Distributed Diagnosis of Byzantine Processors and Links

Joel C. Adams  K. V. S. Ramarao

Department of Computer Science
University of Pittsburgh

ABSTRACT
We examine the problem of correctly identifying the faulty processors and links in a distributed system where faulty behavior is unrestricted (byzantine). A very general class of algorithms called evidence-based diagnosis algorithms is proposed, which encompasses all past approaches to the diagnosis problem. An algorithm is presented which is proven optimal in this class. We argue both analytically and from experimental data that in systems of N processors of which t can be faulty, the complexity of this algorithm is $O(\max(2^t, N^2))$.

1. Introduction
The problem of diagnosing (i.e. identifying) the faulty components in a distributed system has traditionally been approached by extending the PMC model [19] where a system's components test one another for faultiness. This model utilizes a well-known testing-graph, whose vertices are the system's processors, and whose (directed) edges $a \rightarrow b$ imply that processor a tests processor b. We call diagnosis algorithms which utilize this model PMC-based diagnosis algorithms.

Mallala and Masson [15] have observed that a correct but incomplete PMC-based diagnosis can occur from (1) connectivity of $i < t$ within the testing-graph; or from (2) faults occurring intermittently instead of permanently. Accordingly, most past investigators have restricted their investigations to permanent faults [4, 6-9, 11, 12, 14, 17, 22].

When faults are permanent, the strongest PMC-based diagnosis algorithms assume that
(1) the outcome of any non-faulty component's test of a faulty component is positive;
(2) the outcome of any non-faulty component's test of a non-faulty component is negative; and
(3) the outcome of any faulty component's test of another component is unpredictable (i.e. either positive or negative).

When faults can occur intermittently (as opposed to being permanent), PMC-based diagnosis algorithms can no longer ensure (1), since it cannot be guaranteed that the testing of a component coincides with its faulty behavior.

One PMC-based approach to diagnosis in which intermittent faults have been examined involves the processing of a syndrome, which is essentially a testing-graph whose edges are augmented with the outcomes of their tests. In [15], hybrid fault diagnosis is examined, in which faulty behavior might be permanent and/or intermittent. Repeated testing is proposed to increase the likelihood of a test coinciding with a faulty behavior. A syndrome may then be "updated" with any more recent positive test outcomes, as tests are repeated.

In [16], it is shown that unless a syndrome is $pf$-compatible (i.e. the syndrome could have been produced by a set of components exhibiting permanent faults), a PMC-based diagnosis algorithm cannot guarantee both correctness and completeness, even with repeated testing. [4] provides an efficient algorithm for processing a $pf$-compatible syndrome, which first noted the relationship between the vertex cover problem [10] and PMC-based diagnosis.

Efficient PMC-based diagnosis algorithms are given in [23] and [24] for systems in which only processors are faulty, and systems in which both processors and links can be faulty, respectively. These algorithms were demonstrated to always return a correct diagnosis, and a complete diagnosis if the syndrome being examined was $pf$-compatible.

As implied by [15], the occurrence of intermittent faults can result in an incomplete diagnosis by any PMC-based algorithm for the following reason: unless a component is tested continuously, it might (in the worst case) fail immediately prior to and following each repetition of a test. However continuous testing essentially prevents a component's participation in the operation of the system (assuming that a component cannot perform useful work simultaneously with being tested), and hence is not a practical solution. Thus, intermittent fault capability can, in the worst case scenarios, result in no faulty components being identified by a PMC-based diagnosis algorithm.

To avoid these shortcomings, Shin and Ramanathan have proposed [21] an approach to the diagnosis of intermittent (and malicious) faults different from that of the PMC-based algorithms. They perform diagnosis during the execution of a full-information consensus protocol [13], by having each participating component monitor each of its peers for deviations from the behavior specified by the protocol. When any component C detects such a behavior by any component B, C records an accusation (our terminology) against B. That is, each component determines the correctness (or faultiness) of each message it receives (or fails to receive) by utilizing the equivalent of a system-level acceptance test, similar in nature to that utilized by Anderson and Kerr in their N-version programming recovery blocks [2, 3].

At the conclusion of the consensus protocol, each component distributes its list of accusations to every other component, and a diagnosis algorithm then processes the set of evidences which results from combining these lists. The problem of processing these accusations becomes interesting when a (maliciously) faulty component sends a list containing false accusations. An algorithm is presented which guarantees a correct diagnosis, but not a complete diagnosis.

1. A diagnosis is said to be correct if no non-faulty component is identified as faulty.
2. In a diagnosis is said to be complete if every faulty component is identified. Hence an incomplete diagnosis occurs when one or more faulty components remain unidentified.

