An Artificial Intelligence Approach to Network Fault Management[‡]

Denise W. Gürer, Irfan Khan, Richard Ogier
SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025 USA
`phone: (415)859-5911 fax: (415)859-4812 email: {gurer, khan, ogier}@erg.sri.com`
Renee Keffer
Sprint, 9300 Metcalf, Overland Park, KS 66212 USA
`phone: (913) 967-5342 email: renee.j.keffer@sprint.sprint.com`

Abstract

Traditionally, network management activities, such as fault management, have been performed with direct human involvement. However, these activities are becoming more demanding and data intensive, due to the heterogeneous nature and increasing size of networks today. For these reasons, it is becoming necessary to automate network management activities. Artificial intelligence technologies can play an important role in the problem solving and reasoning techniques that are employed in fault management. Expert systems have been successfully applied to some types of fault management. However, these systems are not flexible enough for today's evolving network needs. We propose a hybrid AI solution that employs both neural networks and case-based reasoning techniques for the fault management of heterogeneous distributed information networks.

Overview of Fault Management Activities

Today's high speed, heterogeneous networks represent a complex and data intensive environment that requires different solutions from the traditional methods performed by human operators. Automation of network management activities can benefit from the use of artificial intelligence (AI) technologies, including fault management, performance analysis, and traffic management. Here we focus on fault management, where the goal is to proactively diagnose the cause of abnormal network behavior and to propose, and if possible, take corrective actions. First an overview of fault management activities and responsibilities is given. Then follows a discussion on how AI technologies may be used to automate the fault management process, in particular neural networks (NNs) and case-based reasoning (CBR). The following discussion assumes a telecommunications synchronous optical network (SONET) with asynchronous transfer mode (ATM) switching. However, the same model can be applied to any heterogeneous distributed information network.

Essentially, network faults can be classified into hardware and software faults, which cause elements to produce incorrect outputs, which in turn can cause overall failure effects in the network such as congestion (Wang, 1989). Examples of hardware faults are failures of an element due to a failing or a weakness in their logical design, or elements malfunctioning due to simple wear and tear or through external forces such as accidents, acts of nature, being mishandled, or improperly installed. Examples of software faults include failure of elements due to incorrect or incomplete design of their software, erratic behavior of elements or the network due to software bugs (e.g., incorrect packet header processing), and slow or faulty service by the network due to incorrect information (e.g., incorrect routing tables).

The flow of fault management, shown in Figure 1, can be described as follows; (1) collect alarms, (2) maintain customer satisfaction through immediate action, (3) filter and correlate the alarms, (4) diagnose faults through analysis and testing, (5) determine a plan for correction, display correction options to users, and implement the correction plan, (6) verify the fault is eliminated, (7) record data and determine the effectiveness of the current fault management function.

The first step in fault management is to collect monitoring and performance alarms. Typically alarms are produced by either managed network elements (e.g., ATM switches, customer premise equipment) or by a statistical analysis of the network that monitors trends and threshold crossings. Alarms can be classified into two categories, physical and logical, where physical alarms are hard errors (e.g., a link is down), typically reported through an element manager, and logical alarms are statistical errors (e.g., performance degradation due to congestion).

Once the alarms have been reported and collected, adequate service must be maintained through immediate action. This action serves as a temporary stop gap while the fault diagnosis process proceeds, in order to ensure the customer does not experience a loss or decrease in service. An example may be routing traffic in an opposite direction in the case of a SONET ring break, or in the case of a malfunctioning switch, rerouting around the problem area.

[‡]This research was partly supported by Sprint Corporation and partly by internal funding at SRI International. The opinions expressed in this publication do not necessarily reflect a position or policy of Sprint.

