

Intelligent Distributed Fault Management for Communication Networks

Hongjun Li, John S. Baras
Center for Satellite and Hybrid Communication Networks
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
{ hjli, baras}@isr.umd.edu

Abstract

In this paper, we present an intelligent, distributed fault management system for communication networks using belief networks as fault model and inference engine. The managed network is divided into domains and for each domain, there is an intelligent agent called Domain Diagnostic Agent attached to it, which is responsible for this domain's fault management. Belief network models are embedded in such an agent and under symptoms observation, the posterior probabilities of each candidate fault node being faulty is computed. We define the notion of right diagnosis, describe the diagnosis process based on this concept, and present a strategy for generation of test sequence.

KeywordsFault Diagnosis, Belief networks, Network Management, Fault Management, Sequential Decision Problem, Dynamic Programming

1 Introduction

In a network management system, the role of fault management is to detect, diagnose and correct the possible faults during network operations. Due to the growing number of networks that have served as the critical components in the infrastructure of many organizations, interest in fault management has increased during the past decade, both in academia and in industry [13]. Fault Management is based on three main assumptions [7]: The objective is to deal with malfunctions, not the design faults, of the system, so it is basically a *fault diagnosis* problem, not fault tolerant system design; Tests are more expensive than computations, so it is more favorable to compute and infer the faults rather than brute-force tests; Mis-diagnosis is more expensive than tests, so it is desirable to cover and diagnose as many fault scenarios as possible in a cost efficient manner.

In legacy communication networks, fault diagnosis is often not too difficult since the knowledge of the network manager combined with the alarms reported is usually enough to rapidly locate most failures. But as communication networks evolve, which are expected to be broadband, giant, heterogeneous and complex, things will not be that easy. For example, a single fault can generate a lot of alarms in a variety of domains, with many of them not helpful. Multiple faults will make things even worse. In such cases, it is almost impossible for the network manager, inundated in the ocean of alarms, to correlate the alarms and localize the faults rapidly and correctly just by his/her experience. Therefore, fault management has to be *automated*.

Knowledge-based expert systems, as examples of automated systems, have been very appealing for complex system fault diagnosis [14] and the effort in this field is still growing. Nevertheless, most of the developed expert systems were built in an *ad-hoc* and unstructured manner by simply transferring the human expert knowledge to an automated system. Usually, such systems are based on deterministic network models. A serious problem of using deterministic models is their inability to isolate primary sources of failures from uncoordinated network alarms, which makes automated fault diagnosis a difficult task. Observing that the cause-and-effect relationship between symptoms and possible causes is inherently nondeterministic, *probabilistic* models can be considered to gain a more accurate representation.

A conventional network management system assumes a centralized architecture where all of the monitoring information has to be sent to the central manager for processing. One example is the simple manager-agent paradigm adopted by SNMP [24]. Such a paradigm works well for small networks. But as the networks become larger, the centralized paradigm will incur vast amounts of information communication and thus occupy too much bandwidth unwisely. Since not all the data are relevant and necessary for the manager to process, and there are many cases where the processing can be done on-the-spot, there is no need to centralize all the intelligence at the manager site. In this regard, we propose to distribute some intelligence around the network, by dividing the managed network into domains, and by embedding the network element with some codes for more responsibility.

The embedded code within the network element is called a *delegated agent*. In telecommunications arena, the capability of placing new or added functionality into network element is extremely important, especially as it potentially provides network operators with a mechanism for dynamically updating network elements processing logic. In conventional network monitoring systems, however, the set of services offered by the element agents is fixed and is accessible through interfaces that are statically defined and implemented. This service set can not be modified or extended on-the-fly without recompilation, reinstallation and reinstantiation of the server process on the element agent. To this end, not only would we distribute intelligence via the delegated agents, we would also provide a dynamically extensible interface between such agents and the manager site, such that the manager could change the parameter values, and extend the functionality/processing logic of the delegated agents dynamically. To fulfill all this, we need a more flexible network management framework, by adopting the idea of managing by delegation [10] and the concepts of intelligent mobile agent [6][9][18][27].

In previous research on fault management [1][5][22], the term “fault” was usually taken the same as “failure”, which means component malfunctions, e.g. sensor failures, broken links or software malfunctions. Such faults are called *hard* faults and can be solved by replacing hardware elements or software debugging and/or re-initialization. The diagnosis of the hard faults is called *re-active* diagnosis in the sense that it consists of basically the reactions to the actual failures. In communication networks, however, there are still some other important kinds of faults that need to be considered. For example, the performance of a switch is degrading or there exists congestion on one of the links. Another example is to model faults as deviations from normal behavior [19]. Since there might not be a failure in any of the components, we call such faults *soft* faults. *Soft* faults are in many cases indications of some serious problems and for this reason, the diagnosis of such faults is called *pro-active* diagnosis. By early attention and diagnosis, such pro-active management will sense and prevent disastrous failures and thus can increase the survivability and efficiency of the networks.

In summary, our goal is to come up with an automated, distributed and intelligent fault management system for communication networks, which assumes a *probabilistic* model and integrates the management of both *hard* and *soft* faults. This paper focuses on the fault management aspects; The systems designs of the adaptive, distributed network management framework are discussed in a sister paper [17].

The rest of the paper is organized as follows. In section 2 we briefly describe the main ideas of our proposed system followed by an introduction of belief networks in section 3. In section 4 we describe our belief networks for fault diagnosis, define the notion of right diagnosis and present the general fault diagnosis process. Fault diagnosis strategies, or particularly, node selection schemes are discussed in section 5. Finally, we conclude the paper in section 6.

2 The Proposed System

The managed network is divided into several domains and for each domain, there is an intelligent agent attached to it, which is responsible for this domain’s fault management. A domain is an abstract notion, for example it might be a subnet, a cluster, a host or a member of a functional partition. For those problems that none of the individual agent can solve, there is a mechanism by which the agents can report to a central coordinator and share the information in order to get a global view and solve it cooperatively. So the whole system is, from the agent point of view, a distributed, cooperative multi-agent system. Each agent is called a Domain Diagnostic Agent (DDA) with the goals of monitoring the health of the domain, and diagnosing the faults in a cost-efficient manner. A DDA consists of the following three types of components: Intelligent Monitoring and Detection Assistant (IMDA), Intelligent Domain Trouble-shooting Assistant (IDTA), and Intelligent Communication Assistant (ICA).

