Automatic Alarm Correlation for Fault Identification *

Isabelle Rouvellou† and George W. Hart
Columbia University

Abstract

In communication networks, a large number of alarms exist to signal any abnormal behavior of the network. As network faults typically result in a number of alarms, correlating these different alarms and identifying their source is a major problem in fault management. The alarm correlation problem is of major practical significance. Alarms that have not been correlated may not only lead to significant misdirected efforts, based on insufficient information, but may cause multiple corrective actions (possibly contradictory) as each alert is handled independently. This paper proposes a general framework to solve the alarm correlation problem. We introduce a new model for faults and alarms based on probabilistic finite state machines. We propose two algorithms. The first one acquires the fault models starting from possibly incomplete and incorrect data. The second one correlates alarms in the presence of multiple faults and noisy information. Both algorithms have polynomial time complexity, use an extension of the Viterbi algorithm to deal with the corrupted data, and can be implemented in hardware. As an example, they are applied to analyze faults using data generated by the ANS (Advanced Network and Services, Inc.)/NSF T3 network.

1 Introduction

In large and complex communication networks, faults are unavoidable, but quick detection and identification can significantly improve network reliability. A network fault typically results in a number of alarms, correlating these different alarms to identify their source is a major problem in fault management. This problem, often referred to as alarm-correlation, is of major practical significance. Rule-based expert systems have been so far the main approach to solving the alarm correlation problem [1, 2, 3, 4, 5, 6]. Although the effort in this area is still growing, most of the existing fault management expert systems were built on an ad-hoc basis, transferring the knowledge of a human expert into an automated system. Recently, other approaches based on more formal methodologies were proposed. The work in [7, 8] is based on a formal language representation of the communication system. A. Bouloutas in [7] focuses on identifying errors in a known protocol: it is not alarm correlation as such. The problem considered by A. Bouloutas and S. Calo in [8] is fault localization from alarms. It is a related although different problem. In particular, it does not consider the time factor (i.e., the order in which the alarms appear does not matter) and assumes knowledge of the network topology. In [9, 10], the authors present a probabilistic theory of diagnosis given a set of manifestations (alarms). Their approach differs from ours as they do not consider noisy data. One of their assumption is the "Mandatory Causation Assumption" which rules out any possibility of unreliable alarms (i.e., alarms which may not be due to a real fault).

Our approach is novel in the fault model it is using and in the algorithms which are developed. It allows us to automatically recognize time patterns of alarms associated to a given fault and to identify possibly simultaneous faults in a noisy environment. It does not assume any knowledge of the network structure such as its topology for example. It is adaptive acquiring the fault models from possibly incomplete/incorrect data.

We model each fault as a probabilistic finite state machine (PFSM). The intuition behind the PFSM model is that faults often result in a set of possible alarm sequences. This can be explained by two main phenomena. As the ramifications of the fault spread out away from the fault initial location, we will in general observe a sequence of alarms (as opposed to a single alarm or a set of alarms in which order does not matter). Furthermore, a fault may in general result in several possible sequence as opposed to a single one (e.g., depending on the state of the network when the fault occurs, the sequence generated may be slightly different). PFSMs are excellent models for flexibly describing sets of sequences.

Our approach then divides in two phases. The first phase acquires the right PFSM model for each fault. The PFSM is built from history data which associate sequences of alarms to a fault occurrence (sequences may be corrupted and/or incomplete). The data used for this phase could come from the network itself, from an expert operator, from simulation, or could be provided by an expert system. Thus our system is not exclusive with expert systems (e.g., expert system might provide initial training data, but be too slow for on-line use, or it could turn out that expert systems are suitable for detecting certain types of faults, while PFSMs are better for others). The second phase, given the fault models, correlates alarms "on-line" and identifies the fault(s) at the origin of the observed alarms. The on-line data is in general an interleaving of alarm sequences resulting from a number of faults. Again this data may be noisy and incomplete. Note that the fault models can be updated each time new data are observed and new diagnoses made. The system learns and gets better.

The algorithms proposed for each phase of our approach have polynomial time complexity, and use an extension of the Viterbi algorithm [18, 17] to process the corrupted data. They

*Portions of this work were presented at ORSA/TIMS'93, Chicago.

†Now with IBM T.J. Watson Research Center

0743-166X/95 $04.00 © 1995 IEEE

are applied to data from the ANS/NFS T3 network. PFSM models are generated for a set of faults including a bug in an implementation of the TCP protocol, route flapping, and a problem with a T3 interface card. Data resulting from the simultaneous occurrence of several of those faults is then analyzed, and the sources of the alarms successfully identified.

The paper proceeds as follows. Section 2 gives some definitions. Section 3 presents the model used for network faults and alarms. Section 4 and 5 respectively describe the first and second phase of our approach. Section 6 shows how our approach was used to analyze faults in the ANS/NFS network.

2 Definitions

Definition 1 *A PFSM F is defined as a sextuple $(Q(F), L(F), A(F), P^{\circ}(F), P^{f}(F), P(F))$.*

1. $Q(F) = \{Q_1, Q_2, ..., Q_n\}$ *is a finite set of states.*
2. $L(F) = \{s_1, ..., s_m\}$ *is a finite set of labels (i.e., outputs associated to transitions between states).*
3. $A(F)$ *is a finite set of arcs (labeled transitions):* $A(F) \subseteq Q(F) \times Q(F) \times L(F)$. Q_i, Q_j *and* s *are respectively called the origin, destination and label of the arc* (Q_i, Q_j, s).
4. $P^{\circ} = (p^{\circ}_{Q_1}, p^{\circ}_{Q_2}, ..., p^{\circ}_{Q_n})$ *is the initial state distribution:*

$$\sum_{Q_j \in Q(F)} p^{\circ}_{Q_j} = 1 \quad and \quad p^{\circ}_{Q_j} \geq 0 \quad \forall j$$

5. $P^{f} = (p^{f}_{Q_1}, p^{f}_{Q_2}, ..., p^{f}_{Q_n})$ *is the final state distribution:*

$$\sum_{Q_j \in Q(F)} p^{f}_{Q_j} = 1 \quad and \quad p^{f}_{Q_j} \geq 0 \quad \forall j$$

6. $P = \{p_{Q_i Q_j s}\}$ *is a table, where* $p_{Q_i Q_j s}$ *is the conditional probability associated with the arc* (Q_i, Q_j, s), *given that the process is in state* Q_i. *We have:*

$$p_{Q_i Q_j s} \begin{cases} > 0 & if\ (Q_i, Q_j, s) \in A \\ = 0 & otherwise \end{cases}$$

$$\sum_{Q_j \in Q(F), s \in L(F)} p_{Q_i Q_j s} = 1 \quad \forall i$$

The triplet $(Q(F), L(F), A(F))$ is a non-probabilistic FSM. It is referred in this paper as the structure of the PFSM F.