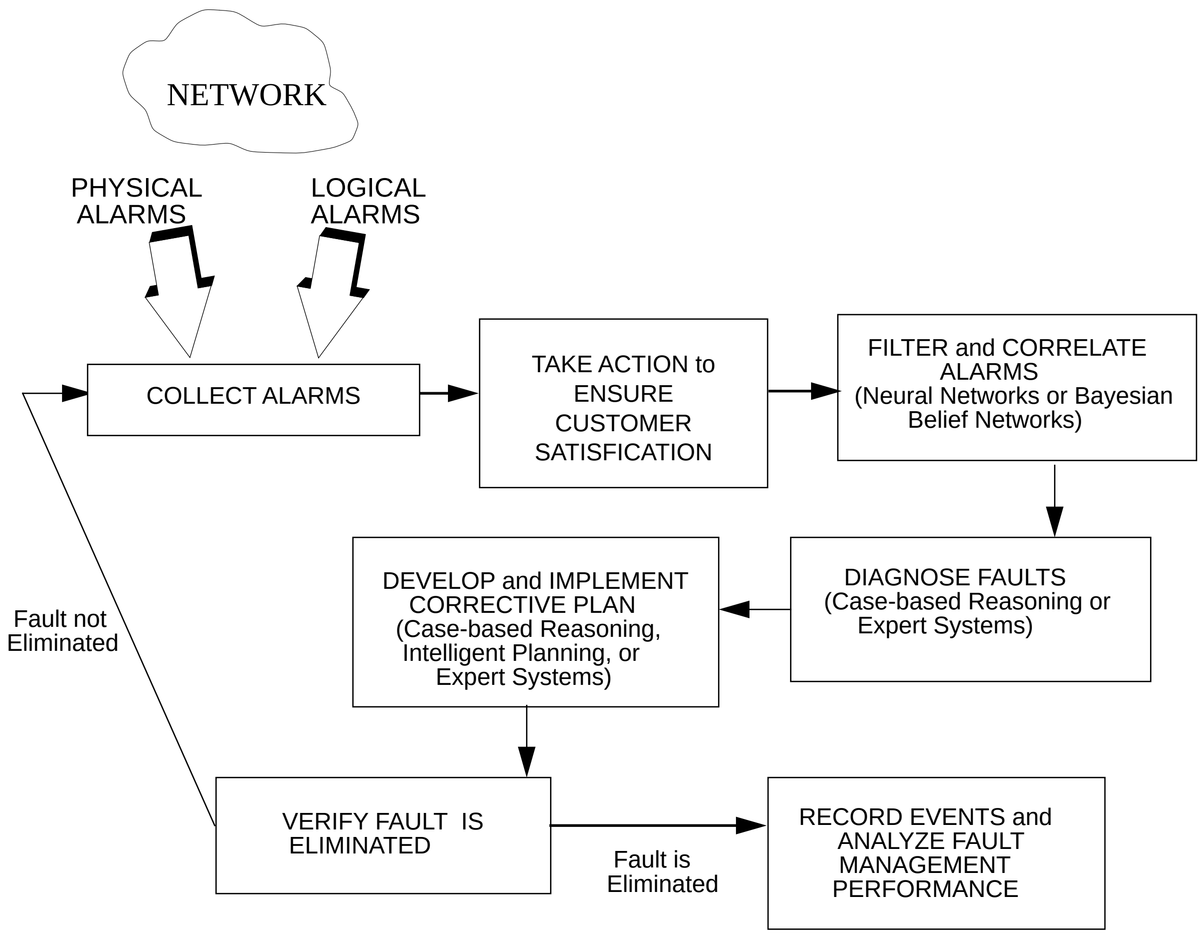
NETWORK
PHYSICAL ALARMS
LOGICAL ALARMS
COLLECT ALARMS
TAKE ACTION to ENSURE CUSTOMER SATISFICATION
FILTER and CORRELATE ALARMS (Neural Networks or Bayesian Belief Networks)
DIAGNOSE FAULTS (Case-based Reasoning or Expert Systems)
DEVELOP and IMPLEMENT CORRECTIVE PLAN (Case-based Reasoning, Intelligent Planning, or Expert Systems)
Fault not Eliminated
VERIFY FAULT IS ELIMINATED
Fault is Eliminated
RECORD EVENTS and ANALYZE FAULT MANAGEMENT PERFORMANCE

Figure 1: The fault management process and possible AI technologies.

After customer satisfaction is ensured, the next step is to filter and correlate the alarms. Alarm filtering is a process that analyzes the multitude of alarms received and eliminates the redundant alarms (e.g., multiple occurrences of the same alarm),. Alarm correlation is the interpretation of multiple alarms such that new conceptual meanings can be assigned to the alarms, creating derived alarms. Faults are identified by analyzing the filtered and correlated alarms and by requesting tests and status updates from the element managers, which provide additional information for diagnosis.

Once a fault has been diagnosed, corrective procedures may be undertaken by the network to eliminate the cause of the fault. The fault management system's role in correction is to develop a plan or series of actions, and to initiate this plan with other functions within the network. As much of the correction as possible is performed automatically without human intervention, although at times it is necessary for a technician to physically go to a site to replace a part, or for a programmer to debug some software. The correction must be verified through testing requests sent to the element managers, where if the fault does not disappear, more data is analyzed and the diagnostic process is repeated.

Another step in fault management is to collect data about the effectiveness of the fault management process in order to monitor damage perpetrated by faults and the costs of repair. As outlined in (Byrne, 1994), questions regarding how often faults are occurring and how many faults are affecting service should be normalized to account for network size and number of customers. Other questions regarding length of service interrupts, number of times a fault is correctly identified, and number of hours to repair, should be normalized according to the number of relevant faults detected. These statistics can be used to analyze the performance of the fault management system and can be used in other network management activities such as capacity planning in determining current and projected costs for the maintenance of the network. In addition, a finer grained analysis of the types of faults can shed some light on the reliability of different types of equipment.

AI Applications for Fault management

The more complex processes of fault management include alarm filtering and correlation, fault identification, and correction. Many of these functions involve analysis, correlation, pattern recognition, clustering or categorization, problem solving, planning, and interpreting data from a knowledge base that contains descriptions of network elements and topology. Artificial intelligence technologies are ideal for these types of functionalities.