The Intelligent Monitoring and Detection Assistants are located at the lowest level and serve to interface with the managed network elements. The inputs are data from network element agents and the output is the activation status on the output nodes, each of which is called a Problem Definition Node (PDN) that represents a certain type of fault. The functions are basically monitoring and fault detection. It will poll the network element for operation information and execute some processing like threshold checking, etc. Alarms are also accepted. In order to decide whether or not there exist fault(s), there must be some form of internal representations of the concerned variables’ expected behavior with which the comparison can be made. Such representations are usually referred to as system models, and they can be set up in various ways, such as AR modeling or neural networks, etc. There are many such IMDAs within a network management system and it is those that are called delegated agents and are to be distributed to the network elements to achieve our goal of intelligent, distributed

management.

The Intelligent Domain Trouble-shooting Assistant resides in the manager station and it includes a probabilistic expert system, which consists of basically a belief network database and decision making modules. The input is the activation status of the problem definition nodes, and the outputs are primary causes and the suggested test sequence. Based on the activation status of the PDNs, a sub-belief network is extracted from the database and then the inference and trouble-shooting begin, as shown in Figure 1. Given the extracted belief network, the beliefs of any non-PDN nodes to be faulty can be calculated, based on which static or dynamic trouble-shooting strategies can be adopted to generate the test sequence. Re-actions are embodied in the handling of the alarms. For pro-actions, however, we have two implications. First, since the “abnormal” PDNs with status other than “alarm” can also be dealt with, the diagnosis afterwards is actually pro-diagnosis in the sense that it is dealing with something before it really goes wrong. Second, the belief network nodes are not restricted to be physical entities, they can also be “logical” or performance nodes, such as “link congestion”, so that “soft” faults can also be included.

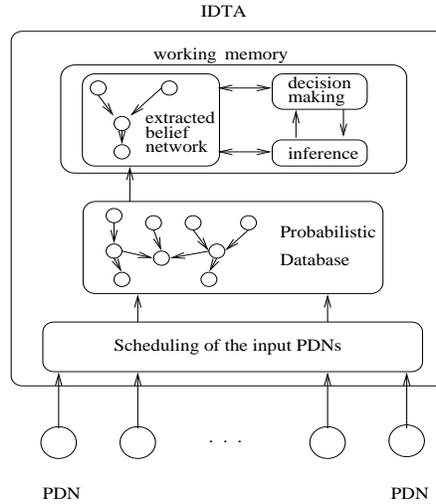


Figure 1. Illustration of an Intelligent Domain Trouble-shooting Assistant

When the problems cannot be solved by an individual DDA, the Intelligent Communication Assistant reports the problems to an upper layer, where, by correlation and coordination, a conclusion can be drawn from a global view. The inputs are results of belief computations and results from test sequences, and the outputs are compressed versions of symptom statistics and of the results given as inputs. The output is then transmitted to a coordinator in the upper layer via some communication links. For details of the proposed system, see [2][16].

3 Belief Network as the Probabilistic Fault Model

DEFINITION 3.1. A belief network is a Directed Acyclic Graph (DAG) in which: The nodes represent variables of interest (propositions); The set of directed links or arrows represent the causal influence among the variables and the parents of a node are all those nodes with arrows pointing to it; The strength of an influence is represented by conditional probabilities attached to each cluster of parent-child nodes in the network.

A belief network can be represented as $\mathcal{B} = (V, L, P)$, where V represents the set of vertices, L represents the set of directed links, and P is the joint probability distribution stored as local conditional probability tables (CPT). As a whole, a belief network represents a joint probability distribution over the interested variables in a compact way due to the embedded conditional independence structures. Belief networks can also serve as the inference engine, and can compute essentially any queries over the variables modeled within the belief network. For example, as new observation (evidence) accumulates, it can provide the updated beliefs (posterior probabilities) of the candidate faults and thus provide us the clue to do further diagnosis. In our research, we use HUGIN expert system API [11]. An example belief network with each node assuming binary value, is shown in figure 2, where the table associated with each node represents the conditional probability distribution, given its parent nodes’ instantiations.

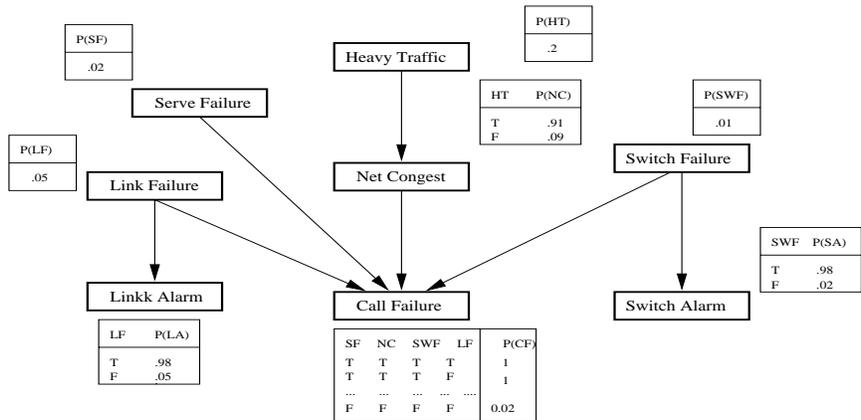


Figure 2. An example Belief Network

Such a representation incorporates our knowledge of the problem domain, and without any observations, the initial marginal probabilities of each node are shown in figure 3.

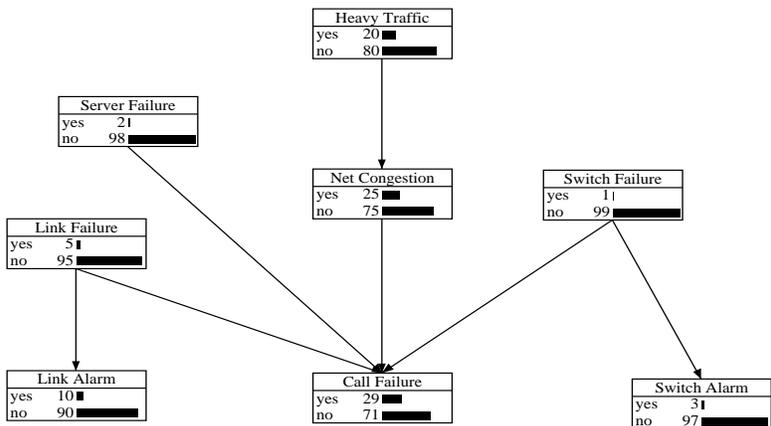


Figure 3. Initial marginal probabilities

Now suppose we observe that there are call failures. We wish to infer the most probable cause for this symptom from the belief network model. To do this, we input this evidence and execute the belief propagation. Figure 4 illustrates the updated beliefs of each non-evidential node after such propagation. Note that for each candidate fault node, namely Link Failure, Server Failure, Heavy Traffic and Switch Failure, the probability of being fault is increased.