Definition 2 *A sequence* $S = s_1 s_2 ... s_m$ *of labels is accepted by the PFSM* F *if there exists a sequence of arcs (or route),* $R = a_1 a_2 ... a_m$, *such that:*

1. $a_i \in A(F)$ *for* $i = 1 \ ... \ m$.
2. *The label of arc* a_i *is* s_i *for* $i = 1 \ ... \ m$.
3. *The destination of arc* a_i *is the origin of arc* a_{i+1}, *for* $i = 1$ *to* $m - 1$. *Let* Q_i *be the origin of arc* a_i *then :*

$$a_i = (Q_i, Q_{i+1}, s_i) \quad and \quad a_{i+1} = (Q_{i+1}, Q_{i+2}, s_{i+1})$$

The probability of route R *is defined as:*

$$p_F(R) = p^{\circ}_{Q_1} \times p^{f}_{Q_{m+1}} \times \prod_{i=1}^{i=m} p_{a_i}$$

Definition 3 *Let* $\mathcal{N}$ *be a set of corruption operations on a sequence of labels. A data sequence* S *is compatible with the PFSM* F *if and only if there exists a route* R *generating a sequence* $S^* = s^*_1 ... s^*_n$ *of labels, and a sequence of corruption operations* $\mathcal{N} = N_1 ... N_{n_c}$ *such that* S^* *is accepted by* F *and* $\mathcal{N}$ *gives the sequence* S *when applied to* S^**:*

$$\mathcal{N}(S^*) = N_1(...N_{n_c}(S^*)) = S$$

The probability of the pair $(R, \mathcal{N})$ *given the PFSM* F *is :*

$$P_F(R, \mathcal{N}) = p_F(R) \times p(\mathcal{N})$$

where $p_F(R)$ *is the probability of* R *given* F *and* $p(\mathcal{N})$ *is the noise probability. The triplet* $(S, R, \mathcal{N})$ *is said to be consistent with the PFSM* F.

3 System representation

Our representation intends to capture the relation between the occurrence of a fault and the alarms observed in the network (we do not claim to model the full behavior of the network). Capturing this relationship leads to relatively simple models for the faults which can then be used efficiently to correlate alarms.

3.1 Fault model

We model a fault as a PFSM (see Def.1) such that the possible outputs of the PFSM correspond to the possible alarm sequences generated by the fault.

Network protocols have long be modeled as FSMs [11, 12, 13]. Communication networks, where a large number of those protocols interact are often modeled as a set of communicating FSMS [11]. The FSM capturing the whole network behavior has a prohibitive complexity, but our point is that the PFSM approximating the network behavior in term of the alarms generated by a specific fault is considerably less complex. In some sense, the states of this PFSM can be interpreted as the result of some operations such as merger on a number of the states of the general FSM representing the network.

The occurrence of a fault results in general in a sequence of alarms as the effects of the fault spread out away from the fault original location. The first symptoms are probably noticed in the immediate "neighborhood" of the fault location, other symptoms may later be observed in more distant locations. The same phenomenon occurs as a results of the network protocols. Abnormal behavior of the network is detected by some of the network protocols, which communicate the information to some other protocols, leading to a sequence of alarms in the network. Because of this propagation effect, sequences rather than sets of alarms have to be considered (i.e., the time order matters). Moreover, the probabilistic nature of our model allows us to capture in a compact form such random facts as the current state of the different protocols when the fault occurs (possibly leading to different alarms) or the "probabilistic nature" of the fault (e.g., the fault "communication card down" encompasses many failures: any chips may be down, several causes may be at the origin of the failure, and in each case the alarms generated may be slightly different). As no restriction is put on the PFSM structure, we can capture typical fault behavior such as repetitive phenomena where messages are sent and some negative acknowledgements are coming back (cycle in the PFSM structure).

Note that if two faults f_1 and f_2 are independent of each other in the sense that the occurrence of one fault does not modify the symptoms associated to the other fault, then the product of the two PFSMs F_1 and F_2 corresponding to each fault is a PFSM model for the simultaneous occurrence of f_1 and f_2.

3.2 Data (alarm) model

A fault results in a number of alarms. The information carried by each alarm may vary from one network to the other. The "ideal" alarm would carry enough information to make the correlation of alarms trivial (an example of ideal alarm is given in [8]). However, as mentioned in [8], the alarms in most communication networks are far from ideal, mainly because alarms are emitted by devices which have only partial information about the rest of the network. We assume here that each alarm carries the following information: identification (device name and symptom) and time of occurrence. This assumption holds in most networks.

The alarms resulting from a fault are then modeled as the output of the PFSM representing the fault (i.e., sequence of labels accepted by the PFSM, see Def. 2). The time information is used to order the sequence. The identification of the alarm is used as label. Since the data collected by the network management system is likely to be corrupted and/or incomplete, we model an observed sequence of alarms as the corrupted version of some original sequence accepted by the PFSM modeling the fault. Fig 1 illustrates this model.