Currently, most systems employing AI technologies for fault diagnosis are expert or production rule systems (Corn et al., 1988; Joseph et al., 1989; Wright, Zielinski & Horton, 1988; Yamahira, Kiriha & Sakata, 1989). Many of these systems are well developed; however, they have their limitations. Generally speaking:

- Expert Systems (ESs) cannot handle new and changing data. Rules are brittle and not robust when faced with unforeseen situations (e.g., a new combination of alarms due to changing network topology).
- They cannot learn from experience (i.e., they cannot use analogy to reason from past experiences or remember past successes and failures in the context of a current problem). The rules that are incorporated at development time cannot easily adapt as the network evolves.
- They do not scale well to large dynamic real world domains. It is difficult, especially for technicians or operators not familiar with AI, to add new rules without a comprehensive understanding of what the current rule base is and how a new rule may impact the rule base.
- They require extensive maintenance when the domain knowledge changes; new rules have to be added and old rules adapted or deleted.
- They are not good at handling probability or uncertainty. Fuzzy logic can be employed to create fuzzy rules. However, fuzzy expert systems still have the problems discussed above.
- They have difficulty in analyzing large amounts of uncorrelated, ambiguous and incomplete data. The domain must be well understood and thought out. This is not entirely possible in domains such as fault management.

These drawbacks argue for the use of different AI technologies that can overcome the above mentioned difficulties, either alone or as an enhancement of ESs. Probabilistic methods such as neural networks (NNs) or Bayesian belief networks (BBNs) are appropriate for correlation, while symbolic methods such as case-based reasoning (CBR) or expert systems are appropriate for fault identification. In many cases it is beneficial to use these technologies in cooperation with each other.

Another area of fault management where AI technologies can have a positive impact, is fault correction. CBR systems, ESs, or intelligent planning systems can develop plans or courses of action that will correct a fault that has been identified and verified. The application of these methods and the reasoning behind using them are discussed in the following sections.

Fault diagnosis

With today's large and dynamic networks, it is necessary for a fault management system to be able to adjust to changes that occur within the network elements and topology. Thus a NN should be trained or a BBN built to filter, correlate, and identify general categories of faults. However, in order to identify the exact location of a fault, a procedure is necessary where appropriate tests are requested and the results analyzed to further pinpoint a fault's location. This procedure involves a series of decisions based on human expertise, and therefore cannot be implemented by a NN or BBN alone. Therefore, it is appropriate to train a NN or construct a BBN to correlate alarms and recognize basic fault patterns, and to use symbolic processing such as CBR to further analyze the data, run tests, and pinpoint the exact identity and location of the fault.

A hybrid AI system is ideal due to the diverse nature of the fault management task. Rather than performing the whole task with one technique that is not ideal for all aspects, a couple of techniques are used as appropriate. Thus, the strengths of each technique are emphasized while the weaknesses are overcome by the other. A drawback to using a hybrid AI system is knowledge acquisition must be performed twice and in very different ways. For example, a neural network must be trained with large amounts of input/output data pairs and a CBR system must be seeded with initial cases drawn from experts and/or other symbolic data sources. This adds time to the development process and requires two types of knowledge. However, we believe the advantages of using a hybrid system in fault management outweigh the costs through the gain of robustness and accuracy.

The following discussion outlines the use of AI technologies as a solution to fault diagnosis where diagnosis has been split into a two step process: alarm filtering and correlation, and fault identification. First the fault management system receives an enormous number of alarms that need to be filtered and correlated into manageable categories. Second, the categorized alarms are further analyzed and tests are initiated on network elements in order to identify the fault that is the cause of the alarms.

Alarm Filtering and Correlation

Alarm filtering can be thought of as four processes: compression, count, suppression, and generalization (Jakobson & Weissman, 1993). Compression is the reduction of multiple occurrences of the same alarm into a single alarm; count is the substitution of a specified number of occurrences of similar alarms or alarm categories with a new alarm; suppression is inhibiting a low-priority alarm in the presence of a higher-priority alarm, and generalization is referring to an alarm by its superclass where the superclasses are determined by domain experts. These four processes are well defined and can thus be achieved through the use of rules or the application of ESs.

In addition to filtering, the alarms need to be correlated. Correlating alarms is a difficult task due to their inherent ambiguity. Even with large detailed amounts of data, there can still exist a significant amount of uncertainties and/or inconsistencies. In many cases, there is more than one plausible explanation for the underlying cause of a group of alarms. For instance, the non-occurrence of a remote event may cause a device waiting for that event, to time-out, with several possible causes for this lack of response: the device could be faulty, the response could be delayed due to congestion, or the local device's timer could be faulty. The production of incomplete data due to crash-recovery cycles, is another problem area for correlating alarms. Some network elements may have built-in mechanisms to reset themselves when a local fault tolerance level has been reached. This can cause the destruction of important evidence needed to diagnose faults, and can indicate that all is well when the failure condition still exists, causing a crash-recovery cycle.

The complex process of correlation can be thought of as substituting a set of alarms that match a predefined pattern with a new alarm. As described above, this process involves ambiguous, incomplete, and inconsistent data and can be thought of as a pattern recognition problem. Thus a probabilistic AI technology such as NNs or BBNs are appropriate for alarm correlation since they can analyze patterns of common behavior over circuits, and/or can handle ambiguity, and incomplete data.