If we also observe link alarms, then we hope that this extra information would help locate the most probable fault. After input this evidence, we obtain the updated beliefs again, as shown in figure 5. As we would expect, the evidence of link alarms legitimates the node Link Failure as the most probable fault, and this evidence also "explains away" the other possible candidates, in the sense that the updated beliefs of other candidates is decreased, as compared with those in figure 4.

The above example shows the sketch of doing diagnosis: obtain evidence, update beliefs, obtain evidence again, and so on. One of the aims of this research is to decide which should be the next node to observe. This is rather *activediagnosis* than passive diagnosis in that we are trying to find the most relevant and helpful information on the fly during the diagnosis process. Note that as evidence accumulates, we may input them one by one followed by a propagation right after each evidence-input, as we have shown in this case, or we may input them once altogether and do only one propagation. This provides us the flexibility to do either on-line diagnosis or off-line diagnosis/analysis. For more information on belief networks, we refer to [20] [23].

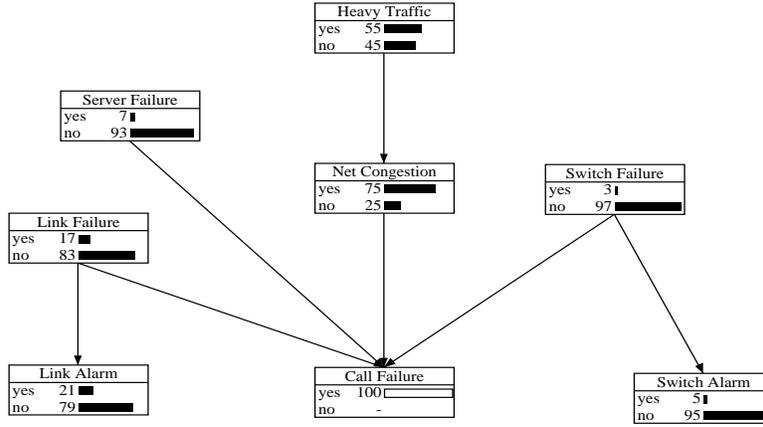


Figure 4. Updated beliefs after observing Call Failure

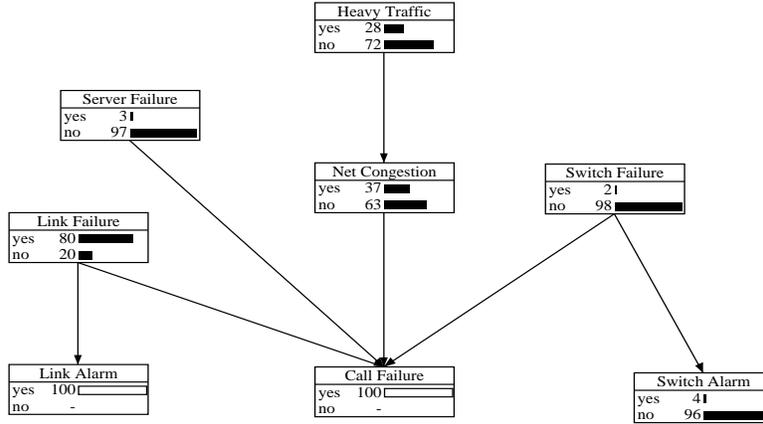


Figure 5. Updated beliefs after observing Call Failure and Link Alarm

4 Fault Diagnosis Problems using Belief Networks

Belief networks models can be built to help fault management for communication networks. Faults can be categorized as either *hard*, if they reflect the failures of some hardware or software components in the network, or *soft*, if what they represent is not failures but some performance degradation or deviation from some “normal” behavior. Faults are usually not directly observable and fault diagnosis is the process of locating the faults based on current observations (symptoms) and possibly further observations.

4.1 Belief Network Nodes Classification

In a communication network environment, probes are attached to some hardware/software components to get operation status. The data returned from the probes are called raw data. Typically the raw data will be grouped into vector form $\mathbf{d} \in \mathbf{R}^n$ and then processed to get an aggregate value that indicates some properties of interest (e.g. average, peak value, etc.).

DEFINITION 4.1. A statistics is a function from \mathbf{R}^n to \mathbf{R} that maps the raw data vector \mathbf{d} to a real number.

Such statistics would usually be computed locally, namely near the network elements or even embedded in them, and further quantified and represented using discrete values. We use value 0 to represent “normal”, and all other values other than 0 represent different level of severity of the statistics.

DEFINITION 4.2. A vertex or node v in a belief network model $\mathcal{B} = (V, L, P)$ is called observable if and only if it represents the health status of a statistics, or corresponds to user report. The set of observable nodes is denoted by O , and obviously $O \subseteq V$. Also define the non-observable set \tilde{O} as $\tilde{O} = V \setminus O$.

These observable nodes correspond to the Problem Definition Nodes (PDN) mentioned in section 2, and we

restrict them to be leaf nodes only, and vice versa. We also restrict that all root nodes are binary valued.

DEFINITION 4.3. The regular evidence set or symptom set R contains those nodes which we observe during normal network monitoring operations. Each $r \in R$ is called a symptom node.

DEFINITION 4.4. The test set ST contains all other observable nodes that are not currently in the regular evidence set, namely $ST = O \setminus R$.

DEFINITION 4.5. The fault set F is the set of root nodes. Obviously $F \subseteq \tilde{O}$. For any $f \in F$, $f = 0$ means not faulty and $f = 1$ means faulty.

DEFINITION 4.6. The hidden node set H contains all nodes in \tilde{O} but not in fault set F , $H = \tilde{O} \setminus F$, or $H = V \setminus (O \cup F)$.

Hidden nodes may be intermediate nodes between faults and symptoms and we don't usually have the interest to put queries on them during diagnosis.

DEFINITION 4.7. The problem domain is said to be working in normal status with respect to regular evidence set R if and only if every node in R takes value 0, or vector $\mathbf{r} = \mathbf{0}$, where $\mathbf{r} = (r_1, r_2, \dots, r_{|R|})$.

DEFINITION 4.8. The problem domain is said to be working in abnormal status with respect to regular evidence set R if and only if there is at least one $r \in R$ whose value is other than "0".

DEFINITION 4.9. The syndrome with respect to regular evidence set R is simply the nonzero vector of values for all elements in R , $\mathbf{r} = (r_1, r_2, \dots, r_{|R|})$, and $\mathbf{r} \neq \mathbf{0}$.

There might be cases when multiple symptom nodes in R take nonzero values. We assume here that any syndrome may trigger the diagnosis process.

4.2 Fault Diagnosis Process

After the fault diagnosis is triggered, the evidence is propagated and the posterior probability of any $f \in F$ being faulty can be calculated through standard belief network inference algorithms. It would be ideal if we can locate the fault with efforts up to this; but most of the time, similar to what happens in medical diagnosis, we need more information to help pinpoint the fault. In our work we postulate the limited observation assumption, meaning that all the observation and tests are constrained within the belief network model.

