

COMPUTATION OF FAULT DETECTION DELAY IN DISCRETE-EVENT SYSTEMS

Tae-Sic Yoo and Humberto E. Garcia *

* *Systems Analysis and Control Group*
Argonne National Laboratory
P.O. Box 2528, Idaho Falls, ID 83403-2528
{tyoo, garcia}@anlw.anl.gov

Abstract: The notion of diagnosability based on failure-event specifications is revisited. We present a modified version of diagnosability in which terminating faulty traces are handled differently. We also introduce the notion of language-diagnosability based on failure-language specifications that generalizes diagnosability based on failure-event specifications. A polynomial-time algorithm for verifying language-diagnosability is developed. Building upon the verification algorithm, we introduce a polynomial-time algorithm for computing the worst case diagnosis delay of a given system. Despite of significant practical importance, this delay computation has not been previously considered in the literature. The computation of the worst case diagnosis delay involves the shortest path computation of a weighted, directed graph. We exploit a special weighting structure of the graph resulting from the verification algorithm, which enables an algorithm with a lower complexity than the commonly used Bellman-Ford shortest path algorithm.

Keywords: Discrete-Event Systems, Fault Diagnosis, Fault Detection Delay

1. INTRODUCTION

The objective of the failure diagnosis is to monitor the system behavior and detect and identify abnormalities in the behavior of the system under partial observations. In (Sampath *et al.*, 1995), the definition of diagnosability based on failure-event specifications was first introduced. The property of diagnosability in (Sampath *et al.*, 1995)¹ is related to the ability to infer, from observed event sequences, about the occurrence of certain events (the “fault” events). Polynomial-time algorithms

for verifying the property of diagnosability are reported in (Jiang *et al.*, 2001; Yoo and Lafortune, 2002) independently. Some variations of diagnosability of (Sampath *et al.*, 1995; Sampath *et al.*, 1998) were proposed recently. Instead of failure events, failure states were employed in (Zad, 1999) and the corresponding notion of diagnosability was characterized. In (Jiang and Kumar, 2002), the problem of failure diagnosis was studied in the framework of temporal logic. More recently, the issues of intermittent and repeated faults were addressed in (Contant *et al.*, 2002) and (Jiang *et al.*, 2002).

However, while the current literature considers the existence of a finite fault diagnosis delay, no explicit algorithm has been provided to determine its exact value. Only a conservative upper bound is given, equal to the square of the number of

¹ Note that two conditions on system behaviors, liveness and unobservable-cycle freeness, were assumed in (Sampath *et al.*, 1995). In (Sampath *et al.*, 1998), liveness assumption was relaxed and diagnosability accounting for terminating traces was defined. We will use this relaxed version of diagnosability when we refer the definition of diagnosability based on failure-event specifications.

states describing the system behavior (Yoo and Lafortune, 2002). Many applications, however, require the exact determination of the diagnosis delay in order to assure that any fault can be diagnosed within a specified limit. Failure to meet delay response requirements may lead to the re-design of the observation mask.

Motivated by this practical importance, this paper introduces an algorithm that computes the worst case diagnosis delay explicitly. The detailed contributions of this paper are:

- In Section 2.1, we first revisit the definition of diagnosability of (Sampath *et al.*, 1995; Sampath *et al.*, 1998). We present a modified version of diagnosability where terminating traces are handled differently. The justification of this modification is given in details with an example.
- The diagnosability of (Sampath *et al.*, 1995; Sampath *et al.*, 1998) is based on the failures that are specified with “events”. In Section 2.2, we introduce the notion of language-diagnosability that specifies the failures as “languages”. In this manner, language-diagnosability generalizes the diagnosability of (Sampath *et al.*, 1995; Sampath *et al.*, 1998). The modification conducted in Section 2.1 is also applied to language-diagnosability.
- In Section 3.1, a polynomial-time algorithm for verifying language-diagnosability is developed. This algorithm accounts for the existence of terminating traces and unobservable cycles.
- Building upon the results in Section 3.1, a polynomial-time algorithm computing the exact worst case diagnosis delay is developed in Section 3.2.

Due to page limitations, we omit the proofs of technical results in the main presentation. The proofs and additional illustrative examples can be found in (Yoo and Garcia, 2003).