CH2706-0/89/0000/0562$01.00 © 1989 IEEE
While interesting for its novelty, Shin and Ramanathan's work restricts the behavior of faulty components by implicitly assuming that such components cannot omit to record accusations: a faulty component's list of accusations is always a superset of the set of deviant behaviors it observes. Their work also assumes that the system's links behave perfectly. Their algorithm is also specifically designed to operate in the context of the full-information consensus protocol. Our work differs from theirs in (1) permitting faulty processors to omit accusations, (2) applicability to systems in which both processors and links are faulty, and (3) applicability to any message-driven protocol which permits the formation of accusations. Thus, our diagnosis algorithm might be run in conjunction with termination detection, deadlock detection, snapshots, election protocols, etc.

In [20], we have constructed a formal model of algorithms which utilize accusations, for systems in which only processors are faulty. Using this formal model, an algorithm was derived which was demonstrably optimal, in that no other diagnosis algorithm could identify a faulty processor which it was unable to. It was also shown that no algorithm of this class (whose diagnosis was correct) could guarantee a complete diagnosis. As in [4], this work made extensive use of vertex covers.

In this paper, we extend our previous work to systems in which both processors and links can be faulty. This work makes three important contributions to the problem of diagnosis:

1. A very general formal model of evidence-based diagnosis algorithms is given, for systems with faulty processors and links, which encompasses both the PMC-based diagnosis algorithms and the approach proposed by [21];
2. A correct evidence-based diagnosis algorithm is derived for systems in which both processors and links can be arbitrarily faulty. This algorithm is shown to be optimal in the sense that no other evidence-based diagnosis algorithm can identify more faulty components than it can; and
3. An extensive experimental implementation of this algorithm is detailed, in which it is shown that for a system of N processors of which t can be faulty, the complexity of our algorithm is O(max(2^t N^2)).

The paper is organized as follows: in Section 2, we define our formal models and give the assumptions we make. In Section 3, we construct our algorithm, and examine its characteristics. In Section 4, we detail our experimental implementation, and we conclude in Section 5 with some discussion of our work.

2. System Model

A system S consists of a set of N processors and L links which connect those processors. Each processor executes a process, modeled as an automaton, the combination of which comprises a message-driven protocol. So long as a processor behaves as specified by its automaton, we describe it as non-faulty. We assume that up to t of the processors and l of the links may be faulty, that is, may behave in a completely arbitrary (i.e. byzantine [13]) manner. A faulty processor p may send spurious messages, omit to send messages, receive unsent messages, etc. A faulty link (p,q) may deliver messages to q (or p) that were never sent by p (or q), deliver altered messages to q (or p) that were sent by p (or q).

Each processor p can communicate with any of its peers q solely through the exchange of messages with q. In this paper, we assume that the receiver of a message always knows its sender. In a real system, this might be realized either through the use of authentication [5], or by having complete connectivity within the system.

3. Diagnosing Evidence

An instance I of a system S is a quadruple \( \langle \lambda, PF, L, L' \rangle \) where \( \lambda \) bounds the number of faulty processors, \( PF \) is the set of faulty processors in the system, and \( L' \) is the set of faulty links in the system, such that \( |PF| \leq \lambda \) and \( |L'| \leq \lambda \).

2.1. Evidences

In comparing the approach of [21] with that of the more traditional PMC-based algorithms, the former approach gathers accusations during the execution of a full-information consensus protocol and then processes them, while the latter gathers positive test outcomes during some phase of system testing and then processes them. We might thus represent either of these approaches using the following abstraction:

1. Gather a set of evidences;
2. Process those evidences;

where an evidence consists of either an accusation (as per [21]), or a positive test outcome (as per PMC-based algorithms). We thus see that the two approaches are inherently similar and differ only in how they gather evidences. Based on this observation, we derive the most general model as follows: Let \( E_I \) be an execution of a distributed process \( \Pi \) during which the participating processors construct evidences based on their observations of the behaviors of their peers.

Formally, an evidence is a triplet \( \langle p, q, (p,q) \rangle \) where \( p \) is the suspect processor, \( q \) is the suspect link, and \( (p,q) \) is the accusing processor. Such an evidence might be formed by a non-faulty processor \( q \) in either of two ways:

1. During \( E_I \), \( q \) receives (or fails to receive) a message from \( p \) (via \( (p,q) \)) such that that behavior fails \( q \)'s acceptance test; and/or
2. \( q \) tests \( p \) for faultiness, and the outcome of \( q \)'s test is positive.

Of course, a faulty processor \( q \) can form such evidences arbitrarily.

For any such evidence \( \langle p, (p,q), q \rangle \), \( p \) could be faulty, the link \( (p,q) \) could be faulty (i.e. \( (p,q) \) corrupts \( p \)'s communication to \( q )\), \( q \) could be faulty (i.e. \( q \) records a false evidence), or some combination of these. If, for an evidence \( \langle p, (p,q), q \rangle \), either \( p \) is faulty or \( (p,q) \) is faulty, we describe that evidence as true. We say that \( \langle p, (p,q), q \rangle \) is false, otherwise.