Neural Networks for Alarm Correlation

Feedforward neural networks have already been proven effective in medical diagnosis, target tracking from multiple sensors, and image/data compression. It is therefore plausible that NNs would be effective for the similar problem of alarm correlation, found in fault diagnosis. In fact, the following properties of multilayer feedforward neural networks make them a powerful tool for solving these problems.

- NNs can recognize conditions similar to previous conditions for which the solution is known (i.e., pattern matching).
- They can approximate any function, given enough neurons, including boolean functions and classifiers. This gives NNs great flexibility in being able to be trained for different alarm patterns.
- They can generalize well and learn an approximation of a given function, without requiring a deep understanding of the knowledge domain. This is especially important in new technological areas such as ATM switch networks.
- They provide a fast and efficient method for analyzing incoming alarms.
- They can handle incomplete, ambiguous, and imperfect data.

In a feedforward neural net, shown in Figure 2, the neurons are arranged into layers, with the outputs of each layer feeding into the next layer. This model has a single input layer, a single output layer, and zero, one, or more hidden layers. As the name suggests, all connections are in the forward direction where there is no feedback. Feedforward networks are useful because of their ability to approximate any function, given enough neurons, and their ability to learn (generalize) from samples of input-output pairs. Learning is accomplished by adjusting the connection weights in response to input-output pairs, and training can be done either off-line, or on-line during actual use. Depending on how the training is done, these NNs can be characterized as being trained by supervised methods or by unsupervised methods.

Supervised NNs training data consists of correct input vector/output vector pairs as examples, used to adjust the neural net connection weights. An input vector is applied to the NN, the output vector obtained from the NN is compared with the correct output vector, and the connection weights are changed to minimize the difference. A well trained neural net can successfully generalize what it has learned from the training set (i.e., given an input vector not in the training set, it produces the correct output vector most of the time).

In unsupervised training there is no training data based on known input/output pairs. The NN discovers patterns, regularities, correlations, or categories in the input data and accounts for them in the output. For example, an unsupervised neural net where the variance of the output is minimized could serve as a categorizer which clusters inputs into various groups.

Unsupervised training is typically faster than supervised training and provides the opportunity to present patterns to operations personnel who can identify new output relations. For these reasons unsupervised training is used even in situations where supervised training is possible. However, for the domain of alarm correlation, input/output pairs can be easily produced, making supervised trained NNs a plausible choice for alarm correlation.

A different neural net approach for alarm correlation (Patton, Chen & Siew, 1994) uses the ability of neural networks to predict future behavior of general nonlinear dynamic systems. In this approach, a neural network predicts normal system behavior based on past observations and the current state of the system. A residual signal is generated based on a comparison between the actual and predicted behavior, and a second neural network is trained to detect and classify the alarms based on characteristics of the residual signal. This method can be used to identify basic categories for the alarms.

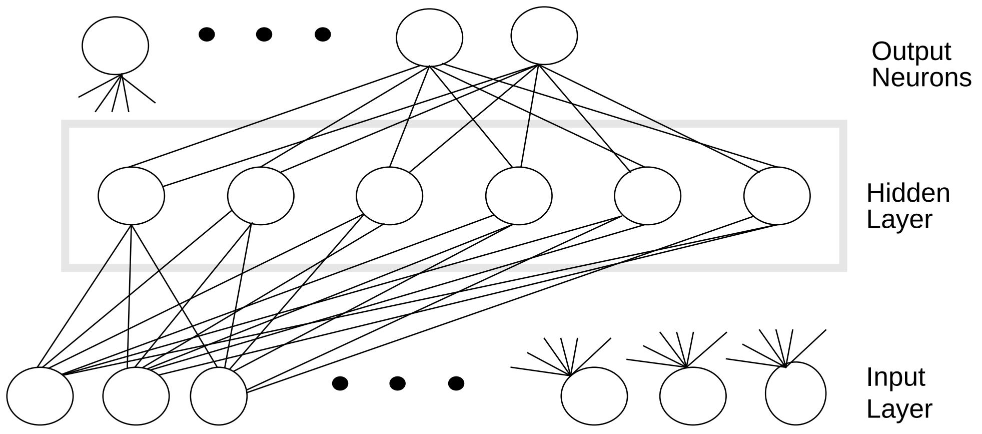

Figure 2: Model of a feedforward neural network.

An additional approach is to cast the pattern recognition problem into an optimization problem, making Hopfield NNs an appropriate tool for alarm correlation. This type of NN operates by using gradient methods to find a local minimum of a quadratic energy function that represents an optimization problem, and whose coefficients depend on the network's interconnection strengths. Methods such as mean-field annealing, repeated trials with random initial states, and tabu learning (Beyer & Ogier, 1990) can be used to find a local minimum that is nearly optimal. For example, in alarm correlation, the optimization problem is to identify the hypothesis that best explains the observed data. Goel et al. (1988) propose a neural network model based on Hopfield nets for solving a special case of this problem. In this case, the neural net would be designed so that states corresponding to the most likely hypotheses have the lowest energy.