2. NOTIONS OF DIAGNOSABILITY

We model the untimed discrete-event system as a deterministic finite-state automaton: $A = (Q^A, \Sigma^A, \delta^A, q_0^A)$ where Q^A is the finite state space, Σ^A is the set of events, and q_0^A is the initial state of the system. δ^A is the partial transition function and $\delta^A(q_1, \sigma) = q_2$ implies the existence of a transition from state q_1 to state q_2 with event label σ . The superscript A may be dropped if this is not likely to cause confusion. The language generated by A is denoted by $\mathcal{L}(A)$ and is defined in the usual manner (Cassandras and Lafortune, 1999).

To reflect limitations on observation, we define the observation mask function $M : \Sigma^A \rightarrow \Delta^A \cup \{\epsilon\}$

where Δ^A is the set of observed symbols and it may be disjoint with Σ^A . The definition of M can be extended to sequences of events (traces) inductively as follows: $\forall s \in (\Sigma^A)^*$, $\forall \sigma \in \Sigma^A$, $M(s\sigma) = M(s)M(\sigma)$.

2.1 Event-Diagnosability

Consider a language L and a mask function M^2 over the events defined in L , denoted by Σ_L . In the context of failure diagnosis, let $\Sigma_f \subseteq \Sigma_L$ denote the set of failure events which should be diagnosed. The set of failure events is partitioned into disjoint sets corresponding to different failure types: $\Sigma_f = \Sigma_{f_1} \dot{\cup} \dots \dot{\cup} \Sigma_{f_m}$. We denote this partition by Π_f . The formal definition of diagnosability was first presented in (Sampath *et al.*, 1995; Sampath *et al.*, 1998). In order to highlight the objective of this version of diagnosability, identifying the occurrence of the failure “events”, we will call this notion of diagnosability as *event-diagnosability* hereafter. Note that we do not assume that failure events are unobservable. The failure events may be observable but may not have an unique observation symbol under the mask function M .

In order to define event-diagnosability, we need the following notation. We will write $s \in \Psi(\Sigma_{f_i})$ to denote that the last event of a trace $s \in L$ is a failure event of type Σ_{f_i} . That is,

$$\Psi(\Sigma_{f_i}) := \{s = t\sigma_f \in L : \sigma_f \in \Sigma_{f_i}\}.$$

We denote by L/s the postlanguage of L after s , i.e. $L/s := \{t \in (\Sigma_L)^* : st \in L\}$. With slight abuse of notation, we write $\Sigma_{f_i} \in s$ to denote that $\bar{s} \cap \Psi(\Sigma_{f_i}) \neq \emptyset$. We are now ready to state the definition of event-diagnosability introduced in (Sampath *et al.*, 1995; Sampath *et al.*, 1998).

Definition 2.1. A prefix-closed language L is said to be event-diagnosable with respect to a mask function M and Π_f on Σ_f if the following holds:

$(\forall i \in \Pi_f)(\exists n_{d_i} \in \mathbb{N})(\forall s \in \Psi(\Sigma_{f_i}))(\forall t \in L/s)$
 $[(|t| < n_{d_i})(L/st = \emptyset) \Rightarrow D_1] \wedge [|t| \geq n_{d_i} \Rightarrow D_2]$
 where \mathbb{N} is the set of non-negative integers and the event-diagnosability conditions D_1 and D_2 are

$$D_1 : (\forall w \in M^{-1}M(st) \cap L) \\
 [\text{if } L/w = \emptyset \Rightarrow \Sigma_{f_i} \in w] \text{ and} \\
 D_2 : (\forall w \in M^{-1}M(st) \cap L) [\Sigma_{f_i} \in w].$$

The condition D_1 implies that even if the language L has a terminating trace st that ends in a failure event of type Σ_{f_i} , L may still be diagnosable as

² In (Sampath *et al.*, 1995; Sampath *et al.*, 1998), the plain projection function P is used instead of the (non-projection) mask function M .

long as there does not exist in L a trace $s't'$ such that $s't'$ is also terminating and generates the same masked observation as the trace st but does not contain a failure event of type Σ_{f_i} . Though it is not mentioned in (Sampath *et al.*, 1998), note that the above condition should rely on the implicit assumption that the termination of faulty traces is detectable. This may not be an adequate assumption since the termination of traces is not always possible to detect under partial observations. In Fig. 1, we present a simple example in order to clarify our arguments. We set that $\{f\} \in \Sigma_f$ and $M(f) = \epsilon$. Then, let

