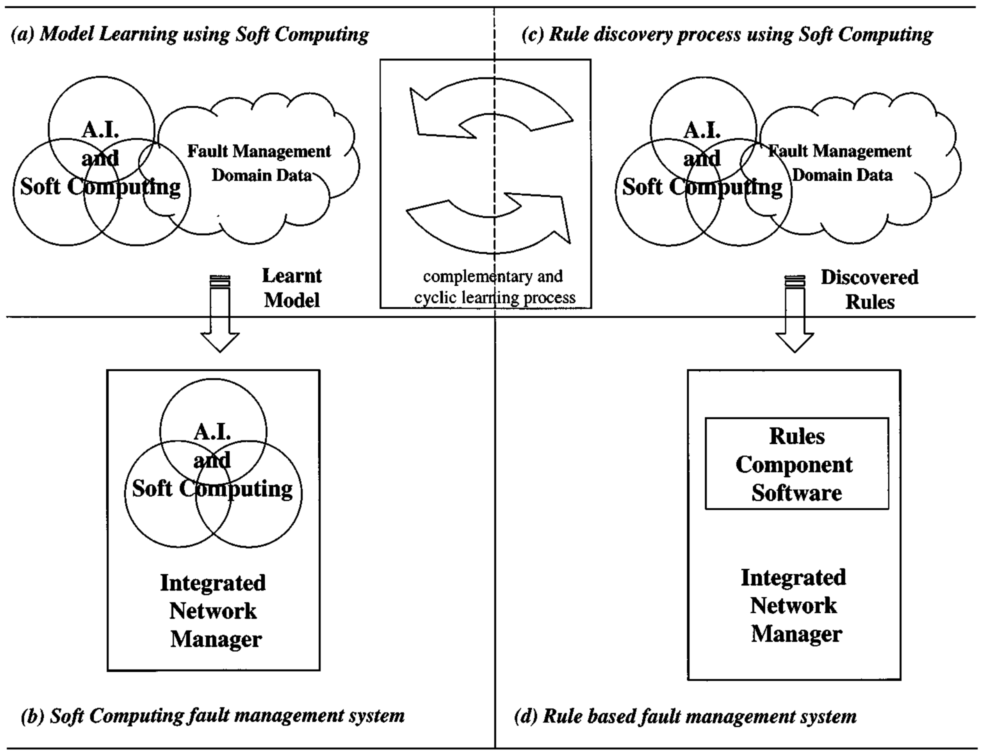

(a) Model Learning using Soft Computing
A.I. and Soft Computing
Fault Management Domain Data
Learnt Model
complementary and cyclic learning process
(c) Rule discovery process using Soft Computing
A.I. and Soft Computing
Fault Management Domain Data
Discovered Rules
A.I. and Soft Computing
Integrated Network Manager
Rules Component Software
Integrated Network Manager
(b) Soft Computing fault management system
(d) Rule based fault management system

Fig. 4. Fusing soft computing and hard computing for fault management. Two target systems are depicted: (b) one incorporating a soft computing solution within the network manager and the other (d) incorporating rules discovered via soft computing techniques. (a) Learning and (c) discovery processes for both approaches may be complementary and cyclic in nature.

The algorithm takes as input a data (contingency) table, the variables (nodes) and any existing beliefs. Each variable is specified by its name, a description, and the number of values that it can take, together with a descriptor for each value. In the data table, each tuple is replaced by a vector of integer values—one integer for each value—and the number of occurrences of each distinct vector is also recorded. The optional user-specified list of belief network structures, defined by child–parent lists, may have been generated by other algorithms or be the beliefs of an expert.

The algorithm works in the following way. Internally, breeders are specified by a square matrix of binary values in which a unit value indicates a parent–child relationship between two variables and a zero indicates that there is no direct relationship. An initial breeding stock is created from the breeders specified in the input data file and by random creation of graphs that represent the belief network structure. BBNs require acyclic graphs but not all binary matrices define valid acyclic graphs, thus any invalid matrices created by breeding or mutation are pruned until they are valid. At each generation, pairs of breeders are selected randomly to exchange genetic material and thereby create new breeders. Others are selected for random mutations. Scoring is performed using the posterior probability function of Cooper and Herskovits [26]. However, there is a problem of overfitting models when the saturated model is a perfect match to the data. A penalty is applied to the score to overcome this problem by limiting the number of parents each child may possess.

Although not considering the work as soft computing, Larrañaga has performed extensive research in the area of learning belief networks by employing genetic algorithms. In 1997, he also looked at using Cooper and Herskovits' [26] K2 algorithm in a genetic algorithm [31].

D. Using Soft Computing for Rule-Discovery

In the suggested approach, soft computing is not replacing the conventional rule-based approach but fusing with it. Thus the problems associated with rule-type systems still remain, namely

- the knowledge acquisition bottleneck;
- the inability to handle uncertainty;
- once developed, the burden of maintaining the rule-base.