Each processor \( p \) thus generates an evidence list \( EV_q \) in which it records all evidences it forms. We stipulate that if a processor \( q \) is non-faulty, then \( EV_q \) contains only true evidences. However, a faulty processor \( q \) can behave arbitrarily, recording true and/or false evidences in \( EV_q \). Also, the possibility exists that a faulty processor \( q \) does not record an evidence \( \langle p, (p,q), q \rangle \) in \( EV_q \), which \( q \) would record if it were non-faulty (e.g. \( q \)'s test of \( p \) is positive, but \( q \) omits \( (p,q) \) from \( EV_q \) ). In such an instance, we describe \( \langle p, (p,q), q \rangle \) as a hidden evidence.

Eventually, these lists are consolidated (perhaps using an interactive consistency protocol [18]) into a master evidence set, \( E_{\text{M}} \). Note that this might be done either during or following the execution \( E_I \). We say that an instance \( I \) of a system in which \( E_{\text{M}} \) is collected produces \( E_{\text{M}} \), and any instance \( I' \) in which \( E_{\text{M}} \) could be produced is capable of producing \( E_{\text{M}} \).

For simplicity, we do not allow a non-faulty processor to generate any evidence in which it is not the accusing processor. Thus, if \( p \) generates \( \langle q, (q,s), s \rangle \), then \( p \) must be faulty and can be identified as faulty when \( E_{\text{M}} \) is formed (since the receiver of any message knows the sender). Hence, we will not consider such evidences any further.
A Testimony graph is a graphical representation of a set of evidences. That is, a Testimony graph \( T \) is an undirected graph \( (V,E) \), whose vertices \( V \) are the processors of the system, and edge \((p,q) \in E\) if and only if \( (p,q) \in E_T \). Thus, a Testimony graph \( T \) might be the subgraph of a syndrome whose edges have positive test outcomes (which Dahbura and Masson call an L-graph [4]), or \( T \) might represent a set of failed acceptance tests, or perhaps some combination of these. We say that the instance \( I \) of the system in which \( EV_T \) is produced produces \( T \), and that any instance \( I' \) capable of producing \( EV_T \) is capable of producing \( T \).

2.2. Evidence-Based Diagnosis Algorithms

An evidence-based diagnosis algorithm accepts a Testimony graph as input, and outputs a set of processors and a set of links. An execution of an evidence-based diagnosis algorithm \( A \) is thus a triplet \((T,A,P,A,L)\) where \( T \) is the Testimony graph used as input, \( A \) is the set of processors caught (i.e. identified as faulty) by \( A \), and \( A,L \) is the set of links caught by \( A \).

We describe two executions \((T,A,P,A,L)\) and \((T',A',P',A',L')\) of \( A \) as equivalent if and only if \( T=T' \). The deterministic nature of such algorithms ensures that for any two such equivalent executions, \( A,P=A',P,A,L=A',L' \). This fact is of critical importance for many of our later results.

We describe an evidence-based diagnosis algorithm as correct if it guarantees the following desirable property:

**Correctness:** no non-faulty processor or link in the instance \( I \) producing \( T \) is identified as faulty.

An evidence-based diagnosis algorithm is complete if it guarantees that

**Completeness:** every faulty processor and link in the instance \( I \) producing \( T \) is identified.

In the next section, we construct an optimal, correct evidence-based diagnosis algorithm.

3. Main Results

Since an illustrative example is often useful to clarify a construction, we will utilize the following running example to illustrate the principles by which our algorithm operates.

**Example 1.** Suppose that a system \( S \) has \( N=7 \) processors. Let \( I \) be an instance of \( S \) in which \( t=1, \lambda=2 \), processor 1 and links (2,5) and (3,4) are faulty, and the Testimony graph \( T \) of Figure 1 is formed.

![Testimony Graph T: Example 1.](image)

**Figure 1.**

In the worst case, it is possible that a non-faulty processor \( p \) has degree \( t+\lambda \), if \( p \) is involved in evidences with \( t \) faulty processors, and with \( \lambda \) faulty links. Hence, we might construct a correct diagnosis algorithm which only declares a processor faulty if it has degree \( d \geq t+\lambda \) in \( T \). Given the Testimony graph of Figure 1, such an algorithm would declare no processors (and no links) faulty, since processor 1 has the maximum degree of \( 3 \leq t+\lambda \). The problem, as we shall see is that while having degree \( d \geq t+\lambda \) in \( T \) is a sufficient condition for declaring a processor faulty, it is not the strongest sufficient condition.