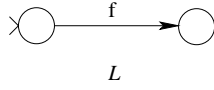


Fig. 1. Diagnosable or not?

us examine if L is event-diagnosable with the proposed setting. If we follow the above definition of event-diagnosability, L is diagnosable since f is the terminating faulty trace, the only terminating trace looking identical to f is f itself, and $\Sigma_f \in f$. However, since $M(f) = \epsilon$, there is no way of knowing if the system has executed f . The definition of event-diagnosability makes sense if we assume that the termination after executing event f is somehow detectable (e.g. time-out). However, it may imply that the proposed model does not sufficiently reflect the behaviors to be modelled and a more complete model may be live itself (by adding self-loops of termination detection events to terminating states). With this, we believe that it is more adequate to classify the model in Fig. 1 as undiagnosable.

In order to account for this observation and reach a proper diagnostic decision with terminating faulty traces, the traces to be examined should include not only terminating look-alike traces but also nonterminating look-alike traces. Particular, in the above example, $\epsilon \in L$ is a nonterminating non-faulty trace and $M(\epsilon) = M(f)$.

The graphical description of general situation is depicted in Fig 2. If the trace executed by the system is s_2 , we may be able to decide if the behavior is faulty or non-faulty after the system executes more events. However, if the trace executed by the system is s_0 or s_1 , we cannot decide if the behavior is faulty or non-faulty indefinitely since we cannot know if the system is terminated or not in the first place. This situation is more appropriate to be classified as the violation of event-diagnosability.

Reflecting this observation, we modify the definition of event-diagnosability as follows.

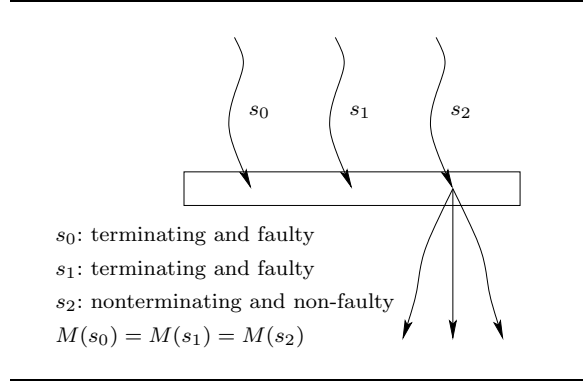


Fig. 2. Violation of diagnosability: case of terminating faulty trace

Definition 2.2. (Modified Event-Diagnosability)

A prefix-closed language L is said to be event-diagnosable with respect to a mask function M and Π_f on Σ_f if the following holds:

$$(\forall i \in \Pi_f)(\exists n_{d_i} \in \mathbb{N})(\forall s \in \Psi(\Sigma_{f_i}))(\forall t \in L/s) [\{(L/st = \emptyset) \vee (|t| \geq n_{d_i})\} \Rightarrow D_e]$$

where \mathbb{N} is the set of non-negative integers and the event-diagnosability condition D_e is

$$D_e : (\forall w \in M^{-1}M(st) \cap L) [\Sigma_{f_i} \in w].$$

The modified definition of event-diagnosability also considers two cases.

(i) If a faulty behavior st of type Σ_{f_i} is terminating ($L/st = \emptyset$), then *all* possible behaviors generate the same masked observation as st should be faulty in terms of type Σ_{f_i} (the condition D_e).

(ii) If suffixes of faulty behavior regarding type Σ_{f_i} are long enough ($|t| \geq n_{d_i}$), then they should contain enough information to indicate that all possible behaviors generate the same masked observation as st should be faulty in terms of type Σ_{f_i} .

Note that the condition D_1 of Definition 1 examines if there is a trace s' such that s' is also *terminating* and generates the same masked observation as the trace s but does not contain a failure event of a certain type. In order to accommodate the observation that the termination of traces may not be detectable, in the new definition, all possible (terminating and nonterminating) look-alike behaviors are examined.

In the definition of event-diagnosability, the failures are specified with some special “events”. Depending on applications and models of system behaviors, faults can be easily represented as sequences of events but not as a certain set of events. In order to account for this observation, we will introduce the notion of *language-diagnosability* where the failures are specified with “languages” rather than “events” in the following section. In this manner, we generalize the notion of event-diagnosability naturally.