Perhaps the most fundamental question to ask is: When can I say that I get the right diagnosis? In particular, when can I say that a node in F explains the symptoms? What do we mean by explanation? Further, if one single fault can not explain the symptoms, we are facing the multiple faults problem. Again, when can we say that the nodes we are suspicious of are really the reasons?

Consider what a human would think during diagnosis. After obtaining one possible reason, one would naturally ask, for example, "Would the problematic circuit work normally if I replace this suspicious component with a good one?" He/she then goes ahead and see what happens after the replacement. If the syndrome disappears, one can claim that he/she actually found and trouble-shooted the fault. If the problem domain is tiny, not very complex, and the replacement burden is light, this paradigm would work well. But for communication network environment, the story is totally different. As we mentioned in section 1, we would like to do intelligent diagnosis via *computation* rather than brutal replacement before we are very confident what the fault is.

To do this, we need to distinguish between two kinds of semantics of the instantiation for a node in a belief network: passive observation and active setting. All the instantiations of nodes we've been talking about so far are passive observations, based on which we would infer how our beliefs of other variables would change using belief propagation algorithms. For example we would like to know the consequences of, and the possible causes for such observations, and so on. The alternative semantics is that we can also *set* the value of a node via active experiment. One example would be the above question "Would the problematic circuit work normally if I replace this suspicious component with a good one?" Note that in this situation, external reasons (the human diagnoser in this case) explain why the suspicious component becomes good (because the human diagnoser replace that) and thus all the parent nodes for this node should not count as causes during belief propagation. To see this effect, we could delete all links that point to this node and thus make this node a root node, with its value specified to "normal". Other belief updating like evaluating consequences, however, are not influenced by this active setting since as long as a node takes its value, its non-predecessors can not tell how this value is obtained. This external force is called intervention by Judea pearl in his recent book [21].

With this *set* semantics, we could do *virtual* replacement in our belief network model. For simplicity, we

assume here that the single symptom node is $S1$. For each node in F of belief network \mathcal{B} , we could get the posterior probability of being faulty given $S1 = 1$. Let node f be the most probable faulty node in set F , namely $f = \operatorname{argmax}_{g \in F} P(g = 1 | S1 = 1)$. We would like to see the effects on node $S1$ by setting node f to normal, in other words, we would evaluate $P(S1 = 0 | \text{setting}(f = 0))$. In our belief network model, all the possible fault nodes are root nodes and we do not need to delete any links that point to these root nodes. Obviously, $P(S1 = 0 | \text{setting}(f = 0)) = P(S1 = 0 | \text{observe}(f = 0))$, so in implementation this setting is treated the same as usual evidence input. Other nodes in F are treated as background variables and they keep at the same status as what has been just updated. Note that in belief network \mathcal{B} , the node $S1$ has already been instantiated; while our purpose here is to obtain $P(S1 = 0 | \text{setting}(f = 0))$. It would be messy to carry out intervention on belief network \mathcal{B} itself, since we would have to retract the evidence of $S1 = 1$ before intervention and then restore this evidence after it for further diagnosis. In this paper, we introduce the so-called *intervention belief network* $\tilde{\mathcal{B}} = (V, L, P, S, Fs)$ to help this virtual replacement.

DEFINITION 4.10. An *intervention belief network* $\tilde{\mathcal{B}} = (V, L, P, S, Fs)$ is obtained from the original belief network $\mathcal{B} = (V, L, P)$ with the same V, L, P . S is the symptom set and $Fs \in F$ is the set of suspicious nodes. We would compute for each $s \in S$ the probability $P(s = 0 | \text{setting}(Fs_i = 0), \forall i \in [1, |Fs|])$ using $\tilde{\mathcal{B}}$.

Here is the virtual replacement procedure for our particular example above:

- Step1: In belief network $\mathcal{B} = (V, L, P)$, for each node f_i in F , update the probability $p_i \triangleq P(f_i = 1 | S1 = 1)$
- Step2: Suppose that $f_1 = \operatorname{argmax}_{g \in F} P(g = 1 | S1 = 1)$
- Step3: In intervention belief network $\tilde{\mathcal{B}} = (V, L, P, S1, f)$, set node $f_1 = 0$, and with $P(f_i = 1) = p_i, i = 2, \dots, |F|$, compute $P(S1 = 0 | \text{setting}(f = 0))$.

To determine whether or not this virtual replacement has led $S1$ to an acceptable status, we need a reference value for the computed $P(S1 = 0 | \text{setting}(f = 0))$ to compare with for such decision. Without any evidence input, the belief network model \mathcal{B} itself gives the probability of each leaf node to be normal. We could use these values as the reference.

DEFINITION 4.11. Given a small number ϵ , we say that node $S1$ becomes ϵ -normal via intervention if and only if $P(S1 = 0) - P(S1 = 0 | \text{setting}(f = 0)) < \epsilon$,

In the above example, we assume one symptom node and one suspicious node. More generally, we have,

DEFINITION 4.12. A nonempty faulty set Fs is called the explanation or right diagnosis of the initial syndrome r if and only if every node in set R becomes ϵ -normal if we set every node in Fs to normal in the intervention belief network $\tilde{\mathcal{B}} = (V, L, P, R, Fs)$.

It is when set Fs explains the symptom set R that we terminate the diagnosis process. In diagnosis, however, it is always an issue how many suspicious nodes we should choose from the fault set F . In our belief network model and the parallel intervention model, we should be discrete in choosing multiple nodes, since, if we simply choose all nodes in F and do the intervention, the symptom nodes would all of course become ϵ -normal. But clearly, calling every node in F as faulty is not acceptable; One of the most important goals of fault diagnosis in general is to bias among the all possible faults and locate the real one(s)! In our work, we sort all the nodes in F according to their probabilities of being faulty in a descending order, and only choose the first a couple of nodes as suspicious nodes to be input to the intervention belief network. In particular, if we let $p_i, i = 1, \dots, |F|$ denotes the sorted probabilities of the nodes in F being faulty, with $p_1 \geq p_2 \geq \dots \geq p_{|F|}$, we only choose the first j nodes such that

$$\frac{\sum_{k=1}^j p_k}{\sum_{k=1}^{|F|} p_k} \geq \text{PERCENTAGE_GURAD}, \quad (1)$$

where PERCENTAGE_GURAD is a real number between 0 and 1. It should not be close to 1, since in that case, we would have to choose almost all nodes in F . In our work, we choose 0.4 initially and we are testing on one single suspicious node on all symptom scenarios. If no single fault would explain the current symptoms, we increase PERCENTAGE_GURAD by a small amount and do the diagnosis again.