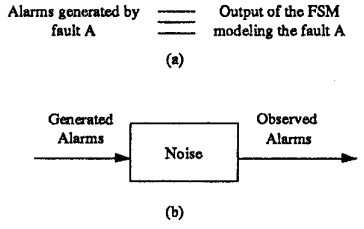

Figure 1: Data model: (a) "Ideal" data, (b) Observed data.

We define the noise as a sequence $\mathcal{N} = N_1...N_{n_\mathcal{N}}$ of corruption operations. The following operations are considered:

1. alarm addition $Add_{X,n}(S)$ e.g., $Add_{A,2}(BC) = BAC$
2. alarm deletion $Del_{X,n}(S)$ e.g., $Del_{A,2}(BAC) = BC$
3. alarm change $C_{X,Y,n}(S)$ e.g., $C_{A,B,1}(AC) = BC$
4. order change of two alarms $O_{X,Y,n}(S)$ e.g., $O_{A,B,1}(AB) = BA$

The operations listed above can model a variety of real phenomena. Delays and wrong time stamps lead to changes in the ordering of the alarms. Superposition of faults can lead both to alarm additions (alarms related to other faults than the one studied are observed) and alarm deletions (one of the elements supposed to emit an alarm may be dead, therefore the alarm is not observed). Transient failures can result in unreliable alarms and thus additions. We assume that the operations are independent of each others. An observed alarm sequence S is then modeled as obtained from some ideal sequence S^* through a sequence $\mathcal{N}$ of corruption operations: i.e., if F is the PFSM associated to the fault, we specify a route R in F and a corruption sequence $\mathcal{N}$ such that $(S, R, \mathcal{N})$ is consistent with F (Def. 3).

4 Phase I: Learning

Given history data associated to a known network fault, we propose to infer its PFSM model. The initial identification of faults in the training data could come from an expert human operator or a model-based expert system.

4.1 Formulation

The history data used to infer the PFSM model F of the fault f is a concatenation of alarm sequences resulting from f, $S(f) = S_1(f)...S_{n_d}(f)$, where each sequence $S_i(f)$ is modeled as the corrupted version of some sequence accepted by the fault PFSM model. We formulate the problem of inferring the PFSM F representing a fault f from $S(f)$, as a cost minimization problem. The formulation is based on [14], where Hart proposes a model for the identification of discrete structures from observed data based on a formal language description of both the data and the structure, and a criterion which is the information content of this description.

We define a complexity cost measuring the PFSM complexity as:

$$C_{complex}(F) = A(2\log Q + \log L)$$

where Q, A and L are respectively the number of states, arcs and labels of the PFSM F. We define a cost $C_{fit}(F, S(f))$ measuring how well the F fits the data $S(f)$ as:

$$C_{fit}(F, S(f)) = \sum_{j=1}^{n_d} \min_{(R_i,N_i)\in\Lambda(F,S_i(f))} -\log P_F(R_i, N_i)$$

where R_i is a path in F (generating an ideal sequence S_i^*), N_i is a sequence of corruption operations, and $\Lambda(F, S_i(f))$ is the set of pairs (R_i, N_i) such that $(S_i(f), R_i, N_i)$ is consistent with the PFSM F (if the set is empty, i.e., $S_i(f)$ is incompatible with F, then $C_{fit} = \infty$). The total cost of the PFSM F given the data $S(f)$ is then defined as:

$$C(F, S(f)) = C_{complex}(F) + C_{fit}(F, S(f)) \quad (1)$$

This cost allows us to trade-off between the PFSM fit to the data and its complexity. For comparison with other criteria proposed in the literature, refer to [15].

Having defined the cost $C(F, S(f))$ of a PFSM F given the data $S(f)$, we can now formulate the inference problem as:

Problem 1 *Given S(f), choose F which minimizes C(F,S(f))*

4.2 Algorithm

The problem of minimizing $C(F, S(f))$ is a hard problem. The algorithm proposed here is based on the work presented in [15, 16, 17]. [15] considers a similar though less complex problem (the data was exact and the class of FSMs studied smaller). As in [15], we first simplify the search by choosing to search only the space of non-probabilistic finite state machines, i.e., the transitions are initially considered as present or absent. Then, for each non probabilistic FSM considered, we generate a PFSM which minimizes the cost $C(F, S(f))$ for this given structure. As the size of FSM space, although

smaller than the one of the PFSM space, makes any exhaustive search impractical, we opt for a local search and settle for a local minimum of our cost function. This solution appears to be very good for our purposes as shown by a range of examples. Section 4.2.1 gives the structure of the main algorithm. Section 4.2.2 discusses briefly the complexity of the algorithm.

4.2.1 Main algorithm

The algorithm is basically a greedy search in the space of non-probabilistic FSMs. We use a local search, where at each step (i.e., as new data is observed and processed), the algorithm searches through the neighborhoods of the solutions obtained at the previous step. The neighborhood of a given non-probabilistic FSM is obtained through a set of operations: splitting of a state, merging of two states, addition of an arc, deletion of an arc, change of the destination of an arc. Once a non probabilistic FSM is generated, the algorithm calls a routine which produces a PFSM minimizing the cost given the structure. The structures or non probabilistic FSMs corresponding to the K lowest cost PFSMs are kept as starting points for the next step search. K is a parameter which we typically make one, two, or three. Two routines, *fit* and *FSM-to-PFSM*, are introduced to deal with the corrupted data. The *fit* routine computes the fitting cost C_{fit} of the PFSMs given the corrupted data. This routine can be seen as an extension of the Viterbi algorithm —a form of dynamic programming [18]— which is an efficient optimal algorithm for correcting symbol sequences generated by a FSM (or PFSM) and corrupted by independent errors (details can be found in [17]). The *fit* routine is more powerful than the Viterbi algorithm in that it corrects a wider class of errors including insertions, deletions, and multi-symbol errors such as the transposition of two symbols. These are important classes of errors for data sent by networks as explained in section 3. The *FSM-to-PFSM* routine iterates the *fit* routine to generate the "best" PFSM from a given FSM. This results in an algorithm similar to the EM algorithm [19, 20] or the Baum-Welsh algorithm [21] but allowing a wider range of corruptions [1]. The main algorithm as well as the *FSM-to-PFSM* routine are described in detail in [22]. Fig. 2 illustrates a simple walk through of this algorithm.