Bayesian Belief Networks for Alarm Correlation

Bayesian belief networks are another possible choice for alarm correlation due to their ability to handle uncertainty and represent cause and effect relationships. A BBN is a representation consisting of nodes representing uncertain variables that are connected by arcs that represent cause and effect dependencies among the nodes.The information known about one node (i.e., effect node) depends on the information of its predecessor nodes that represent its causes. This relationship is expressed by a probability distribution for each effect node, based on the possible values of its predecessor nodes' variables. Note that an effect node can also lead into other nodes, where it then plays the role of a cause node. An important advantage that BBN's offer is the avoidance of building huge joint probability distribution tables that include permutations of all the nodes in the network. Rather, only a node's immediate predecessor's possible states and their effects on the node are necessary.

Due to BBN's form of knowledge representation, large amounts of interconnected and causally linked data can be represented. Generally speaking:

- BBNs can represent deep knowledge by modeling the functionality of the transmission network in terms of cause and effect relationships between element and network behavior and faults.
- They can provide guidance in diagnosis. Calculations over the same BBN can determine both the precedence of alarms and the areas that need further clarification in order to provide a finer grained diagnosis.

- They can handle noisy, transient, and ambiguous data due to their grounding in probability theory.
- They have a modular, compact, and easy to understand representation, when compared to other probabilistic methods.
- They provide a compact and well-defined problem space because they use an exact solution method for any combination of evidence or set of faults.

BBNs are appropriate for automated diagnosis because of their deep representations and precise calculations. A concise and direct way to represent a system's diagnostic model is as a BBN constructed from relationships between failure symptoms and underlying problems. A BBN represents cause and effect between observable symptoms and the unobserved problems so that when a set of symptoms are observed the problems most likely to be the cause can be determined. In practice, the network is built from descriptions of the likely effects for a chosen fault. In use as a diagnostic tool, the system reasons from effects back to causes.

The development of a diagnostic BBN requires a deep understanding of the cause and effect relationships in a domain, provided by domain experts. This is both an advantage and disadvantage. An advantage is the knowledge is not represented as a black box, as are the NNs. Thus, humanly understandable explanations of diagnoses can be given. The disadvantage is the realm of ATM-based networks is technologically immature enough that expertise in ATM fault diagnosis may be hard to find and to implement.

Fault Identification

The filtering and correlation of alarms is the first step of fault diagnosis. The second step involves further analysis and identification of the exact cause of the alarms, or the fault. This process is an iterative one where alarm data are analyzed and decisions are made whether more data should be gathered, a finer grained analysis should be executed, or problem solving should be performed. Gathering more data can consist of sending tests to network elements or requesting network performance data. Problem solving requires expert knowledge which consists of domain knowledge (i.e., knowledge about the network elements and topology, and typical faults), and meta-knowledge (i.e., knowledge about how to diagnose faults).

A symbolic AI technology such as ESs or CBR is ideal for achieving this functionality. Due to the limitations of expert systems described earlier, and the evolving nature of this process, CBR is the better choice for fault identification. CBR systems are more robust than ESs because they

- can handle new and changing data through their ability to use analogy,
- can learn from experience through the acquisition of new cases,
- can scale well to large knowledge domains due to the ability of their knowledge representation structures to collapse or merge into each other,
- do not require extensive maintenance,
- can critique or identify the "goodness" of a proposed solution based on simulations and/or previous cases from the case library,
- can use a method of knowledge acquisition that is less time consuming than expert system rule development.

Case-based reasoning is based on the premise that situations recur with regularity. Studies of experts and their problem-solving techniques have found that experts rely quite strongly on applying their previous experiences to the current problem at hand. CBR can be thought of as such an expert that applies previous experiences stored as cases in a case library. Thus, the problem-solving process becomes one of recalling old experiences and interpreting the new situation in terms of those old experiences.

Experiences are contextualized pieces of knowledge that are stored as cases in a case library. An important component of any CBR system (as with any knowledge-rich AI system) is deciding what the key attributes of a case are and on which attributes the cases will be indexed. Indices are combinations of the cases' attributes or descriptors and are used to predict which cases are useful to the current situation. Typically, the front-end of an application that uses CBR will have a graphical user interface (GUI) that allows a user to easily input new cases without requiring any detailed understanding of the underlying CBR engine. The cases can be input in a free-form manner (i.e., text) and are automatically indexed into the case library. A key benefit in using CBR systems is their ability to learn. One obvious method of learning is to add new cases with their new solutions and evaluations to the case library, emulating specialization. Another type of learning occurs because of the ability of CBR systems to collapse cases or parts of cases that have similar attributes or problem solutions into a single case, emulating generalization.