As discussed above, we would do intervention on those suspicious nodes. If they explain the symptoms, the diagnosis stops; otherwise, we would have to determine a new node to test in order to get more information. Since we can have at most $|ST|$ tests, and if we choose to test them one by one, the time step set is defined as $T = \{1, 2, \dots, |ST|\}$.

DEFINITION 4.13. The active evidence set AE contains the nodes that are instantiated during the process of diagnosis.

Set AE keeps a record of currently instantiated nodes. Initially, $AE = R$. As diagnosis proceeds, more and more test nodes in ST are selected, tested, and then added into AE . Such already chosen test nodes are not to be chosen again for future use.

DEFINITION 4.14. The candidate test set C_{st} contains the nodes in ST that are available and yet to be chosen and tested.

Initially, $C_{st} = ST$. As diagnosis proceeds, C_{st} shrinks as instantiated nodes are removed from it.

DEFINITION 4.15. The action set A is defined as $A = C_{st} \cup \{STOP\}$, which states that during diagnosis, the action $a(t)$ we can take at time step t is either to choose a node to test, or just stop there.

DEFINITION 4.16. There is an immediate cost incurred by selecting an action $a \in A$ during diagnosis. If $a = STOP$, the cost is zero. For any $s_i \in ST$, cost function is $C(s_i, t)$, where t is the time step.

Cost function reflects expense of a test, which is cumulative from many factors, like the difficulty, time to be consumed, severity, etc. The determination of such a cost function entails careful deliberation about what factors to consider, to what extent, and how they should be combined. We assume in our framework that the cost function is homogeneous, namely, cost function is of form $C(s_i)$. This is usually the case in that the cost is normally associated with the test itself only, and the test itself does not usually change with time. Our goal of fault diagnosis is to locate the right diagnosis with minimum diagnosing steps and optimum costs:

$$\min_{a_1, \dots, a_{|ST|}} J = \sum_{k=1}^{ST} c(a_k) + f(a_k, k) \quad (2)$$

$c(a_k)$ is the immediate cost associated with action a_k at time step k , $c(STOP) = 0$, and if $a_K = STOP$ at some certain step K , all $a_j = STOP$ for $j > K$. $f(a_k, k)$ is the measure for diagnosing steps, and we define it as

$$f(a_k, k) = \begin{cases} 0 & \text{if } a_k = STOP \\ g(k) & \text{otherwise} \end{cases} \quad (3)$$

where $g(k)$ is a function of current step k . We simply take $g(k) = 1$ for all k in our simulation.

Using the same ideas of choosing suspicious nodes set F_s and executing intervention, various diagnosis strategies differ only in how the action, or the next test node, would be chosen. We would discuss different strategies in section 5. Other than that, our fault diagnosis process could be summarized as follows.

- Step 1. Initialization
 - For belief network model $\mathcal{B} = (V, L, P)$, make a copy and call it $\tilde{\mathcal{B}}_t$.
 - Set time step $tp = 0$.
 - Initialize active evidence set $AE = R$, candidate test set $C_{st} = ST$.
 - Input evidence by setting the nodes in set AE according to current active evidence values ae .
- Step 2. Belief Propagation in belief network \mathcal{B}
 - Execute standard propagation.
 - For each node f_i in set F , obtain the posterior probability $P(f_i = 1 | AE = ae)$.
 - Sort the set F in decreasing order according to the probability of being faulty.
- Step 3. Intervention
 - Get the set F_s of suspicious nodes that satisfy (1)
 - Set the root nodes in intervention belief network $\tilde{\mathcal{B}} = (V, L, P, R, F_s)$ accordingly, and execute the intervention
 - * If by intervention, F_s explains R , update total cost and TERMINATE.
 - * Else, goto Step 4

- Step 4. Get next testing node
 - If $C_{st} = \Phi$, empty set
 - * Update total cost
 - * Give out the set F_s and say "Didn't find the right diagnosis, but here is the list of possible faults in decreasing order".
 - Else Get node st according to some node selection scheme.
- Step 5. Observing test node st and get observation Z_{st}
 - Input this evidence $st = Z_{st}$ to original belief network \mathcal{B} .
 - Update time step, $tp = tp + 1$.
 - Update the candidate test set $C_{st} = C_{st} - st$
 - Update the active evidence set $AE = AE + a(tp)$
 - Goto Step 2.

5 Fault Diagnosis Strategies

As shown in (2), our problem here is to minimize J over the possible test sequence $(a_1, a_2, \dots, a_{|ST|})$. Let a_t denotes the action chosen from set A at time step t , $t \leq |ST|$, and Z_{a_t} denotes the value obtained by observing a_t , for example $Z_{a_t} = 0$, for $a_t \neq STOP$.

DEFINITION 5.1. We define the history process up to time step k as

$$I_k = (Z_0, (a_1, Z_{a_1}), \dots, (a_k, Z_{a_k})),$$

where $Z_0 = ((r_1, Z_{r_1}), (r_2, Z_{r_2}), \dots, (r_{|R|}, Z_{r_{|R|}}))$ the evidence set and corresponding instantiations.

Z_0 summarizes the history up to the time diagnosis is triggered. I_k grows along with diagnosis and obviously we have $I_k = (I_{k-1}, (a_k, Z_{a_k}))$ and we call this history process *Markov*. Since at time step k , the history process I_k contains all the necessary information about the current status of the belief network model, we can simply take the history process I_k as the *state* at time step k . So equivalently, our problem here is to create the optimal state trajectory such that J is minimized. Before describing our diagnosis algorithm, let us first see a similar but different situation for intuition.

5.1 c/P Algorithm for Trouble-shooting Single Fault

Suppose we are given a similar yet different problem where the concern is to locate the single faulty component. There are symptoms indicating the malfunction (e.g. car doesn't start) and for each possible faulty component there is a *direct* test associated with it (e.g. testing whether or not the fan belt is OK), which tells whether or not this component is working well. The cost for testing component i is c_i . Based on the symptoms, we obtain P_i , the probability that component i is in failure, for every component. We are supposed to test those components one at a time. As soon as one component fails its associated test, we claim that we find the single fault and stop. There are many ways to choose the order of testing, so the question is: In what order should we choose tests to minimize the expected accumulated cost?