4.2.2 Complexity of the algorithm

The complexity of the inference algorithm can be estimated as follows. Let Q and A be respectively the number of states and arcs of the FSM. At a given step, the algorithm performs two main operations: it generates new structures, and for each new generated structure it runs the *FSM-to-PFSM* routine. The generation phase is roughly the same as in [15]. An upper bound on its complexity is $O(Q^2A^3)$. Because arcs not evidenced in the data are not proposed or retained, A and Q can not exceed the length of the input string. The second phase complexity is dominated by the K iterations of the *fit* routine. This routine is shown in [17] to be polynomial in the number Q of states in the PFSM and the number N_C of possible corruption operations: its complexity is of order $O(N_C Q^3)$

[1] Both the EM and Baum-Welsh algorithms are iterative procedures to obtain maximum likelihood estimates of probabilities.

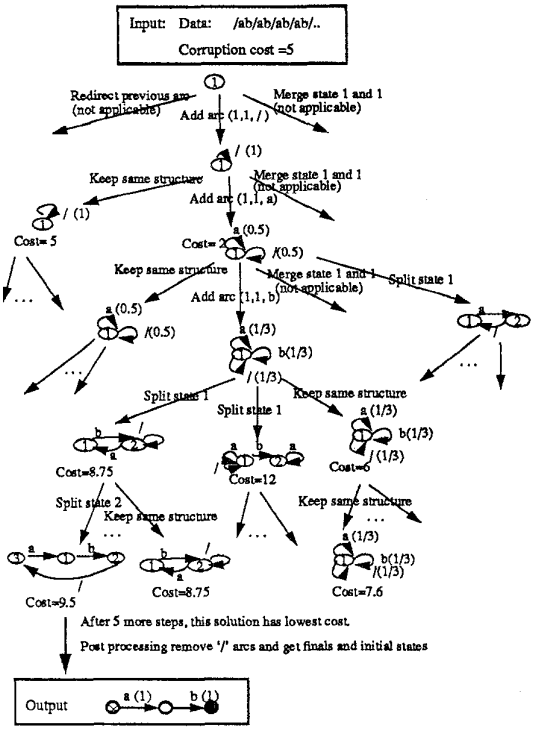

Figure 2: Simple walk through of the inference algorithm.

(slightly higher than the Viterbi algorithm complexity). The total complexity of the second phase is thus $O(KN_CQ^3)$.

5 Phase II: on-line correlation

The second phase of our approach addresses the problem of the on-line identification of the faults. We formulate this second problem as a maximization problem and propose two algorithms. Those algorithms interpret the on-line data as a set of possibly interleaved subsequences, where each subsequence is compatible with a PFSM F (i.e., corrupted alarms resulting from the fault modeled by F). Each interpretation is given a "likelihood" measure. We choose the "most likely" one.

5.1 Formulation

Given a sequence of alarms, we would like to decide whether a fault has occurred or not, and if a fault has occurred which one is most probable. We would also like to detect multiple faults if indeed several faults are responsible for the alarms we observe. The first step in our formulation is to define a model for the occurrence of "no fault" (we denote this specific occurrence as f_0). In the absence of known faults, alarms are occasionally generated. We assume that this generation is random, and is the main factor behind the corruption "alarm insertion." We can then model the "no fault" occurrence as a PFSM F_0 with a single state and no arcs: a sequence of n alarms is thus "seen" by F_0 as n insertions[2]. Let F_{set} be the set of possible faults (including f_0). The observed data

[2] If the background noise was not random and had some structure, F_0 could be acquired as the faults models

is in general an interleaving of alarm sequences generated by different faults where each alarm is generated by one and only one fault. Those sequences may overlap (e.g., if a fault results in sequence AB and another one in C, you may observe ACB rather than ABC or CAB). One of the problems of on-line identification is thus to decide which portion of the observed data corresponds to the alarm sequence generated by a given fault. We define the concept of a p-coverage to formalize the idea that different sequences due to different faults are interleaved, i.e., shuffled together, without losses or reorderings, to give the observed sequence.

Definition 4 *Let $S = s_1 ... s_{n_a}$ be a sequence of alarms and p be a positive integer. A p-coverage of the sequence S is a set of p sequences, $\{S_1, ..., S_i = s_i^1 ... s_i^{l_i}, S_p\}$ such that:*

1. *A given alarm s_j is in one and only one of the p sequences, i.e., s_j has one and only one cause (be it noise)*

$$\exists!\ (i,l) \text{ such that } s_j = s_i^l \text{ and } \sum_{i=1}^{p} l_i = n_a$$

2. *The total ordering is maintained:*

$$\forall i = 1, ..., p \quad s_i^j = s_l,\ s_i^k = s_m, \text{ and } k > j \Longrightarrow m > l$$

The set of all p-coverages of S is denoted $Cov^p(S)$.

If p faults occur "simultaneously", the set containing the sequences they generate is a p-coverage of the total observed sequence. An integer p, a set of p faults, and a corresponding p-coverage constitute then a possible interpretation of the data. Our problem is to find the best possible interpretation for the observed alarms. It is stated as a maximization problem:

Problem 2 *Let $S = s_1 ... s_{n_a}$ be a sequence of alarms (possibly noisy). Define $\hat{P}(S/F)$ as the probability of the most likely route/noise pair (R, N) such that (S, N, R) is consistent with the PFSM F (Def. 3):*

$$\hat{P}(S/F) = \max_{(R,N) \in \Lambda(F,S)} P_F(R, N)$$

where $\Lambda(F, S)$ is the set of pairs (R, N) such that (S, R, N) is consistent with F.
Find the integer p^* and the set of faults $\{f_1^*, ..., f_{p^*}^*\}$ such that the tuple $(p^*, f_1^*, ..., f_{p^*}^*)$ maximizes:

$$\max_{Cov^p(S)} P(p/S) P(f_1 f_2 ... f_p) \prod_{i=1}^{p} \hat{P}(S_i/F_i) \qquad (2)$$

S_i is the ith sub-sequence in the p-coverage $\{S_1, ..., S_n\}$ of $Cov^p(S)$. $P(p/S)$ is the probability of having p faults given S. $P(f_1, ..., f_p)$ is the a priori probability of the simultaneous occurrence of the p faults $f_1, ..., f_p$. F_i is the PFSM corresponding to fault f_i.