Soft computing techniques can assist with such problems by being used for rule discovery as suggested in Fig. 4(c) and (d). It is also proposed that the induction of cause and effect graphs from the data be used. Just as a rule base can be converted into a belief network [32], strong relations from the belief network can be converted into rules. Thus the two approaches are complementary, leading to a cyclic process of refinement when the system is under development or in maintenance [Fig. 4(a) and (c)] [33].

V. Conclusion

Downtime in a network not only results in a loss of revenue but can lead to serious financial contractual penalties. It is therefore not surprising that network operators are extremely keen to remedy faults as quickly as possible. To this end not only is the identification of faults critical but a prediction of any likely faults would greatly assist in managing and assessing maintenance strategies.

Effective fault management means dealing efficiently with the sheer volume of alarm event traffic that occurs and helping the engineer locate a specific fault from the symptoms presented. The rule-based approach of monitoring, filtering and masking, currently used in telecommunication systems successfully reduces the number of alarms for analysis but gives little further guidance on the source of a fault.

This paper considered how soft computing techniques might be used to enhance the traditional hard approach. Specifically, it presented the use of Bayesian belief networks and genetic algorithms to supplement a rule-based approach.

In the future, the present convergence between the traditional telecommunications industry and the data communications market is likely to be viewed as only the starting point of the merging of computing and telecommunications. From this perspective every computer or device connected to a network can be viewed as a network element potentially raising its own alarm events resulting in even more complexity. The need for improved techniques, such as those suggested in this paper, will therefore become ever more acute.

REFERENCES

[1] L. A. Zadeh, "Fuzzy logic and soft computing: Issues, contentions and perspectives," in *Proc. Int. Conf. Soft Computing*, Iizuka, Fukuoka, Japan, 1994, pp. 1–2.

[2] D. Dubois and H. Prade, "Soft computing, fuzzy logic, and artificial intelligence," *Soft Comput.*, vol. 2, no. 1, pp. 7–11, 1998.

[3] P. P. Bonissone, Y.-T. Chen, K. Goebel, and P. S. Khedkar, "Hybrid soft computing systems: Industrial and commercial applications," *Proc. IEEE*, vol. 87, pp. 1641–1667, Sept. 1999.

[4] Y. Suzuki, S. J. Ovaska, T. Furuhashi, T. Roy, R. Cranfield, and Y. Dote, Eds., *Soft Computing in Industrial Applications*. Berlin, Germany: Springer-Verlag, 2000.

[5] L. A. Zadeh, "The role of soft computing and fuzzy logic in conception, design and deployment of intelligent systems," in *Proc. Int. Workshop on Soft Computing in Industry*, Muroran, Hokkaido, Japan, 1996, pp. 136–137.

[6] R. John and J. R. Birkenhead, Eds., *Developments in Soft Computing, Advances in Soft Computing*. Berlin, Germany: Springer-Verlag, 2002.

[7] P. P. Bonissone, "Soft computing: The convergence of emerging reasoning techniques," *Soft Comput.*, vol. 1, no. 1, pp. 6–18, 1997.

[8] D. Bustard, W. Liu, and R. Sterritt, Eds., *Computing in an Imperfect World, First International Conference, Soft-Ware 2002*. Berlin, Germany: Springer-Verlag, 2002, LNCS 2311.

[9] A. T. Bouloutas, S. Calo, and A. Finkel, "Alarm correlation and fault identification in communication networks," *IEEE Trans. Commun.*, vol. 42, pp. 523–533, Feb.–Apr. 1994.

[10] M. Klemettinen, "A knowledge discovery methodology for telecommunication network alarm databases," Ph.D. dissertation, Univ. Helsinki, Finland, 1999.

[11] G. Jackobson and M. D. Weissman, "Alarm correlation," *IEEE Network*, vol. 7, no. 6, pp. 52–59, Nov. 1993.

[12] "ITU-T recommendations, M.3010 principles for a telecommunications management network,", Feb. 2000.

[13] K. Harrison, "A novel approach to event correlation," Intelligent Networked Computing Lab, HP Labs, Bristol, U.K., HP-94-68, 1-10, , July 1994.

[14] I. Bratko and S. Muggleton, "Applications of inductive logic programming," *ACM Trans. Commun.*, vol. 38, no. 11, pp. 65–70, 1995.

[15] D. Gürer, I. Khan, R. Ogier, and R. Keffer, *An Artificial Intelligence Approach to Network Fault Management*. Menlo Park, CA: SRI Int..

[16] R. Sterritt, K. Adamson, E. P. Curran, and C. M. Shapcott, "Visualization and context of telecommunications data," in *Proc. 17th Int IASTED Conf Applied Informatics*, Innsbruck, Austria, 1999, pp. 588–591.

[17] "ITU-T recommendations, G.732 characteristics of primary PCM multiplex equipment operating at 2048 kbit/s,", Nov. 1988.

[18] R. Sterritt, A. H. Marshall, C. M. Shapcott, and S. I. Mcclean, "Exploring dynamic Bayesian belief networks for intelligent fault management systems," *IEEE Int. Conf. Systems, Man, and Cybernetics*, vol. V, pp. 3646–3652, 2000.