2.2 Language-Diagnosability

We present the notion of language-diagnosability where the failures are specified with the set of languages as provided below.

Definition 2.3. A set of prefix-closed languages $\mathcal{L}_f := \{L_i : L_i \subseteq L \text{ for } i = 1, \dots, m\}$ is said to be language-diagnosable with respect to a prefix-closed language L , and a mask function M over the events defined in L if the following holds:

$$\begin{aligned} & (\forall i \in \{1, \dots, m\})(\exists n_{d_i} \in \mathbb{N}) \\ & (\forall s \in L \setminus L_i)(\forall t \in L/s) \\ & [\{(L/st = \emptyset) \vee (|t| \geq n_{d_i})\} \Rightarrow D_i] \end{aligned}$$

where \mathbb{N} is the set of non-negative integers and the condition D_i is

$$D_i : M^{-1}M(st) \cap L_i = \emptyset.$$

The worst case diagnosis delay of \mathcal{L}_f with respect to L and M is defined as follows:

$$d_{dia} = \max\{\min(n_{d_i}) : i \in \{1, \dots, m\}\}.$$

Provided with d_{dia} , we call that \mathcal{L}_f is d_{dia} -step language-diagnosable w.r.t. L and M .

In the above definition, the languages \mathcal{L}_f and L represent the set of non-faulty behaviors and the possible behavior, respectively. Naturally, the faulty (or abnormal) behavior regarding type i is represented by $L \setminus L_i := L \cap L_i^c$.

The notion of language-diagnosability provides an unspecified finite-step diagnosis delay. In practice, it is desirable to know the exact diagnosis delay for assuring timely responses to the identified failures. The notion of d_{dia} -step language-diagnosability is defined in order to reflect the practical importance of response delay assurance.

Before we move to the next section, we present the following proposition that enables the modular verification of language-diagnosability.

Proposition 2.1. A set of prefix-closed languages $\mathcal{L}_f = \{L_i : L_i \subseteq L \text{ for } i = 1, \dots, m\}$ is language-diagnosable with respect to a prefix-closed language L , and a mask function M over the events defined in L iff $\{L_i\}$ is language-diagnosable with respect to L and M for all $i \in \{1, \dots, m\}$.

3. COMPUTATION OF THE WORST CASE DIAGNOSIS DELAY

In this section, we are interested in computing the worst case diagnosis delay d_{dia} . It is clear that $\min(n_{d_i})$ can be computed separately for each $i \in \{1, \dots, m\}$. Upon obtaining $\min(n_{d_i})$ for each i , computing d_{dia} is straightforward. Therefore, we

will concentrate on the computation of $\min(n_{d_i})$ afterward.

We assume that the non-faulty behavior and the possible behavior are generated by trim finite-state deterministic automata $N = (Q^N, \Sigma^N, \delta^N, q_0^N)$ and $P = (Q^P, \Sigma^P, \delta^P, q_0^P)$, respectively, where $\mathcal{L}(N) \subseteq \mathcal{L}(P)$. First we investigate the issue of the existence of a finite diagnosis delay in the following section. Then, building upon the results of the following section, an algorithm computing the worst case diagnosis delay will be developed in Section 3.2.

3.1 Existence of Finite Diagnosis Delay

Let N and P be two finite-state automata such that $\mathcal{L}(N) \subseteq \mathcal{L}(P)$, and let M be a mask function for events defined over Σ_P . Remind that P and N may not be live.

Now, we construct a weighted, directed graph $G(N, P, M) = (V(N, P), E(N, P, M))$. For notational convenience, we may drop the dependency notation of $G(N, P, M)$, when it is considered to be clear from the context. The set of vertexes V is

$$\begin{aligned} V \subseteq & Q^N \times Q^N \times Q^P \times \{non\text{-faulty}, confused\} \\ & \cup \{Block\}, (q_0^N, q_0^N, q_0^P, non\text{-faulty}) \in V, \end{aligned}$$

and a weight function w is defined as $w : E \rightarrow \{-1, 0\}$. The implication of the weighted, directed graph G will be explained after we complete the description of G .