5.2 Algorithms

We propose here two algorithms. Both algorithms process a given "window" of data at a time, producing the most "probable" set of faults for every window (if the resulting set is $\{f_0\}$, we conclude that no fault has occurred). The consecutive windows may in general be overlapping. In certain cases, we may choose to process the most recent alarms at regular interval of time, e.g., process the most recent $\frac{1}{2}$ hour of data every two minutes. The "window" of data considered is then whatever amount of data came in during the last half hour. This paper focuses on the processing of a given window. The problem of choosing the appropriate window and overlapping (if any) is another very interresting problem, but is beyond the scope of this paper. We would just like to note at this point that because of its ability to deal with noise (and in particular, with additions and deletions), our algorithm proves to be robust to different window and/or overlapping choices.

Let S be the window of data processed at a given time, the set $F^*_{probable} = \{f_1^*, ..., f_{p^*}^*\}$ given as output by the first and second algorithms is such that $(p^*, F^*_{probable})$ is respectively a global and a local maximum of (2). The first algorithm is optimal, but of exponential complexity. The second one settles for a "good" solution which is a local maximum of (2) in polynomial time. Both algorithms are based on an *interpretation* routine which assesses "how good" an hypothesis is. Section 5.2.1 describes this routine. Section 5.2.2 and 5.2.3 describe respectively the first and second algorithm.

5.2.1 *Interpretation* routine

The *interpretation* routine is based on the *fit* routine. It finds the best possible interpretation of the data (p-coverage) for a given set of faults and uses this interpretation to associate a likelihood measure to the set.
The general structure of the routine is:

- Input:
  1. The observed data S.
  2. A set of faults $\{f_0, f_1, ..., f_{p-1}\}$ and their corresponding PFSM models $\{F_0, ..., F_{p-1}\}$.
  3. A set of corruption operations and their probability
- Output:
  1. The best p-coverage $(S_0, ..., S_{p-1})$ of S given this set of faults.
  2. A cost

$$C(p, f_0 ... f_{p-1}) = P(f_0 ... f_{p-1}) P(p/S) \max_{Cov^p(S)} \prod_{i=0}^{p-1} \hat{P}(S_i/F_i)$$

A description of the different steps of the routine follows:

1. Build the PFSM E corresponding to the simultaneous occurrence of the faults given as input: E is the p-product of the PFSMs F_1 to F_P[3]. Note that since a PFSM is the product of itself with F_0, the sets of faults considered always contain the no fault occurrence f_0.
2. Run the *fit* algorithm with input:

[3] This comes directly from the assumption of independent faults.

- The PFSM E
- The data S
- A set of possible corruption operations and their likelihood[4].

3. Let S^* be the original output sequence of E given by the *fit* algorithm.
Let $R = a_1 ... a_n$ be the path generating S^*.
Define a p-coverage $(R_0, ..., R_{p-1})$ of R, where $R_i = a_i^1 ... a_i^{l_i}$ is defined as follows:

   (a) $a_i^x \in R \quad \forall x = 1, ..., l_i$

   (b) $a_i^x \in A(F_i) \quad \forall x = 1, ..., l_i$

   (c) if $a_i^x = a_l$ and $a_i^y = a_m$ with $y \geq x$ then $m \geq l$

   It is straightforward to check that $(R_0, ..., R_{p-1})$ is indeed a p-coverage of R (note that $(R_0, S_0^*) = (\emptyset, \emptyset)$).

4. Let S_i^* be the sequence generated by R_i. $(S_0^*, ..., S_{p-1}^*)$ is a p-coverage of S^* such that S_i^* is generated by F_i (by definition of the R_i). Let N be the noise description given by the *fit* algorithm. Define N_0 as the subsequence of N containing all the addition operations in N. Define $N_i (i \neq 0)$ as the subsequence of N consisting of the operations (besides additions) applying to the elements of S_i^*. Define $S_i = N_i(S_i^*)$. Then:

   Result 1 *$(S_0, ..., S_{p-1})$ is a p-coverage of S and*

$$\prod_{i=0}^{p-1} P_{F_i}(R_i, N_i) = \max_{Cov^p(S)} \prod_{i=0}^{p-1} \max_{(R,N) \in A(F_i, S_i)} P_{F_i}(R, N)$$

   Proof in [22].

5. Compute the "likelihood" $C(p, f_0, ..., f_{p-1})$ of E:

$$C(p, f_0, ..., f_{p-1}) = P(f_0 ... f_{p-1}) P(p/S) \prod_{i=0}^{p-1} P_{F_i}(R_i, N_i)$$

5.2.2 Algorithm I

This algorithm applies the *interpretation* routine to every possible set of occurrences[5]. If N is the number of possible faults, the number of possible sets is then of order 2^N.

1. For every tuple $(p, f_0, ..., f_{p-1})$, run the *interpretation* routine with input:
   - The set of faults $\{f_0, ..., f_{p-1}\}$
   - The data S
   - The set of possible corruption operations and their likelihood.
2. Let $(p^*, f_0, f_1^*, ..., f_{p-1}^*) = argmax C(p, f_0, ..., f_{p-1})$

   Result 2 *$(p^*, f_0, f_1^*, ..., f_{p-1}^*)$ is a global maximum of (2). The proof follows directly from result 1.*

This algorithm although optimal is of prohibitive complexity. This is due to two main factors. The number of possible hypothesis (i.e., sets) grows exponentially with the number of faults considered. The number of states and arcs of the PFSM resulting from the product of p PFSMs grows exponentially with p. Therefore, the *fit* algorithm complexity becomes prohibitive (being polynomial in the number of arcs and states in the PFSM).