[19] T. Clarkson, "Applications of neural networks in telecommunications," in *Proc. ERUDIT Workshop on Application of Computational Intelligence Techniques in Telecommunication*, London, U.K., 1999.

[20] Y. A. Sekercioglu, A. Pitsillides, and A. Vasilakos, "Computational intelligence in management of ATM Networks: A survey of current state of research," in *Proc. ERUDIT Workshop on Application of Computational Intelligence Techniques in Telecommunication*, London, U.K., 1999.

[21] H. Wietgrefe, K. Tochs, K. Jobmann, G. Carls, P. Frohlich, W. Nejdl, and S. Steinfeld, "Using neural networks for alarm correlation in cellular phone networks," in *Proc. Int. Workshop on Applications of Neural Networks in Telecommunications*, 1997.

[22] R. D. Gardner and D. A. Harle, "Alarm correlation and network fault resolution using the kohonen self-organizing map," in *Proc. GLOBECOM'97*, vol. 3, Phoenix, AZ, 1997, pp. 1398–1402.

[23] E. Oja, G. Dorffner, and T. Heskes, *Network of Excellence in Neural Networks (NEuroNet) Technological Roadmap*, June 1999.

[24] K. Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen, "Knowledge discovery from telecommunication network alarm databases," in *Proc. 12th Int. Conf. Data Engineering (ICDE'96)*, New Orleans, LA, 1996, pp. 115–122.

[25] D. Heckerman, "Bayesian networks for knowledge discovery," in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Cambridge, MA: AAAI/MIT Press, 1996, pp. 273–305.

[26] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, vol. 9, pp. 309–347, 1992.

[27] D. M. Chickering and D. Heckerman, "Learning Bayesian networks is NP-hard," Microsoft Research, Microsoft Corp., Redmond, WA, Tech. Rep. MSR-TR-94-17, 1994.

[28] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artif. Intell.*, vol. 42, no. 2-3, pp. 393–405, 1990.

[29] R. Sterritt, K. Adamson, M. Shapcott, D. Bell, and F. McErlean, "Using A.I. for the analysis of complex systems," in *Proc. IASTED Int. Conf. Artificial Intelligence and Soft Computing*, Banff, AB, Canada, 1997, pp. 113–116.

[30] R. Sterritt, K. Adamson, M. Shapcott, N. Welle, D. Bell, and W. Liu, "P-CAEGA: A parallel genetic algorithm for cause and effect networks," in *Proc. IASTED Int. Conf. Artificial Intelligence and Soft Computing*, Banff, AB, Canada, 1997, pp. 113–116.

[31] P. Larrañaga, B. Sierra, M. Y. Gallego, M. J. Michelena, and J. M. Pikaza, "Learning Bayesian networks by genetic algorithms. A case study in the prediction of survival in malignant skin melanoma," in *6th Conf. in Artificial Intelligence in Medicine, Europe, (AIME'97)*, E. Keravnou, C. Garbay, R. Baud, and J. Wyatt, Eds. Grenoble, France, Mar. 23–26, 1997, pp. 261–272.

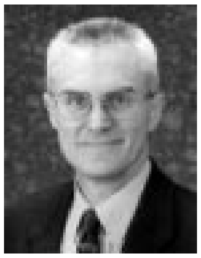
[32] M. Korver and P. J. F. Lucas, "Converting a rule-base expert system into a belief network," *Med. Inform.*, vol. 28, no. 3, pp. 219–241, 1993.

[33] R. Sterritt and D. W. Bustard, "Practical intelligent support for rule discovery in fault management systems," *Cybern. Syst.*, vol. 33, no. 6, 2002.

[34] R. Sterritt, "Fault management and soft computing," in *Proc. Int. Symp. on Soft Computing and Intelligent Systems for Industry*, Paisley, U.K., June 26–29, 2001, 7 pp. (CD-ROM).

Roy Sterritt (M'01) received the B.Sc. (honors) degree in computing and information systems and the M.A. degree in business strategy from the University of Ulster, U.K., in 1994 and 1999, respectively.

He spent three years with IBM North Harbour and IBM Hursley Park, U.K., as a Software Engineer working on applications to support project management and risk assessment. He then returned to the University of Ulster to research automated and intelligent approaches to the development and testing of fault management telecommunications systems in a series of collaborative projects with Nortel Networks. Currently, he is a lecturer in computing science at Ulster, with research interests in the application of artificial intelligence and soft computing to software engineering and autonomic computing.



David W. Bustard (M'95) received the B.Sc. degree in physics in 1971, the M.Sc. degree in computer science 1973, and the Ph.D. degree in computer science in 1980, all from from Queen's University, Belfast, U.K.

He is Head of the School of Information and Software Engineering at the University of Ulster and Coordinator of the Software Engineering Research Group. He has been a full Professor of computing for 12 years, since moving from the Computer Science Department at Queen's University, where he worked since 1974. His main research area is software engineering, with specific interests in requirements engineering, systems evolution and soft computing.