Recall that we have stipulated that a non-faulty processor \( p \) records only true evidences in \( EV_p \). Formally, this implies

\[
P1: \quad \text{for any edge} \ (p,q) \in T, \text{at least one of the following is true:}
\]

\[
(\text{i}) \ p \text{ is faulty;}
\]

\[
(\text{ii}) \ q \text{ is faulty; and/or}
\]

\[
(\text{iii}) \ (p,q) \text{ is faulty.}
\]

**Definition:** A hybrid cover of a graph \( G=(V,E) \) is a pair \( <\nu,ec> ; \nu \subseteq V, ec \subseteq E \) such that for each edge \((p,q)\) \in E: \( p \in \nu, q \in \nu, (p,q) \in ec, \) or some combination of these. Intuitively, a hybrid cover is a set of vertices \( \nu \), and a set of edges \( ec \), such that every edge in the graph is in \( ec \), or has (at least) one of its vertices in \( \nu \), or both. A hybrid cover of \( T \) thus combines the notion of a partial vertex cover and partial edge cover of \( T \). For any hybrid cover \( HC \), we will refer to its first component as \( HC: \nu \) and its second component as \( HC:ec \).

To illustrate the notion of the hybrid cover, consider Figure 1. Some hybrid covers of \( T \) are as follows:

\[
HC_1 = <\{1\},(3,4),(2,5)>,
\]

\[
HC_2 = <\{4,5,6\},\{\}>,
\]

\[
HC_3 = <\{1\},(3,4),(1,4),(2,4),(1,5),(1,6)>,
\]

\[
HC_4 = <\{4,5\},(1,6)>.
\]

We define a sufficient hybrid cover as a hybrid cover \( HC \), where \(| HC: \nu | \leq t, \text{ and } | HC:ec | \leq \lambda \). In examining graph \( T \) of Figure 1, we see that only \( HC_1 \) is a sufficient hybrid cover of \( T \), and it is in fact the unique sufficient hybrid cover of \( T \).

In Example 1, there is a unique sufficient hybrid cover of \( T \), but in other examples there might be multiple sufficient hybrid covers of \( T \). Intuitively, each sufficient hybrid cover \( HC_i \) of a Testimony graph \( T \) represents a distinct instance \( I \) of the system in which the processors in \( HC: \nu \) are faulty, the links in \( HC:ec \) are faulty, and \( I \) is capable of producing \( EV_{I'} \) and \( T \).

Now an evidence-based diagnosis algorithm is given only a Testimony graph \( T \) as input. Hence, if there are multiple sufficient hybrid covers \( HC_1,...,HC_q \) of \( T \), there are multiple instances of the system \( I_1,...,I_q \) capable of producing \( T \). Such an algorithm is thus unable to distinguish which particular instance \( I \) of the system produced \( T \). For this reason, a correct evidence-based diagnosis algorithm must only identify a component as faulty if it is faulty in every such instance of the system, since otherwise, the particular instance which produced \( T \) might be an instance in which that component is non-faulty. We now demonstrate these observations formally.

**Lemma 1.**

Let \( I \) be an instance of the system. \( T \) be a Testimony graph produced by \( I \) and \( HC_{i_1},...,HC_{i_q} \) be the sufficient vertex covers of \( T \). Let \( p \) be any non-faulty processor and \((p,r)\) be any non-faulty link in \( I \). Then there is a sufficient hybrid cover \( HC \) of \( T \) such that \( p \notin HC: \nu \) and \((p,r) \notin HC:ec \).

**Proof.** Let \( p \) be a non-faulty processor and \((p,r)\) be a non-faulty link. Let \( Pp \) be the faulty processors and \( Lr \) be the faulty links in \( I \). Consider \( HC_p =<Pp,Lr> \). From P1, for every edge \((x,y)\) in \( T \) either processor \( x \) is faulty, \( y \) is faulty, link \((x,y)\) is faulty or some combination of these. Hence \( HC_p \) is a hybrid cover of \( T \). \( HC_p \) is clearly sufficient, since \(|Pp| \leq t \) and \(|Lr| \leq \lambda \). Since \( p \notin Pp \) and \((p,r) \notin Lr \), the result follows. \( \square \)
Lemma 2.

Let I be an instance of the system, T be a Testimony graph produced by I, and HC_1, ... , HC_h be the sufficient vertex covers of T. Let VC = \bigcap_{i=1}^{h} HC_i.vc and EC = \bigcap_{i=1}^{h} HC_i.ec. Then all processors p \in VC and all links (q,r) \in EC are faulty.