CBR problem solving can be depicted as a five-step process, as shown in Figure 3: (1) retrieval, (2) interpretation and adaptation, (3) evaluation and repair, (4) implementation, and (5) evaluation and learning. The first step is retrieving cases that best match the current situation or case. Thus it is crucial to use an appropriate indexing method, such as decision trees or nearest neighbor matching. Once a case is retrieved, it must be interpreted and then adapted. The interpretation process is a simple comparison between the retrieved cases and the current case. Adaptation is a complicated, domain-dependent process that uses rules to adapt the current case to the problem situation and propose an initial solution, based on the similarities and differences. The next step is an evaluation and repair cycle where the proposed solution is evaluated through comparisons to cases with similar solutions or through simulation, and the solution is modified accordingly. After the CBR system has found its best solution, the solution is implemented and the results are evaluated. The resulting evaluation, solution steps, and problem context are entered into a new case, which is then indexed into the case library, allowing the system to learn.

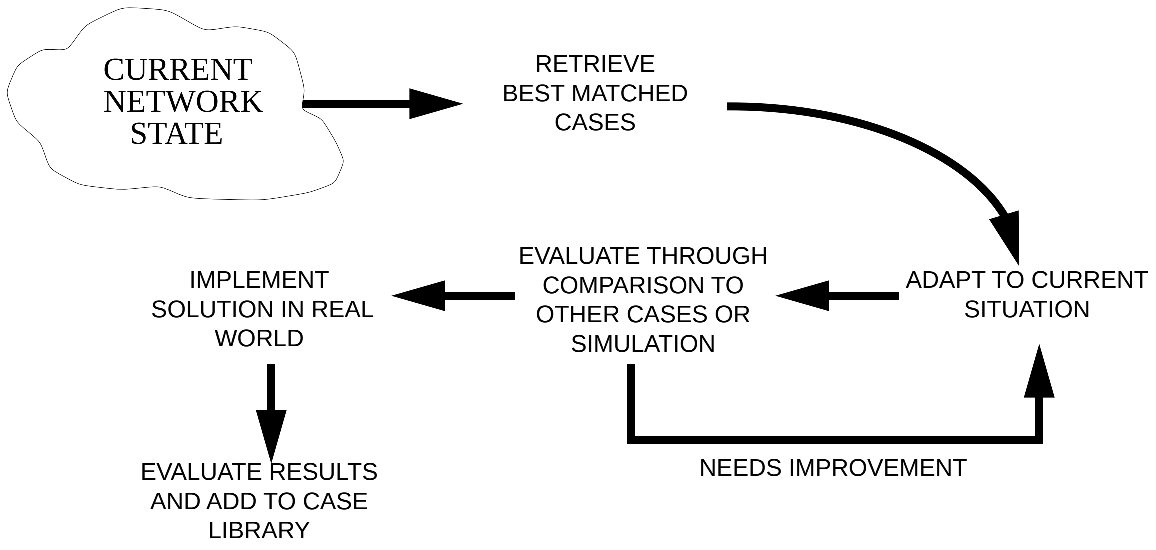
CURRENT NETWORK STATE
RETRIEVE BEST MATCHED CASES
ADAPT TO CURRENT SITUATION
EVALUATE THROUGH COMPARISON TO OTHER CASES OR SIMULATION
NEEDS IMPROVEMENT
IMPLEMENT SOLUTION IN REAL WORLD
EVALUATE RESULTS AND ADD TO CASE LIBRARY

Figure 3: The case-based reasoning process.

A Hybrid AI System for Fault Diagnosis

Based on the discussions of the previous sections, a good candidate for a fault diagnosis system is a hybrid system that uses both probabilistic and symbolic problem solving AI techniques. Below we discuss a NN/CBR system for fault diagnosis, shown in Figure 4.

Network alarms are fed into an ES that filters the alarms through compression, count, suppression, and generalization. The pattern recognizing ability of neural networks can be used for correlating alarms and recognizing alarm patterns. Filtered alarms serve as input into a NN, trained to recognize common fault categories. The output consists of the most likely fault types, protocol types, and geographic area of the fault. This analysis is then fed into a case-based reasoning system where testing and further analysis is performed. The CBR system can make decisions, based on past problem solving experiences, whether to gather more data, problem solve, or run the data through the same NN or a different NN that was trained on a finer grained set of alarms.

The first step in developing a NN component for the fault management system is to train the NN. Neural networks learn to recognize patterns by generalizing from many examples. Supervised training should be used where many examples of similar and different faults are fed into the network with known outcomes. Once the network is trained, it can be integrated with the CBR system.g.

after receiving the filtered and correlated alarms, the CBR system attempts to identify what information would further the diagnosis of the fault. The CBR system needs to decide on its own what additional tests need to be made on the network elements and what granularity of NN to use to further analyze the data. To achieve this end, the CBR library is searched for the cases that most resemble the current situation. Once the cases are found, similarities and differences are discovered and a new solution is proposed. This continues in an iterative manner until the fault is identified.

Once a fault is identified, the whole problem solving episode should be analyzed. The value of the tests (i.e., useful, not useful), the steps taken, any circuitous paths or dead-ends taken, and the success of the analysis should be stored into a new case. This information, in addition to contextual information, comprise a new case which is then indexed into the case library. In this manner, the fault management system is able to learn from its successes and failures.

Fault Correction

Another goal of an automated network management system is automated fault correction. This requires some knowledge about the network elements and topology, and old and new technologies. Some general technical problems that arise in correcting faults as outlined by (Dupuy et al., 1989) are: uncoordinated correction, manual correction, and old technology.