Before we proceed to define the edges of G , for the sake of readability, let us define the following transition notation:

$$\delta^N(q_1, \sigma') = q'_1, \delta^N(q_2, \sigma) = q'_2, \text{ and } \delta^P(q_3, \sigma) = q'_3.$$

Note that we use event σ to define q'_2 and q'_3 . On the other hand, event σ' is used to define q'_1 . Also, observe that σ and σ' can be identical.

The notation $p \xrightarrow{i} q$ below implies that there is an edge $(p, q) \in E$ with weight candidate $i \in \{-1, 0\}$. The weight of edge $(p, q) \in E$ will be determined by choosing the minimum of weight candidates defined over edge (p, q) . Now we define edges with weight candidates as follows.

For $\sigma', \sigma \in \Sigma_P$ such that $M(\sigma') = M(\sigma) = \epsilon$,

$$(q_1, q_2, q_3, non\text{-faulty}) \xrightarrow{0} (q'_1, q_2, q_3, non\text{-faulty}) \quad (1) \\ \text{if } q'_1 \text{ is defined}$$

$$(q_1, q_2, q_3, non\text{-faulty}) \xrightarrow{0} (q_1, q'_2, q'_3, non\text{-faulty}) \quad (2) \\ \text{if } q'_2 \text{ and } q'_3 \text{ are defined}$$

$$(q_1, q_2, q_3, non\text{-faulty}) \xrightarrow{-1} (q_1, q_2, q'_3, confused) \quad (3) \\ \text{if } q'_2 \text{ is not defined but } q'_3 \text{ is defined}$$

$$(q_1, q_2, q_3, confused) \xrightarrow{0} (q'_1, q_2, q_3, confused) \quad (4)$$

if q'_1 is defined

$$(q_1, q_2, q_3, \text{confused}) \xrightarrow{-1} (q_1, q_2, q'_3, \text{confused}) \quad (5)$$

if q'_3 is defined

For $\sigma', \sigma \in \Sigma_P$ such that $M(\sigma') = M(\sigma) \neq \epsilon$,

$$(q_1, q_2, q_3, \text{non-faulty}) \xrightarrow{0} (q'_1, q'_2, q'_3, \text{non-faulty}) \quad (6)$$

if q'_1, q'_2 , and q'_3 are defined

$$(q_1, q_2, q_3, \text{non-faulty}) \xrightarrow{-1} (q'_1, q_2, q'_3, \text{confused}) \quad (7)$$

if q'_1 and q'_3 are defined but q'_2 is not defined

$$(q_1, q_2, q_3, \text{confused}) \xrightarrow{-1} (q'_1, q_2, q'_3, \text{confused}) \quad (8)$$

if q'_1 and q'_3 are defined

The edges to *Block* vertex are defined as follows.

$$(q_1, q_2, q_3, \text{confused}) \xrightarrow{0} \text{Block} \quad (9)$$

if, $\forall \sigma \in \Sigma_P$, q'_3 is not defined

With the above definition, it is possible to have an edge that has two different weight candidates, “-1” and “0”. In this case, we choose “-1” as the weight of the edge. Hereafter, we only consider the accessible part of the weighted, directed graph G from the vertex $(q_0^N, q_0^N, q_0^P, \text{non-faulty})$ when G is referred.

Now, we explain the implication of G . The weighted, directed graph G is designed to track traces $s' \in \mathcal{L}(N)$ and $s \in \mathcal{L}(P)$ such that $M(s') = M(s)$ from the vertex $(q_0^N, q_0^N, q_0^P, \text{non-faulty})$. Specifically, the vertex space and the edge relation are defined to track the traces in the following manner:

$$\underbrace{Q^N}_{s'} \times \underbrace{Q^N \times Q^P}_s \times \{\text{non-faulty, confused}\}.$$

The structure of edge definition is similar to the transition relation of F_i -verifier in (Yoo and Lafortune, 2002). Observe that the *indicator* set $\{\text{non-faulty, confused}\}$ is designed to show whether trace s is in non-faulty behavior $\mathcal{L}(N)$ or faulty behavior $\mathcal{L}(P) \setminus \mathcal{L}(N)$. Note that the same event is used to define q'_2 and q'_3 . Therefore, the second (Q^N) and the third (Q^P) state spaces of V track s simultaneously as long as $s \in \mathcal{L}(N)$ and the indicator remains at “non-faulty”. The change from “non-faulty” to “confused” occurs when q'_2 is not defined but q'_3 is defined. In other word, s becomes faulty, that is, $s \in \mathcal{L}(P) \setminus \mathcal{L}(N)$ at that moment. After s becomes faulty, we only need to update $q_3 \in Q^P$. That is the reason why $q_2 \in Q^N$ with “confused” indicator is not updated further.