Proof. Assume to the contrary that there is some non-faulty processor p in VC or link (q,r) in EC. Then p or (q,r) must be in all sufficient hybrid covers of T. However, this is a contradiction to Lemma 1, since every non-faulty processor (or link) in instance I is absent from some sufficient hybrid cover of T. □

Lemmas 1 and 2 provide us with the strongest characterizations of the non-faulty, and faulty processor, respectively. The following pseudo-code algorithm expresses these characterizations:

Algorithm D(T, D.P, D.L)

begin

Let HC_1, ... , HC_h be the sufficient hybrid covers of T;

D.P:= \bigcap_{i=1}^{h} HC_i.vc;

D.L:= \bigcap_{i=1}^{h} HC_i.ec;

end;

Returning to Example 1, we see that since HC_1 is the unique sufficient hybrid cover of T, processor 1 must be faulty, and links (2,5) and (3,4) must be faulty. We now show that algorithm D always provides a correct diagnosis.

Theorem 3. (Correctness)

Algorithm D is correct.

Proof. Immediate from Lemma 2. □

To further illustrate the operation of algorithm D, consider the following examples:

Example 2. Consider a system S with N = 7 processors. Let I be an instance of S in which t = 2, \lambda = 1, processors 1 and 2 are faulty, link (3,4) is faulty, and the Testimony graph T of Figure 2 is produced.

\begin{center}
\begin{tikzpicture}
  \node (1) at (0,0) {1};
  \node (2) at (1,0) {2};
  \node (3) at (2,1) {3};
  \node (4) at (2,-1) {4};
  \node (5) at (1,-2) {5};
  \node (6) at (3,-1) {6};
  \node (7) at (3,1) {7};
  \draw (1) -- (2);
  \draw (2) -- (3);
  \draw (2) -- (4);
  \draw (3) -- (7);
  \draw (4) -- (7);
  \draw (5) -- (6);
\end{tikzpicture}
\end{center}

Testimony Graph T: Example 2. Figure 2.

In this graph, the following are some hybrid covers of T:

HC_1 = \langle\{1,2\},\{(3,4)\}\rangle,
HC_2 = \langle\{1,6\},\{(2,5),(3,4)\}\rangle,
HC_3 = \langle\{2,6\},\{(1,7),(3,4)\}\rangle,
HC_4 = \langle\{3,6\},\{(1,7),(2,5)\}\rangle,
HC_5 = \langle\{4,2\},\{(1,6),(1,7)\}\rangle,
HC_6 = \langle\{1,2\},\{(3,4)\}\rangle.

Theorem 3. (Correctness)

Algorithm D is correct.

Proof. Immediate from Lemma 2. □

Example 3. Consider a system S with N = 9 processors. Let I be an instance of S in which t = 2, \lambda = 2, processors 1 and 2 are faulty, links (4,5) and (3,6) are faulty, and the Testimony graph T of Figure 3 is produced.

\begin{center}
\begin{tikzpicture}
  \node (1) at (0,0) {1};
  \node (2) at (1,0) {2};
  \node (3) at (2,1) {3};
  \node (4) at (2,-1) {4};
  \node (5) at (3,0) {5};
  \node (6) at (4,0) {6};
  \node (7) at (5,1) {7};
  \node (8) at (5,-1) {8};
  \node (9) at (6,0) {9};
  \draw (1) -- (2);
  \draw (2) -- (3);
  \draw (2) -- (4);
  \draw (3) -- (7);
  \draw (4) -- (7);
  \draw (5) -- (6);
  \draw (5) -- (8);
\end{tikzpicture}
\end{center}

Testimony Graph T: Example 3. Figure 3.

In this graph, the following are each sufficient hybrid covers:

HC_1 = \langle\{1,2\},\{(3,4)\}\rangle,
HC_2 = \langle\{1,3\},\{(2,7),(4,5)\}\rangle.

Note that if processor 1 is not included in the vc component of a hybrid cover HC_i, then regardless of which two other vertices are included, HC_i.ec must include three edges, and hence cannot be sufficient. A similar argument applies for edge (4,5). Hence algorithm D, given this Testimony graph will identify processor 1 and link (4,5) as faulty, but is unable to (correctly) distinguish whether 2 and (3,6) are faulty, or 3 and (2,7) are faulty.

Lemma 2 informs us that any component identified as faulty by algorithm D is indeed faulty. However, Example 3 illustrates that D cannot always identify all of the faulty components. Our next result addresses the following question: what can be said about a faulty component not caught by D?

Lemma 4.

Let I be an instance of the system, T be a Testimony graph produced by I and HC_1, ... , HC_h be the sufficient hybrid covers of T. Let p be a faulty processor in I, such that in execution \langle T,D.P,D.L \rangle of algorithm D, p \not\in D.P. Then there is an instance of the system I' capable of producing T' = T, in which p is non-faulty.