For a test sequence $\{1, 2, \dots, j, k, \dots, n\}$ with $k = j + 1$, and n as the number of components to be tested, the probability that the j th candidate have to be tested is the probability that none of its predecessors have failed the tests, namely $1 - \sum_{i=1}^{j-1} P_i$, or $\sum_{i=j}^n P_i$, the probability that the faulty node is either the j th candidate or its successors. So the expected cost is:

$$EC_1 = c_1 + c_2 \sum_{i=2}^n P_i + \dots + c_j \sum_{i=j}^n P_i + c_k \sum_{i=k}^n P_i + \dots + c_n P_n$$

If we exchange k and j , then we get another test sequence $\{1, 2, \dots, k, j, \dots, n\}$, for which the expected cost is:

$$EC_1 = c_1 + c_2 \sum_{i=2}^n P_i + \dots + c_k \left[\sum_{i=k}^n P_i + P_j \right] + c_j \left[\sum_{i=j}^n P_i - P_k \right] + \dots + c_n P_n$$

The difference between the above costs is: $EC_1 - EC_2 = c_j P_k - c_k P_j$, and it is straightforward that

$$EC_1 \leq EC_2 \Leftrightarrow c_j P_k \leq c_k P_j \Leftrightarrow c_j / P_j \leq c_k / P_k, \forall P_i > 0$$

We see that EC_1 is cheaper than EC_2 if and only if the c/P value for candidate j is less than that for candidate k . Thus, any strategy with an element that has higher c/P value than its successor can be improved upon by simply exchanging the two elements. So for an optimal strategy, all elements must be in non-decreasing sequence of c/P values [12].

5.2 Heuristic Diagnosis Strategy for Our Formulation

The problem used to derive the c/P algorithm is different from our formulation in the following aspects: 1. It tackles hard faults or failures while our problem integrates both hard and soft faults. 2. It assumes that for each candidate fault there is a *direct* test that tells the status of that component while we don't have that luxury. For a communication network environment which is distributed, complex and heterogeneous, it is impossible to predefine and store a corresponding test for each possible cause. Actually one of the goals of our problem is to generate *dynamically* the test sequence on the fly, based on our belief network model and current syndrome. This entails cycles of test node selection, observation of the selected test and belief updating, until we find the right diagnosis or reach the time step limit. 3. In this problem, we go ahead and test the candidate directly to see whether or not it's **the** fault. But in our formulation, right diagnosis is defined via the introduction of the concept of external intervention and determined through computation. 4. Finally, our algorithm should be able to tackle multiple faults diagnosis.

But the c/P algorithm does provide insight in that it reflects the following observation: in order to minimize the total cost, people are more likely to try those more fault-prone, cheaper components before the less-probable, expensive ones. In other words, people prefer more relevant and cheaper tests.

In our diagnosis algorithm, we look at the suspicious node set F_s and wish to find an appropriate test node st if F_s could not explain the evidence set R . Similar to the c/P algorithm ideas, we would like to choose the test node from candidate test set C_{st} that is most relevant to F_s and cheapest at the same time. To achieve this, we need a measure for relevancy between a test node in C_{st} and a fault node in F . The definition of relevancy requires the concepts of entropy and mutual information, which we copy from [8] as follows.

DEFINITION 5.2. The entropy $H(X)$ of a discrete random variable X is defined by

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

The entropy of a random variable is a measure of the uncertainty of the random variable; it is a measure of the amount of information required on the average to describe the random variable.

DEFINITION 5.3. Consider two random variables X and Y with a joint probability mass distribution $p(x, y)$ and marginal probability mass distributions $p(x)$ and $p(y)$. The *mutual information* $I(X; Y)$ is defined as

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

Mutual information is a measure of the amount of information that one random variable contains about another random variable. It is the reduction in the uncertainty of one random variable due to the knowledge of the other. If X and Y are independent, $I(X; Y) = 0$ meaning that knowing one random variable provides no information of the other. If X and Y are actually equivalent (e.g. isomorphic), we denote $X \equiv Y$, and $I(X; Y) = H(X) = H(Y)$.

DEFINITION 5.4. The relevancy of random variable Y relative to random variable X is defined as

$$R(X; Y) = \frac{I(X; Y)}{H(X)}$$

$R(X; Y) \in [0, 1]$ indicates to what extent Y can provide information about X . $R(X; Y) = 1$ means that Y can uniquely determine X , while $R(X; Y) = 0$ indicates that Y and X are independent. Note that $R(X; Y) \neq R(Y; X)$. More generally,

DEFINITION 5.5. The relevancy of random variable Y relative to a set of random variables \mathbf{X} is defined as

$$R(\mathbf{X}; Y) = \frac{I(\mathbf{X}; Y)}{H(\mathbf{X})}$$

where $I(\mathbf{X}; Y)$ and $H(\mathbf{X})$ are defined similarly as above.

With the relevancy definition, our next test node is simply

$$st = \operatorname{argmax}_{g \in C_{st}} R(\mathbf{F}\mathbf{s}; g) / c(g)$$

5.3 Proof of Correctness Simulation

The belief network example in section 3 is over simplistic without many test nodes for us to choose from. To illustrate the effectiveness of our fault diagnosis algorithm, consider the example network shown in figure 6.

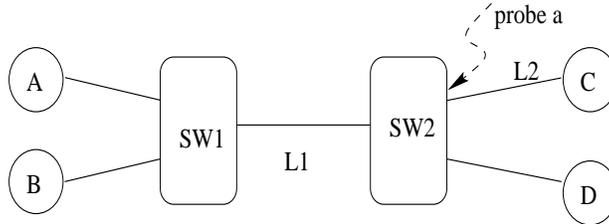


Figure 6. Example Network

Two switches $SW1$ and $SW2$ are connected to each other via link $L1$. Machines A and B are connected to $SW1$ and they would communicate with machines C and D , which are connected to switch $SW2$. We have a probe a hooked at the end of $SW2$ to measure the traffic throughput going out of $SW2$. Suppose the information we could obtain during network operation include whether or not: $SW1$ alarm is normal, A could connect $SW2$, B could connect $SW2$, A could connect C , C could connect $SW1$, throughput at probe a is normal, and D could connect $SW1$. The possible faults are identified as: $SW1$ works normal or not, $L1$ normal or congested, $SW2$ normal or not, and source pumped from C to $L2$ is normal or not. We set up a belief network model for such situations, and figure 7 shows the structure and initial probability distributions.

As a comparison to our node selection scheme, we use the random node selection scheme meaning that each time we need a test node, we simply choose uniformly one node from all current available nodes in C_{st} . In our simulation, the outcome of chosen test node st is uniformly generated as either 0 or 1. The costs for testing each leaf node is shown in Table 1, with 40 as the penalty for not being able to find the right diagnosis. To save space, we only list the diagnosis for the following symptoms: A_Conn_SW1 problem, A_Conn_C problem, and A_Conn_SW1 and A_Conn_C problems happen at the same time. Table 2 shows the comparison of the two test generation schemes with 2000 runs. We can see that node selection via relevancy is much better than that via random selection.