Also note that the weight candidate “-1” is assigned only if q'_3 is defined and the vertex reached by the edge has “confused” indicator. Along with the edge weight selection rule choosing the minimum value of edge candidates, if we encounter edges with the weight “-1”, then it is clear to see that the edges are for updating faulty trace s . On the other hand, it is also clear to see that edges with the weight “0” are for updating non-faulty

traces s' or s . We define the following terminology for further arguments.

Definition 3.1. We say that $\{v_1, v_2, \dots, v_n\} \subseteq V$ form a path, denoted by $\langle v_1, v_2, \dots, v_n \rangle_G$, if there are edges such that $v_1 \xrightarrow{w_1} v_2 \xrightarrow{w_2} \dots \xrightarrow{w_{n-1}} v_n$. We say that a path, $\langle v_1, v_2, \dots, v_n \rangle_G$, forms a cycle if $v_1 = v_n$ and at least one edge is contained along the path.

With G , we can claim the following result.

Theorem 3.1. Given the two automata N, P , and the mask function M , $\{\mathcal{L}(N)\}$ is not language-diagnosable w.r.t. $\mathcal{L}(P)$ and M iff there is a cycle of $G(N, P, M)$ that has an edge with the negative weight or the *Block* vertex is reachable from $(q_0^N, q_0^N, q_0^P, \text{non-faulty})$.

Let $|Q^N| = n_1$, $|Q^P| = n_2$, and $|\Sigma_P| = n_3$. The following result shows that the verification of language-diagnosability can be done in polynomial time.

Theorem 3.2. The language-diagnosability of $\{\mathcal{L}(N)\}$ with respect to $\mathcal{L}(P)$ and a mask function M over the events defined in $\mathcal{L}(P)$ can be decided in $O(n_1^2 \cdot n_2 \cdot n_3^2)$.

With Proposition 2.1 and Theorem 3.1, the following can be shown straightforwardly.

Theorem 3.3. The language-diagnosability of $\mathcal{L}_f = \{\mathcal{L}(N_1), \dots, \mathcal{L}(N_m) \subseteq \mathcal{L}(P)\}$ with respect to $\mathcal{L}(P)$ and a mask function M over the events defined in $\mathcal{L}(P)$ can be decided in $O((|Q^{N_1}|^2 + \dots + |Q^{N_m}|^2) \cdot n_2 \cdot n_3^2)$.

Now we compute the worst case diagnosis delay in the following section.

3.2 Computation of Worst Case Diagnosis Delay

Although we have the result of finite diagnosis delay, it is important to know the exact worst case diagnosis delay in practice. Knowing that language-diagnosability holds, an upper bound for finite diagnosis delay, $|Q^N \times Q^N \times Q^P|$, can be computed applying a similar technique used for Proposition 1 of (Yoo and Lafortune, 2002). However, it is desirable to have the exact determination of the diagnosis delay in order to assure that any fault can be diagnosed within a specified limit.

All prior work regarding failure diagnosis of discrete-event systems listed in the references address the problem of deciding the existence of