Proof. Since p is not caught by D, there must be some sufficient hybrid cover HC_i, such that p \not\in HC_i.vc. Now consider an instance of the system I' in which every processor q \in HC_i.vc is faulty, and every link \langle x,y \rangle \in HC_i.ec is faulty. I' clearly exists, since there are i \geq t faulty processors and j < \lambda faulty links. Moreover, since HC_i covers T, I' is capable of producing T, from which the result follows. □
Theorem 6. (Optimality)
Let I be an instance of the system, T be a Testimony graph produced by D, and \(<T,A,P,A,L>\) be an execution of any correct evidence-based diagnosis algorithm A. Then there is an execution of \(<T,D,P,D,L>\), such that \(A \subseteq D \cap P\) and \(A \subseteq D\).

Proof. Assume to the contrary that there is a correct evidence-based diagnosis algorithm A, and executions \(<T,A,P,A,L>\) and \(<T,D,P,D,L>\), such that \(2 \not\in A \subseteq D\). (i.e. there is some processor which A identifies as faulty that D does not). If p is non-faulty, then A is clearly not correct, so assume that p is faulty.

By Lemma 5, there is some instance of the system I in which p is non-faulty, and Testimony graph T' = T is produced by that instance. Now consider the execution \(<T,P,A',P,A,L'>\). Since A is deterministic, A' = A, hence p \(\not\in A\). But p is non-faulty in I', which contradicts our assumption that A is correct. We conclude that no A exists. Hence in the worst case, \(t+\lambda>\lambda\).

A similar argument for links suffices to demonstrate the result for A.L and D.L, and is omitted.

From Theorem 6, we see that algorithm D is in fact the best possible, since no other evidence-based diagnosis algorithm can identify more faulty components than D does. In Example 2, we saw a Testimony graph where D was unable to form a complete diagnosis. From Theorem 6, it follows that no other evidence-based algorithm can either, and hence no evidence-based diagnosis algorithm which guarantees correctness can also guarantee completeness. More formally, we have

Theorem 7. (Incompleteness)
Let A be any correct, evidence-based diagnosis algorithm for a system with \(t>0\) and \(\lambda\geq0\). Then A cannot guarantee completeness.

Proof. Consider an instance I of a system with \(t=1\) and \(\lambda=0\), which produces a Testimony graph T containing one edge \((p,q)\). Clearly there are two sufficient hybrid covers of T: \(HC_1=\{(p,q)\}\), and \(HC_2=\{t\}\). Since \(HC_1 \cap HC_2 = \emptyset\), D can identify no faulty processors in execution \(<T,D,P,D,L>\). By PI, either p or q is faulty, hence D's diagnosis is not complete in \(<T,D,P,D,L>\) by Theorem 7, no other algorithm can do better.

Thus, we see that no evidence-based diagnosis algorithm can guarantee that its diagnosis is both correct and complete. The reason stems from the possibility of false evidences. That is, for an edge \((p,q)\) in T, the possibility exists that the evidence \([p,\langle p,q \rangle]\) from which \((p,q)\) was formed was false. Without further information, there is no basis for a correct algorithm to decide whether p or q (or \(\langle p,q \rangle\)), if \(\lambda>0\), is faulty.

Note that if \(t>0\) and \(\lambda>0\), even the knowledge that \([p,\langle p,q \rangle]\) is not false is insufficient to permit a correct algorithm to guarantee completeness, since such an algorithm would be unable to determine whether p or \((p,q)\) was the faulty component responsible for an edge \((p,q)\) in T. In the worst case. However, given a system S in which false evidences cannot occur (i.e. every evidence is true), a processor p need not be included in the Testimony graph T if it is an evidence \([p,\langle p,q \rangle]\) (i.e. an evidence in which the p is the accused processor). If there is no such evidence, then (even if there are evidences \([\tau,\langle \tau,p \rangle]\)) p need not be included in T. Hence there may be edges in T with only one vertex. With such a Testimony graph as input, algorithm D will again provide an optimal diagnosis.

Note also that if S is a system in which evidences cannot be hidden (as in [21]) the only effect is that more evidences will be present in EV, and hence more edges present in T.

Our next results examine worst-case scenarios for evidence-based diagnosis algorithms, by examining Testimony graphs containing worst-possible conditions. Our first result bounds the number of (unhidden) evidences a single faulty processor can be involved in and remain uncaught.

Theorem 8.
Let I be an instance of a system S, in which \(\left|\{P_f\} \mid f \in t\right| = 1 \leq \lambda\). Let T be a Testimony graph produced by D. Then a processor p \(\epsilon P_f\) can have degree \(d = \lambda+1\) in T and remain uncaught.

Proof. Let processor p and instance I be as described above, and let I be an instance in which the following are true:
- \(\forall (q,r) \in \ell, r = p \) and \(q \not\in P_f\) (that is, every faulty link has p and some non-faulty processor as its endpoints); and
- \(\forall q \in P_f: \exists (p,q) \in T\) (that is, every faulty processor accuses or is accused by p).

Then p is not caught by an execution of D iff we can construct a sufficient hybrid cover \(HC_j\), such that \(p \not\in HC_j\).