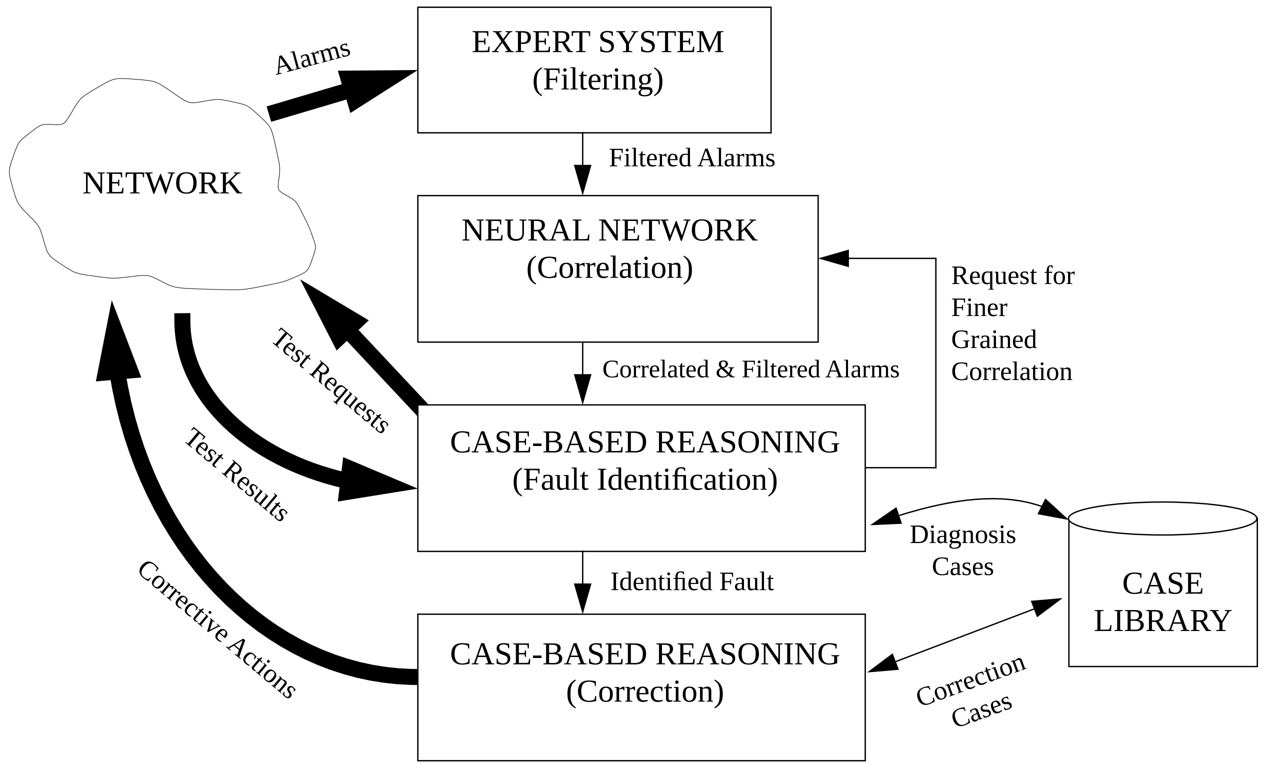
Alarms
EXPERT SYSTEM (Filtering)
Filtered Alarms
NETWORK
NEURAL NETWORK (Correlation)
Request for Finer Grained Correlation
Test Requests
Correlated & Filtered Alarms
Test Results
CASE-BASED REASONING (Fault Identification)
Diagnosis Cases
CASE LIBRARY
Identified Fault
Corrective Actions
CASE-BASED REASONING (Correction)
Correction Cases

Figure 4: A hybrid AI system for automated network fault management.

Uncoordinated correction may cause performance conflicts to occur within a set of network elements, which in turn may cause a surge of other control processing in other nodes, leading to congestion and the emergence of other faults. For example, if a fiber breaks, uncoordinated restoration actions of the section, path, and line layers may result in none of the layers being properly re-established. These restoration actions may also cause congestion within the network.

A logistics problem of fault correction is manual correction. In many cases, correction involves the resetting of a device. If the device is unresponsive to remote communications, it may be required to have someone on site manually correct the device. Obviously, this is undesirable for large scale networks.

Old technology is another technical concern for fault correction. As networks evolve over time, new standards and equipment are incorporated and implemented. In many cases, the new and improved devices are not completely compatible with the old devices and standards. For example, even though a new type of switch may have the same functionality as its older counterpart, it would exhibit improved performance, such as faster recovery abilities in response to a fault. The recovered switches would immediately allow a higher traffic throughput, which in turn would put more pressure on the still recovering older switches.

These concerns argue for a symbolic AI technique that can represent the domain knowledge and problem solving methods. Some possible choices are hierarchical AI planning, expert systems, or case-based reasoning.

AI planning would be able to handle the complicated logistics of planning and scheduling the correction, especially timing corrections so as to avoid uncoordinated recovery and to account for old technology. The main drawback is that today's current planning systems that can be applied to real world domains are still prototypes and are too slow for a real-time solution to fault correction. However, in the near future, when planning systems become more optimized and machines are faster, they will prove to be a viable option.