5.2.3 Algorithm II

We propose here a heuristic polynomial algorithm which iterates the *interpretation* routine. At the *jth* iteration, the data S^j is assumed to have been generated by (1) a single fault and possibly noise (i.e., $p = 2$), or (2) pure noise. We select the best single fault interpretation, put the corresponding fault f_j^* in the set of most probable faults, and repeat our operation on the data considered pure noise in this interpretation. This greedy iterative process is stopped either when the entire remaining sequence is considered noise, (i.e., the best set is $\{f_0\}$) or when there is no data left.
The description of the algorithm follows:

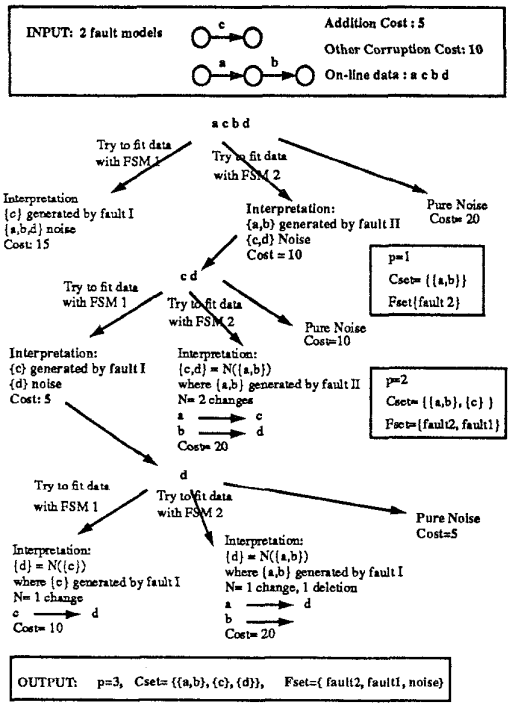

Figure 3: Simple walk through of the correlation algorithm. Cost(x) = - log P(x) (i.e., the maximization appears as a minimization). All values of p and all set of faults are assumed equiprobable (i.e., we ignore the corresponding terms in o ur cost function).

1. Initialization of greedy search to locally maximize (2).
   $j = 0$, $S^0 = S$

[4] The noise can be a sequence of any operation described in section 3. However, we do not allow a switch operation on alarms resulting from two different faults. This operation does not make sense within our model where faults are assumed to be independent

[5] Sets considered include the no fault occurrence f_0. This is required for our approach to be systematic. S_0, the subsequence of S corresponding to pure noise, may end up to be the empty sequence.

2. Greedy iteration: jth iteration

   (a) Consider the data sequence S^j.
   For each fault f_i in F_{set}, apply the *interpretation* routine with input:
   - The set $\{f_0, f_i\}$
   - data S^j
   - Noise description.

   (b) Let f_k be the fault corresponding to the most likely single fault interpretation. Set $f_j^* = f_k$ (f_j^* is the jth element of the set $F^*_{probable}$). Let $S_0^j(k)$ be the subsequence of S^j attributed to f_0 (noise) by the *interpretation* routine when given the input set $\{f_o, f_k\}$.
   If $k = 0$ or $S_0^j(k)$ is empty, stop
   else
   - Set $S^{j+1} = S_0^j(k)$
   - Go to step 2

A simple walk through of the algorithm is illustrated in Fig 3. The algorithm keeps a minimum of data in memory. Its complexity is essentially the complexity of the *fit* algorithm, that is polynomial in the number of possible corruptions and the number of states in the different PFSMs. Furthermore, it could very easily be implemented in hardware since it is essentially based on an extension of the Viterbi algorithm for which numerous hardware implementations have been developed. Those hardware implementations could handle very high alarm rates.

6 Examples

Symptoms description	Symbol
CPU starvation	A
Loss of ibgp session	B
Normal state	C
Packet loss	D
ccstat indicates some problem	E
is-is adjacency loss	F
card "j" does not respond to ping	G_j
snmp circuit error variables show non-zero count	H
dsu report shows high errors	I
AS (Autonomous System) become unreachable	J
Intermittent packet loss	K

Table 1: Alarm Table (Note: ccsat is a utility that gives interface statistics by querying kernel variables).

We apply here our approach to data generated by the ANS/NSF T3 network. The NSF network is a T3-based network with 13 core nodes spanning the United States. It links all National Science Foundation sites, interconnects 818 networks and carries more than 2 billions packet of information each month. We consider alarms generated by the simultaneous occurrences of one or more of the following faults.

- Fault (I): a bug in the tcp fin_wait state code
- Fault (II): an interface card is bad
- Fault (III): a T3 circuit is bad;
- Fault (IV): route flapping (loss of ibgp sessions)

6.1 Learning phase

The training data to acquire the fault models was simulated from information provided by experts running the ANS/NSF T3 network. Table 1 lists all the alarms involved and the symbols used to represent them in the rest of this paper. Fig 4 show the models obtained for the four faults mentioned above. In each case, we also give an example of training data. Note that what we refer to as alarms are symptoms checked on a regular basis in the network. Any symptom which could be checked by the network operator is not considered. Our aim is to provide a tool to identify faults automatically, therefore only information provided automatically by the network is used. Because of this reduced set of symptoms, we may have cases where different faults exhibit similar behavior, and thus result in similar model. If we were to apply this tool to a specific network, we could decide, based on the results we obtain, to enlarge the set of symptoms that are checked. However, the FSMs thus obtained would probably be more complex, and therefore the on-line procedure would take more time (as it is polynomial in the number of states and arcs in the FSM). This results in a tradeoff between the accuracy of the results and the simplicity of the technique.