Let \(W = \{q : (q,p) \in \ell, q \not\in P_f\}\). Then \(W = t+\lambda-1\) and \(|Y| = \lambda\). Now form \(HC_j\), such that \(HC_j \cap HC_j = \emptyset, HC_j \cap HC_j = \emptyset\). Clearly, every edge \(T_j\) in \(HC_j\) has degree \(d = t+1\) and \(HC_j \in (\lambda-1)\) for all faulty processors being identified in \(HC_j\). Hence \(HC_j\) is an edge in \(HC_j\) for all faulty processors being identified. Note that in this construction, a single additional evidence involving p (i.e. \(d+t+\lambda+1\)) will result in algorithm D catching p, since either \(\langle HC_j \rangle > t\), or \(\langle HC_j \rangle \in \lambda\) in such a circumstance. Hence in the worst case, \(t+\lambda+1\) evidences are required to identify a faulty processor. The obvious best case to identify a faulty processor requires two evidences, as in Example 2. Our next result examines the number of evidences needed to identify all faulty processors.

Theorem 9.
Let I be an instance of a system S, let \(P_f\) be the faulty processors in I, and let T be a Testimony graph produced by D. Then Testimony may have \((t+1)(N-t)+C_3\) edges without all faulty processors being identified.

Proof. Consider an execution E in each of which the \(\lambda\) faulty links connect two non-faulty processors. Let each link \((q,r)\) with \(q \not\in \{P_f\}\) fault so that edge \((q,r)\) is in T, giving \(\lambda\) edges. An adversary wishing to maximize the number of edges in Testimony might have t-1 faulty processors each fault to all of the N-1 non-faulty processors in S, providing \((t-1)(N-1)\) edges in T. If each faulty processor also accuses every other faulty processor, that produces \(C_3\) (the number of combinations of t processors taken 2 at a time) additional edges in T. These t-1 faulty processors will thus be in every sufficient vertex cover and can be identified by D as faulty.

Now consider the remaining faulty processor p. If p faults to t non-faulty processor s, p remains uncaught, since \(HC_j = \langle P_f, \ell, HC_j \rangle\) is a sufficient hybrid cover of T, and for any faulty link \((q,r)\), \(HC_j = \langle P_f, (q,r), HC_j \rangle\) is another. Hence \(HC_j\) is a sufficient hybrid cover of T no containing, and D cannot identify p as faulty. Note also that since \(HC_j\) maybe constructed for any faulty edge \((q,r)\), that no faulty edges can be identified by D, either.

Then T has \((t+1)(N-1)+C_3+\lambda+1\) edges, and p (and the faulty links) remains uncaught.
Note that involvement in a single additional evidence is sufficient for p to be caught, as before. Note also that this will result in every faulty edge being identified, since A sufficient HC will be impossible to construct. Thus, \( t(N-1)+N_2+2 \) bounds the number of edges in T required to identify all faulty components. If this is the worst case, what is the best case to identify all faulty components? We answer this question indirectly, by first determining the minimum number of evidences needed to identify any faulty component.

**Theorem 10.**

Let I be an instance of a system S, and T a Testimony graph produced by I. Then if T has \( t+\lambda \) or fewer edges, no faulty component can be identified.

**Proof.** Consider an arbitrary T of \( t+\lambda \) edges. Let \((p,q)\) be an edge within T. Then clearly sufficient hybrid covers HC, ..., HC can be formed such that \( p \in HC_{i}, q \in HC_{j}, \) and \( (p,q) \in HC_{k} \). Since this construction applies to any edge \((p,q)\) in T, no faulty components can be identified using T. \( \square \)

Theorem 10 is intriguing in light of Example 2, where \( t+\lambda+2 \) edges are sufficient to identify all faulty components in the instance. Thus, it can be shown that not only are \( t+\lambda+1 \) evidences necessary to identify any faulty components, \( t+\lambda+2 \) are sufficient to identify all faulty components, in the best case.

In the next section, we detail an experimental implementation, in which we examine the complexity of algorithm D in a system in which only processors may fail.

**4. Experimental Implementation**

In this section, we detail an experimental implementation of algorithm D for use in systems where \( \lambda = 0 \). In such systems, our use of hybrid covers reduces to the use of vertex covers. While the vertex cover problem is NP-Complete, the cost of a vertex cover still further, since the size of T' is at most \( \lambda^2 - \lambda \). Also, it is easy to show that if no vertex of maximum degree \( d \) can be identified as faulty in this manner, then no vertex of degree \( d' < d \) can be either. Hence, this optimization permits early stopping, when no vertex of maximum degree \( d \) can be identified as faulty.