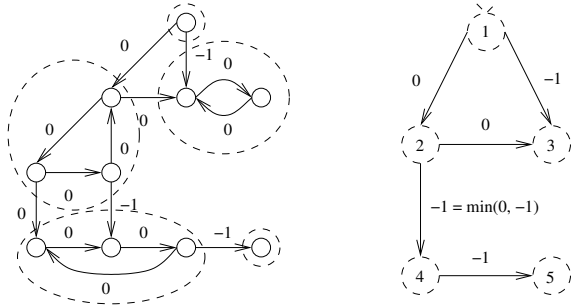


Fig. 3. Obtaining G^{SCC}

a finite diagnosis delay. However, the computation of diagnosis delays in this context has not been addressed before. Moreover, straightforward modifications of the results in the references will doubtfully provide a way to compute the exact diagnosis delay. One may consider constructing a new weighted, directed graph with some counting mechanism and count diagnosis delays with brute force. This direct approach may need the compounded state space for the counter whose size could be up to $|Q^N \times Q^N \times Q^P|$. A more computationally efficient approach would be to apply Bellman-Ford shortest-path algorithm to the weighted, directed graph G with the single source $(q_0^N, q_0^N, q_0^P, \text{non-faulty})$. The negative weight edges are designed to count the extension of faulty traces. Therefore, the absolute value of the minimum of the shortest-path weight is the worst case diagnosis delay. Though this methodology is clear and simple, the computation complexity of running the Bellman-Ford algorithm is $O(VE)$. The weighting structure of G can be exploited to enable an algorithm with a lower complexity, which is $O(V + E)$. If $\{\mathcal{L}(N)\}$ is language-diagnosable with respect to $\mathcal{L}(P)$ and M , all cycles formed in G have zero weight by Theorem 3.1. First we apply an algorithm for finding the strongly connected components and obtain the acyclic component graph G^{SCC} by shrinking each strongly connected component of G to a single vertex. When the vertexes are shrunken, it may be possible to have multiple edges with different weights to another component. We choose the minimum weight for the weight of edges between components. See Fig. 3 for a graphical explanation of this procedure. Now the component graph G^{SCC} is acyclic. We apply an algorithm finding the shortest path in a directed, “acyclic” graph to G^{SCC} with the source vertex as the component that contains $(q_0^N, q_0^N, q_0^P, 0)$. This returns the shortest path from the source component to other components of G^{SCC} . The absolute value of the minimum shortest path value gives the worst case diagnosis delay of $\{\mathcal{L}(N)\}$ with respect to $\mathcal{L}(P)$ and M . See (Cormen *et al.*, 1990) for text book treatments of relevant algorithms.

With this procedure, we can state the computational complexity of obtaining the worst case diagnosis delay as follows.

Theorem 3.4. The worst case diagnosis delay of $\mathcal{L}_f = \{\mathcal{L}(N_1), \dots, \mathcal{L}(N_m)\}$ w.r.t. $\mathcal{L}(P)$ and M can be computed in $O((|Q^{N_1}|^2 + \dots + |Q^{N_m}|^2) \cdot n_2 \cdot n_3^2)$.

ACKNOWLEDGMENT

The research reported in this paper was supported in part by the U.S. Department of Energy under contract W-31-109-Eng-38.

REFERENCES

- Cassandras, C. G. and S. Lafortune (1999). *Introduction to Discrete Event Systems*. Kluwer Academic Publishers.
- Contant, O., S. Lafortune and D. Teneketzis (2002). Failure diagnosis of discrete event systems: The case of intermittent faults. In: *Proc. of CDC 2002, IEEE Conference on Decision and Control*. pp. 4006–4011.
- Cormen, Thomas H., Charles E. Leiserson and Ronald L. Rivest (1990). *Introduction to Algorithms*. The MIT Press.
- Jiang, S. and R. Kumar (2002). Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications. In: *Proc. 2002 Ameri. Contr. Conf.*
- Jiang, S., R. Kumar and H. E. Garcia (2002). Diagnosis of repeated failures in discrete event systems. In: *Proc. of CDC 2002, IEEE Conf. on Decision and Control*. pp. 4000–4005.
- Jiang, S., Z. Huang, V. Chandra and R. Kumar (2001). A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Trans. Automat. Contr.* **46**(8), 1318–1321.
- Sampath, M., R. Sengupta, K. Sinnamohideen S. Lafortune and D. Teneketzis (1995). Diagnosability of discrete event systems. *IEEE Trans. on Automat. Contr.* **40**(9), 1555–1575.
- Sampath, M., S. Lafortune and D. Teneketzis (1998). Active diagnosis of discrete event systems. *IEEE Trans. on Automat. Contr.* **43**(7), 908–929.
- Yoo, T. and H. E. Garcia (2003). Computation of fault detection delay in discrete-event systems. Available from the authors.
- Yoo, T. and S. Lafortune (2002). Polynomial time verification of diagnosability of partially-observed discrete-event systems. *IEEE Trans. Automat. Contr.* **47**(9), 1491–1495.
- Zad, S. H. (1999). Fault diagnosis in discrete-event and hybrid systems. PhD thesis. University of Toronto. Toronto, Canada.