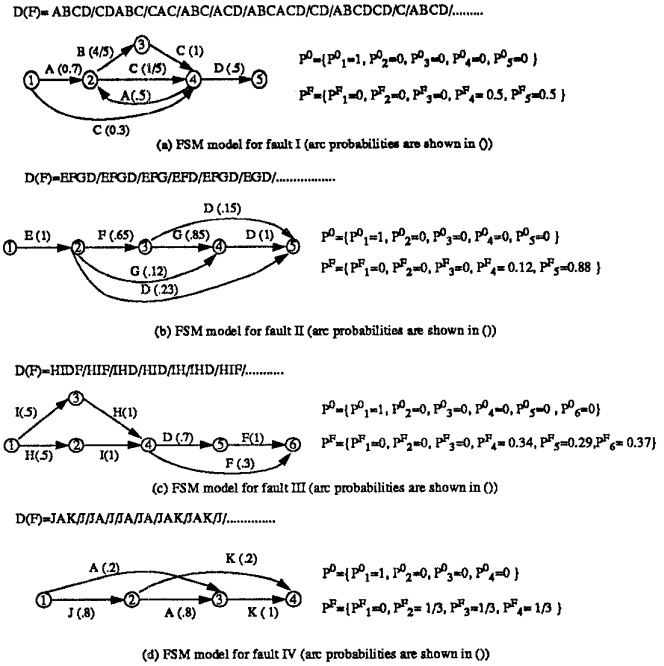

Figure 4: FSM models of the faults

6.2 Correlation phase

We have tried algorithm II on simple alarm sequences generated by a single fault at a time, and it easily identified the fault. We give here some examples of noisy alarm data from multiple faults. In all cases, the data was generated according

to the information provided by ANS. In each case, the algorithm was able to find the sources of the alarms in less than a second. All the faults were supposed to be equiprobable.

Example 1: Alarms generated by simultaneous occurrence of faults I , II and IV. The data is ABEGCJDAD. The algorithm correctly recognizes the occurrence of fault I, II, and IV. The corresponding 3-coverage is $\{C_1 = ABCD, C_2 = EGD, C_3 = JA\}$. Note that the algorithm does not attribute any observed alarms to pure noise. Fig 5 illustrates the steps of the algorithm in this case.

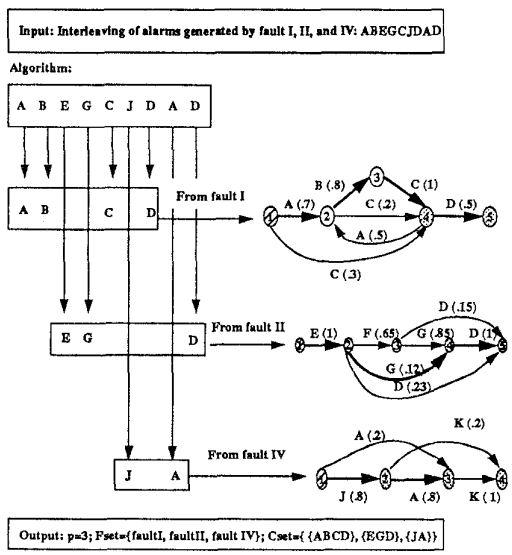

Figure 5: Example 1. Heavy lines in the FSMs indicate the paths generating the alarms.

Example 2: Alarms generated by simultaneous occurrence of faults III and IV. The data is IHDDJ. The output of the algorithm is: $F_{probable} = \{III, IV, f_0\}$ and the corresponding 3-coverage $\{C_1 = IHD, C_2 = J, C_3 = D\}$. As we can see one of the observed alarms is considered noise. If we make the probability of an addition lower than the other corruptions, the output of the algorithm becomes $F_{probable} = \{III, IV\}$ and $C_{set} = \{C_1 = IHDD, C_2 = J\}$. C_1 is interpreted by the fit algorithm as the corrupted version of the sequence $IHDF$ where the last F is misinterpreted as a D. Fig 6 illustrates this example.

Example 3: Alarms generated by two occurrences of fault III. This example shows how the algorithm is able to identify several occurrences of the same fault. Two cases are considered non-corrupted and corrupted data. In the first case, the data is HIIDHFF. The algorithm produces $F_{probable} = \{III, III\}$ and the coverage $\{C_1 = HIDF, C_2 = IHF\}$. If the data is corrupted to $HIIDFF$ (the H alarm resulting from the second occurrence of fault III is not observed), the algorithm still identifies the two occurrences. Its output is: $F_{probable} = \{III, III\}$ and the coverage $\{C_1 = HIDF, C_2 = IF\}$. C_2 is interpreted by the fit algorithm as being the corrupted version of IHF. Note that C_2 could also

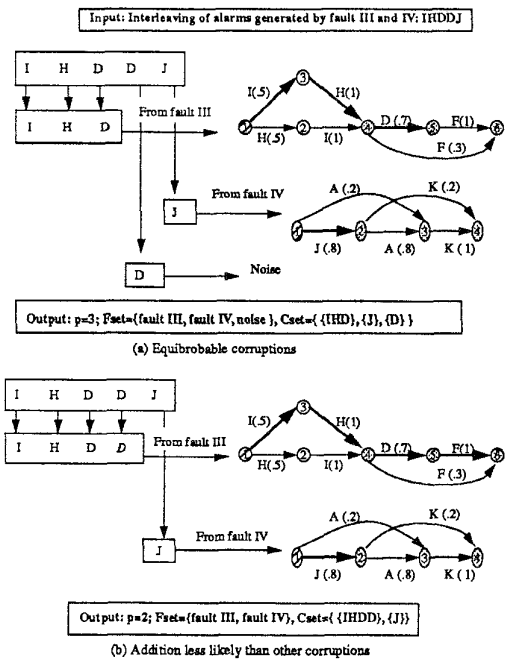

Figure 6: Example 2 (heavy lines in the FSMs indicate the paths generating the alarms).