In implementing algorithms which utilize these optimizations, we have chosen to test their effectiveness using three different versions, where version 1 utilized only optimization 1, version 2 utilized optimizations 1 and 2, and version 3 utilized optimizations 1, 2 and 3. In doing so, it was our purpose to test the optimizations against each other, and see if any significant performance differences could be observed. Our experimental hypothesis was that (in the worst case) each version would show exponential growth with \( t \), but only quadratic growth with \( N \). This hypothesis stems from our observations that for a fixed \( t \), the cost of \( O(2^t) \) becomes a "constant" cost. Hence as \( N \) grows, the \( O(N^2) \) cost of 2-coloring (for versions 2 and 3) will ultimately exceed the cost of vertex covering, giving a quadratic cost for the algorithm. We thus predict worst-case performance of \( O(N^2) \) for versions 2 and 3, but \( O(2^t) \) for version 1. The next section details this experiment.

**4.2. Experimental Results**

In discussing our optimizations, we have discussed upper bounds which can be expected, if faults occur in intelligent patterns. In this experimental environment, we wished to examine the behavior of our algorithm when this is not the case, that is, when random faultiness is likely.

We implemented the three versions of our algorithm using a Sun 3/60 as a centralized diagnostic processor for the processing of Testimony graphs. Such graphs were generated by randomly designating \( 1 \) to \( t \) processors as faulty, and having each faulty processor fault randomly to \( 1 \) to \( N \) other processors.

For each version and combination of \( N \) and \( t \) considered, we generated 200 different Testimony graphs to be processed. Each graph was processed and the required processing time was noted (in milliseconds). Using these recorded times, the average and worst-case processing times were determined for each combination of \( N \) and \( t \), and for each version.

We began by fixing \( N \) and varying \( t \). Since the maximum size of the graphs is \( O(t^2) \), we would expect the worst case cost to be exponential as \( t \) increases. We performed four different experiments, fixing \( N \) at 15, 25, 50 and 100 respectively. In each experiment, \( t \) was varied over a wide range of values. Because of space limitations, we give our results for \( N = 50 \) here (see Figure 4), and the interested reader is referred to to [1] for the details of the others.

Here, we observe significant differences occurring not only in the worst case, but also in the average case. In each of our experiments, these differences occur as \( t \) approaches \( N/2 \). Note in particular that when \( t = 20 \), the worst case performance of version 3 is nearly as good as the average performance of version 2, and far superior to the average performance of version 1. Thus, optimization 3 appeared invaluable in our experiments.
To test these observations further, we performed three additional experiments in which \( t \) was fixed and \( N \) varied. According to our hypothesis, we should observe quadratic growth in (at least) versions 2 and 3. In these experiments, \( t \) was fixed at 5, 10 and 15 respectively, and \( N \) was varied through a wide range of values. We include our results for \( t = 15 \), as illustrated in Figure 5.

Here, we observe quadratic growth, particularly on average. Note also that version 1, without either optimization 2 or 3 experiences some faster growth than its peers when \( N = 300 \). Our other experiments yielded similar results.

Thus, our experimental data generally affirms our hypothesis. We conclude that the cost of our algorithm (when \( \lambda = 0 \)) can be reduced to \( O(\max(2^kN^2)) \).

5. Conclusions

We have presented a new formal model of evidence-based diagnosis algorithms which models any algorithm whose diagnosis is based solely upon evidences. Using this model, we have constructed a correct diagnosis algorithm which is demonstrably optimal. We further show that in the worst case, no evidence-based diagnosis algorithm can guarantee that its diagnosis is both correct and complete, when evidences can be false.

We have also argued, both experimentally and analytically that the complexity of this algorithm (when only processors are faulty) is given by \( O(\max(2^kN^2)) \), and hence is practical for systems in which \( N/t \) is sufficiently large.

In discussing our optimizations, optimization 2 (using 2-colorability to identify bipartite Testimony graphs) is particularly interesting, for the following reason: if the graph is bipartite (where no faulty processor accuses another faulty processor), the graph may be processed more quickly than if this is not the case. Hence, if the faults are intelligently patterned (as in byzantine faultiness), a trade-off exists for the faulty processors: if they fault only to non-faulty processors, the overhead of processing the Testimony graph is reduced to quadratic in \( N \), while if they fault to each other, they can fault to fewer non-faulty processors without exceeding \( t \), restricting their faulty behavior. Thus, optimizations 1 and 2 together combine to form a "dilemma" for byzantine processors, while optimization 3 serves to permit faster vertex covering by reducing the size of the graph and permitting early stopping.

Future work includes the extension of this implementation to systems in which \( \lambda > 0 \), and the development of optimizations for such systems equivalent to those presented here. By first checking for processors involved in more than \( t + \lambda \) evidences, and removing such processors from the Testimony graph, it appears that the complexity of such algorithms can be reduced to \( O(2^{t+\lambda}) \), in the worst case.

We also intend to investigate the construction of algorithms which utilize more complex forms of evidences.

References