Additionally, CBR systems will be able to use previous experiences in formulating a new fault correction strategy. For example, when a device is unresponsive to manual correction, similar cases (i.e., previous fault correction solutions) that had the same problem can be retrieved from the case library and their solutions analyzed. Drawing analogies from those cases, the CBR system could come up with an appropriate solution within the context of the current problem.

Expert systems are a strong candidate for fault correction, as are CBR systems. For the reasons discussed previously, CBR is a better choice. In particular, CBR systems will be able to learn from mistakes. For example, if an uncoordinated recovery occurs which leads to congestion, the fault correction steps that led to that congestion will be remembered and noted as a negative solution that led to congestion. A rule-based system, on the other hand, would have to be updated and refined by a human expert every time a set of rules is found to be inadequate.

The ability to rely on past experience and analogy is also useful when dealing with old technology. Many times, solutions to problems involving new technology are analogous to those involving old technology because most likely they have the same functionalities, but with different performance. Rules can be used to control the case adaptation process and therefore take advantage of the capability to use analogy. For example, the general outline of a fault correction plan for recovering a mixture of old and new switches would be the same as for a plan involving only old technology. The differences would come into play with timing, because the old switches are slower. Rules can account for this by determining what the various timings are and come up with a timing scheme to avoid putting too much pressure on the old switches.

Conclusion

This paper has discussed possible AI technologies that allow for the automation of fault management. Due to the large amount of alarms to process and their inherent ambiguity and uncertainty, a probabilistic AI technology can be used for correlation such as NNs or BBNs. However, additional problem solving methods must be used to further analyze the correlated faults, perform tests, and identify the fault. Methods such as ESs and CBR systems are appropriate due to their ability to handle knowledge rich domains such as diagnosis.

A proposed hybrid system, currently under development, was discussed. Alarms are initially filtered through a rule-based or expert system and correlated with a neural network. The filtered and correlated alarms are then further analyzed through the use of a CBR system that attempts to identify the fault that is the cause of the alarms, based on past cases or problem solving experiences. A second CBR system is used to determine a correction plan, which is then executed and the results analyzed and stored as a new case. We believe the combination of probabilistic and symbolic AI techniques give the fault management system more flexibility and capability for diagnosing faults in today's and tomorrow's large heterogeneous networks.

Acknowledgments

We wish to acknowledge JohnMark Agosta (SRI International) for his input in discussions of the application of Bayesian belief networks to network fault management and Diane Lee (SRI International) for her helpful and insightful review of this paper.

References

Beyer, D. & Ogier, R.G. (1990) Tabu learning for neural network optimization. In *Proceedings of the International Joint Conference on Neural Networks*.

Byrne, C.J. (1994) Fault management. In Aidarous, S. & Plevyak, T. (Eds.) *Telecommunications Network Management into the 21st Century*. New York: IEEE Press.

Corn, P.A., Dube, R., McMichael, A.F., & Tsay, J.L. (1988) An autonomous distributed expert system for switched network maintenance. In *Proceedings of IEEE GLOBECOM'88* (pp. 1530-1537).

Dupuy, A., Schwartz, J., Yemini, Y., Barzilai, G. & Cahana, A. (1989) Network fault management a user's view. In Meandzija, B. & Westcott, J. (Eds.) *Integrated Network Management, I*. North Holland: Elsevier Science Publishers B.V.

Goel, A., Ramanaujam, J. & Sadayappan, P. (1988) Towards a neural architecture for abductive reasoning. In *Proceedings of the IEEE International Conference on Neural Nets*, Vol. I (pp. 681-688).

Jakobson, G. & Weissman, M.D. (1993) Alarm correlation: correlating multiple network alarms improves telecommunications network surveillance and fault management. *IEEE Network*, 52-59, November.

Joseph, C., Kindrick, J. Muralidhar, K. So, C. & Toth-Fejel, T. (1989) MAP fault management expert system. In Meandzija, B. & Westcott, J. (Eds.) *Integrated Network Management, I.* North-Holland: Elsevier Science Publishers B.V.

Patton, R.J., Chen J., & Siew, T.M. (1994) Fault diagnosis in nonlinear dynamic systems via neural networks. In *Proceedings of the International Conference on CONTROL*, Vol. 2 (pp. 1346-1351).

Wang, Z. (1989) Model of network faults. In Meandzija, B. & Westcott, J. (Eds.) *Integrated Network Management, I*. North Holland: Elsevier Science Publishers B.V.

Wright, J.R., Zielinski, J.E. & Horton, E.M. (1988) Expert systems development: the ACE system. In Liebowitz, J. (Ed.) *Expert System Applications to Telecommunications.* New York: John Wiley & Sons.

Yamahira, T., Kiriha, Y. & Sakata, S. (1989) Unified fault management scheme for network troubleshooting expert system. In Meandzija, B. & Westcott, J. (Eds.) *Integrated Network Management, I.* North-Holland: Elsevier Science Publishers B.V.