Table 1. Cost for All Leaf Nodes

SW1_Indicator	A_Conn_SW2	B_Conn_SW2	A_Conn_C	Thru_Prob_A	C_Conn_SW1	D_Conn_SW1
2	1	7	1	3	1	3

The above algorithm is heuristic and lack of rigorous analysis. We could model such diagnosis as a Markov Decision Problem and dynamic programming techniques could be exploited, which comes next.

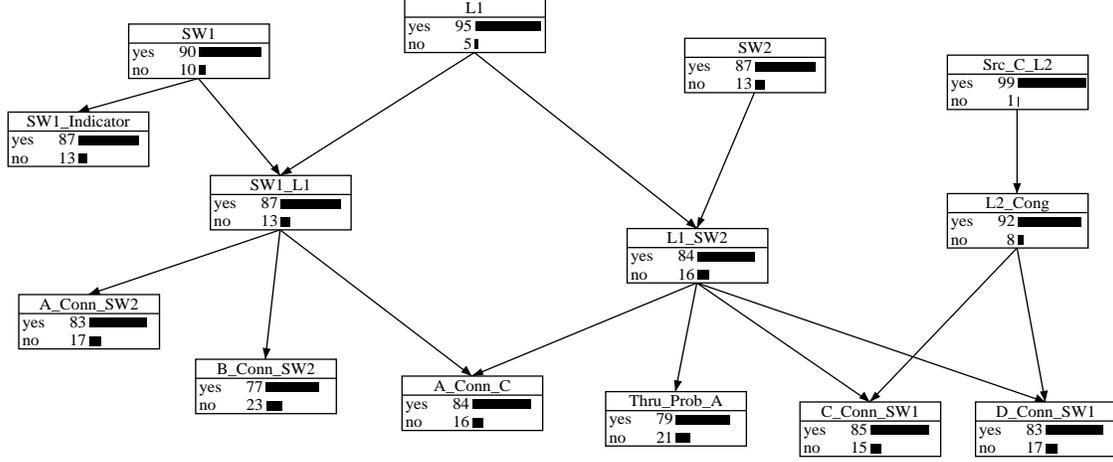


Figure 7. Example Network

Table 2. Comparison of Node Selection Schemes

Symptom Nodes	Random Selection		Relevancy Selection	
	Avg. Cost	Success Rate	Avg. Cost	Success Rate
A_Conn_SW2	15.38	84.5%	9.13	94%
A_Conn_C	26.21	70.1%	14.22	88%
A_Conn_SW1 and A_Conn_C	24.68	67.8%	3	100%

5.4 Dynamic Programming Formulation

For the sequential decision problem with state process as defined above, we have a discrete-time dynamic system and the cost function is additive over time. let $N = |ST|$ be the total time step. At time step k , the immediate cost $g(i, u, j)$, incurred by leaving state i for state j when control $u = a_k$ is chosen, is simply $\tilde{c}(a_k, k)$, with

$$\tilde{c}(a_k, k) = \begin{cases} 0 & \text{if } a_k = STOP \\ c(a_k) + g(k) & \text{otherwise} \end{cases} \quad (4)$$

As discussed in section 4.2, the diagnosis process would terminate if the suspicious node set F_s could explain the symptom set R . When terminated at time step L , namely $u_L = STOP$, the resulting cost is $T + \sum_{l=0}^L \tilde{c}(a_l, l)$, where T is a termination cost. If the process has not terminated up to the final time N , the resulting cost is $g_N(I_N) + \sum_{l=0}^{N-1} \tilde{c}(a_l, l)$, where $g_N(I_N)$ is the terminal cost incurred at the end of the process. Here, $g_N(I_N)$ is a penalty for not being able to find the right diagnosis.

Suppose up to time step k , there is no STOP action yet, and $I_k = i_k$. Notice that the state i_k is equivalent to (AE_k, ae_k) , where AE_k and ae_k are the active evidence set and the corresponding values at time step k . And, the admissible control set $U_k(i_k) = A_k$, with A_k as the action set at time k . Assume also that all nodes in our belief network model is binary-valued. The state transition from I_k to I_{k+1} depends on a control $u \in U_k(i_k)$. If $u = STOP$, $I_{k+1} = (i_k, STOP)$, and the process terminates. Equivalently, we could set $U_l(I_l) = \Phi$ for all $l > k$ and thus the state process could go nowhere else but stay at $(i_k, STOP)$. If the current F_s could not explain the symptom set R , we could get $P_u^k \triangleq P(u = 1 | I_k)$ for a particular control $u \in C_{st}$ at time k . Let $i = I_k$ and $j = I_{k+1}$ for simplicity. The next state $j = (i, (u, Z_u))$ and the corresponding state transition probability $P_{ij}(u)$ may be either:

- $j = j_1$, if node u is faulty, $Z_u = 1$, and $P_{ij_1}(u) = P_u^k$; Or,
- $j = j_0$, if node u is not faulty, $Z_u = 0$, and $P_{ij_0}(u) = 1 - P_u^k$.

To model the diagnosis process in terms of a finite horizon problem, we need to augment the state space with a special termination state. Define t_k as the termination state at the beginning of time k , with $t_k = 0$ meaning terminated and $t_k = 1$ meaning to continue. Initially, $t_0 = 1$. The evolution follows: $t_{k+1} = h_k(t_k, u_k)$, $\forall k =$

$0, \dots, N-2$, with

$$h_k(t_k, u_k) = \begin{cases} 0 & \text{if } u_k = STOP, \text{ or } t_k = 0 \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

So by defining new state process $X_k = [I_k, t_k]$, we have the new state transitions $X_{k+1} = f_k(X_k, u_k)$, where

$$f_k(X_k, u_k) = \begin{cases} X_k & \text{if } t_k = 0 \\ [I_k, 0] & \text{if } t_k \neq 0, u_k = STOP \\ [(I_k, (u, 1)), 1] & \text{with prob. } P_u^k \text{ if } t_k \neq 0, u_k = u \in C_{st} \\ [(I_k, (u, 0)), 1] & \text{with prob. } 1 - P_u^k \text{ if } t_k \neq 0, u_k = u \in C_{st} \end{cases} \quad (6)$$

At time N , set $t_N = 0$. It is easy to show that $f_k(X_k, u_k)$ is equivalent to a transition probability distribution $P_{X_k X_{k+1}}(u_k)$.