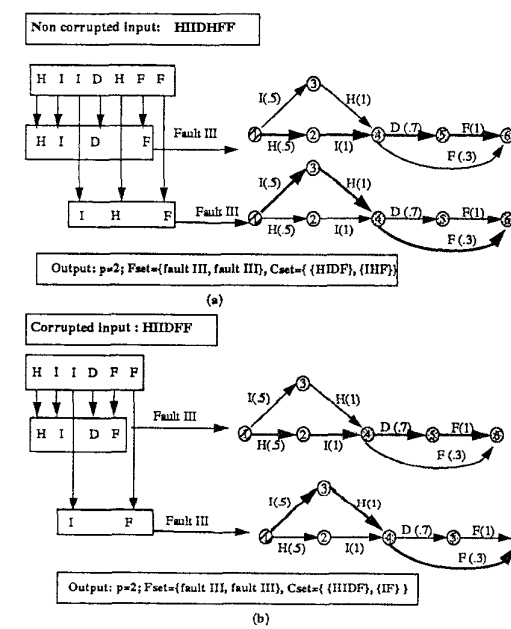

Figure 7: Example 3: Two occurrences of the same fault (heavy lines in the FSMs indicate the paths generating the alarms).

have been interpreted as the corrupted version of HIF. This latter interpretation has same cost. Fig 7 illustrates this example.

7 Conclusion

We presented a new approach to the alarm correlation problem. A fault is modeled as a probabilistic finite state machine such that the possible output sequences correspond to the possible alarm sequences resulting from the fault. The data model includes a noise description allowing us to handle noisy data. Our approach then divides into two phases. A learning phase acquires the model of each fault from possibly incomplete and/or incorrect history data. A correlation phase uses those models to interpret the on-line data and identify faults. Heuristic algorithms are proposed for both phases. Both algorithms have polynomial time complexity, use an extension of the Viterbi algorithm to deal with the corrupted data, and can easily be implemented in hardware. As an example, they are applied to data generated by the ANS/NSF T3 network. We modeled faults including a bug in the TCP protocol, route flapping, and a problem with a T3 interface card. We tested the identification algorithm on data generated by one or several of those faults occurring simultaneously, and were able to identify the source(s) of the alarms successfully.

Acknowledgement

The authors would like to thank Sharad Sanghi for providing the data for the examples in section 6, and aiding in its interpretation.

References

[1] E. C. Ericson, L. T. Ericson, and D. Minoli, eds., *Expert System Applications in Integrated Network Management*, ch. 5: Fault Management Applications. Artech House, 1989.

[2] R. Mathonet, H. V. Cotthem, and L. Vanryckeghem, "DANTES: An Expert Systems for Real-Time Network Troubleshooting," in *Expert System Applications in Integrated Network Management* (E. C. Ericson, L. T. Ericson, and D. Minoli, eds.), pp. 259–262, Artech House, 1989.

[3] T. D. Marques, "A symptom-driven expert system for isolating and correcting network faults," in *Expert System Applications in Integrated Network Management* (E. C. Ericson, L. T. Ericson, and D. Minoli, eds.), pp. 251–258, Artech House, 1989.

[4] P. H. Callahan, "Expert Systems for AT&T Switched Network Maintenance," in *Expert System Applications in Integrated Network Management* (E. C. Ericson, L. T. Ericson, and D. Minoli, eds.), pp. 263–273, Artech House, 1989.

[5] *Expert Fault Management Service- User's Guide*, February 1990.

[6] "IEEE Network, Special Issue on Expert Systems for Network Management in Communication Networks," September 1988.

[7] A. Bouloutas, *Modelling Fault Management in Communication Networks.* PhD thesis, Columbia University, 1990.

[8] A. Bouloutas, S. Calo, and A. Finkel, "Alarm Correlation and Fault Identification in Communication Network." Submitted for publication in IEEE Trans. on Communication.

[9] Y. Peng and J. A. Reggia, "A Probabilistic Causal Model for Diagnostic Problem Solving-Part I: Integrating Symbolic Causal Inference with Numeric Probabilistic Inference," *IEEE Trans. on Systems, Man, and Cybernetics*, March-April 1987.

[10] Y. Peng and J. A. Reggia, "A Probabilistic Causal Model for Diagnostic Problem Solving-Part II: Diagnostic Strategy," *IEEE Trans. on Systems, Man, and Cybernetics*, May-June 1987.

[11] D. Brand and P. Zafiropulo, "On Communicating Finite State Models," *Journal of the ACM*, pp. 323–342, April 1983.

[12] G. V. Bochmann and C. Sunshine, "Use of Formal Methods in Communication Protocol Design," *IEEE Transactions on Communication*, pp. 624–631, April 1980.

[13] A. Danthine, "Protocol Representation with Finite State Models," *IEEE Transactions on Communication*, pp. 632–431, April 1980.

[14] G. W. Hart, *Minimum Information Estimation of Structure.* PhD thesis, MIT, 1987. Laboratory for Information and Decision System, Report LISD-TH-1664.

[15] I. Rouvellou and G. Hart, "Inference of a Probabilistic Finite State Machine from its Output." Presented at ORSA'92, Boca Raton, Florida; extended version to appear in IEEE Trans. on Systems, Man, and Cybernetics (Feb. 95 issue).

[16] A. Bouloutas, G. Hart, and M.Schwartz, "Identification of Finite State Machine using Unreliable Partially Observed Data Sequences," *IEEE trans. on Communication*, 1992.

[17] G. Hart and A. Bouloutas, "Correcting Dependent Errors in Sequences Generated by Finite-state Processes," *IEEE trans. on Information Theory*, July 1993.

[18] G. Forney, "The Viterbi Algorithm," *Proc. IEEE-61*, pp. 268–278, 1973.

[19] L. E. Baum, T. Petri, G. Soules, and N. Weiss, "A Maximization Technique Ocurring in Statistical Analysis of Probabilistic Function of Markov Chains," *Ann. Math. Statistic.*, pp. 164–171, 1970.

[20] A. P. Dempster, N. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Royal Statistical Society Journal*, pp. 1–22, 1976.

[21] L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Model," *IEEE ASSP Magazine*, pp. 4–16, January 1986.

[22] I. Rouvellou, *Identification Techniques Applied to Network Management.* PhD thesis, Columbia University, 1993.