The cost function is then defined by

$$\tilde{c}(X_k, u_k, k) = t_k \tilde{c}(u_k, k),$$

and the terminal cost is $\tilde{g}_N(X_N) = t_{N-1}g_N(I_N) + (1 - t_{N-1})T$. So we formulate the problem as

$$\min_{u_k, k=0, \dots, N-1} E \left\{ \tilde{g}_N(X_N) + \sum_{l=0}^{N-1} \tilde{c}(X_k, u_k, k) \right\} \quad (7)$$

which is the basic form of finite horizon problem [3]. We define $J_k^*(i)$ as the minimum cost when starting from state i and k steps remain, or the best k -step cost-to-go, $\forall k \leq N$. Obviously, $J_0^*(i) = \tilde{g}_N(X_N)$, and by principle of optimality,

$$J_k^*(i) = \min_{u \in U(i)} \sum_j P_{ij}(u) [\tilde{c}(i, u, k) + \alpha J_{k-1}^*(j)], \forall k = 1, \dots, N,$$

where i represents x_k , j represents x_{k+1} , and $P_{ij}(u)$ is just the $P_{X_k X_{k+1}}(u_k)$ obtained from (6). After we get $J_N^*(i)$, we could get the control sequences that achieve the minimum, and the solution for problem (7) is an *optimal policy* $\pi^* = \{\mu_1^*, \dots, \mu_N^*\}$, where μ_k^* is a mapping from state to action for each $k \in T$, that minimizes the expected total costs. Ideally, we wish to obtain closed form solutions by dynamic programming techniques, but this rarely happens in practice. So people turn to find approximate values of the J -functions [3] [3]. We would use Q -learning technique [25] [26] to obtain the approximations.

At each state i , instead of using J -function which is averaged over all actions, we define a Q -function for state-action pair i and a as $Q(i, a)$. $Q(i, a)$ represents the cost incurred by action a when at state i . If we know for all states and all actions the Q -values, we could easily obtain an optimal policy by choosing the best action at each state. Q -learning is a technique to obtain approximations for Q -values, and the update formula of Q -learning is

$$Q(x_t, a_t) \leftarrow (1 - \gamma)Q(x_t, a_t) + \gamma \left(\tilde{c}(x_k, a_k, k) + \alpha \min_a Q(x_{t+1}, a) \right)$$

where γ is the learning rate and $alpha$ is the discount factor. It has been proved that if each action is executed in each state an infinite number of times, and γ is decayed, the Q -values will converge to Q^* , by which an optimal policy can be acquired.

6 Conclusions

In this paper, we present an intelligent, distributed fault management system for communication networks using belief networks as fault model and inference engine. The managed network is divided into domains and for each domain, there is an intelligent agent called Domain Diagnostic Agent attached to it, which is responsible for this domain's fault management. Belief network models are embedded in such an agent and under symptoms observation, the posterior probabilities of each candidate fault node being faulty is computed. We choose suspicious nodes in a discrete way and call them right diagnosis if they could explain the symptoms. The diagnosis process is described based on this concept, and we illustrate our test generation scheme by using the notion of relevancy. Simulation shows that this scheme is much superior than a random selection scheme. Finally we formulate the diagnosis process as a finite horizon sequential decision problem and our future work includes finding solutions with more rigor using Q -learning.

References

- [1] J. S. Baras, M. Ball, S. Gupta, P. Viswanathan and P. Shah, "Automated Network Fault Management", *MILCOM'97*, Monterey, CA, November 2-5, 1997
- [2] J. S. Baras, H. Li and G. Mykoniatis, "Integrated, Distributed Fault Management for Communication Networks", Technical Report, CSHCN TR 98-10, University of Maryland, 1998
- [3] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I and II*, Athena Scientific, Belmont, MA, 1995
- [4] D. P. Bertsekas, and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996
- [5] A. Bouloutas, G. Hart, and M. Schwartz, "On the Design of Observers for Fault Detection in Communication Networks", *Network Management and Control*, Plenum Press, New York, 1990
- [6] M. Cheikhrouhou, P. Conti, J. Iabetoulle, K. Marcus, "Intelligent Agents for Network Management: a Fault Detection Experiment", in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, Boston, MA, May 1999
- [7] J. Chow and J. Rushby, "Model-Based Reconfiguration: Diagnosis and Recovery", Tech. Rep. 4596, NASA Contractor Report, 1994
- [8] T. M. Cover, and J. A. Thomas, *Elements of Information Theory*, Wiley Interscience, 1991
- [9] M. El-Darieby, A. Bieszczad, "Intelligent Mobile Agents: Towards Network Fault Management Automation", in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, Boston, MA, May 1999
- [10] J. Goldszmidt, Y. Yemini, "Distributed Management by Delegation", in *Proceedings of 15th International Conference on Distributed Computing Systems*, 1995
- [11] <http://www.hugin.dk>
- [12] J. Kalagnanam and M. Henrion, "A Comparison of Decision Analysis and Expert Rules for Sequential Diagnosis", in *Uncertainty in Artificial Intelligence 4*, (Amsterdam, The Netherlands), pp. 271-281, Elsevier Science Publishers B. V., 1990
- [13] I. Katzela, *Fault Diagnosis in Telecommunication Networks*, Ph.D. Thesis, Columbia University, 1996
- [14] L. Kerschberg, R. Baum, A. Waisanen, I. Huang and J. Yoon, "Managing Faults in Telecommunications Networks: A Taxonomy to Knowledge-Based Approaches", *IEEE*, pp. 779-784, 1991
- [15] H. Li, "Statistical parameter learning for Belief networks with fixed structure", Technical Report, CHSCN, University of Maryland, 1998
- [16] H. Li, J. S. Baras and G. Mykoniatis, "An Automated, Distributed, Intelligent Fault Management System for Communication Networks", ATIRP'99, 2-4 February, 1999
- [17] H. Li, S. Yang, H. Xi, and J. S. Baras, "Systems Designs for Adaptive, Distributed Network Monitoring and Control", submitted to IM01, Seattle, 2001
- [18] T. Magedanz, "Intelligent Mobile Agents in Telecommunication Environments - Basics, Standards, Products, Applications", Tutorial for International Symposium on Integrated Network Management VI, Boston, MA, May 1999
- [19] R. Maxion, "A case study of ethernet anomalies in a distributed computing environment", *IEEE Trans. on Reliability*, 39(4), Oct 1990
- [20] J. Pearl, *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988
- [21] J. Pearl, *Causality*, Cambridge Press, 2000
- [22] I. Rouvellou, and G. W. Hart, "Automatic alarm correlation for fault identification", In *Proc. IEEE INFOCOM*, page 553-561, 1995
- [23] S. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach*, Prentice-Hall, 1995
- [24] W. Stallings, *SNMP, SNMPv2 and CMIP: the practical guide to network management standards*, Addison-Wesley, Reading, Mass., 1993
- [25] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998
- [26] C.J.C.H. Watkins, P. Dayan, "Q-learning", *Machine Learning*, 8, pp. 279-292, 1992
- [27] M. Wooldridge, N. R. Jennings, "Intelligent Agents: Theory and Practice", submitted to *Knowledge Engineering Review*, 1995