A Model-Based Diagnosis Framework for Distributed Systems[*]

Gregory Provan
Rockwell Scientific Company,
1049 Camino Dos Rios, Thousand Oaks, CA 91360
gprovan@rwsc.com

Abstract

We present a distributed model-based diagnostics architecture for embedded diagnostics. We extend the traditional model-based definition of diagnosis to a distributed diagnosis definition, in which we have a collection of distributed components whose interconnectivity is described by a directed graph. Assuming that each component can compute a local minimal diagnosis based only on sensors internal to that component and knowledge only of its own system description, we describe an algorithm that guarantees a globally sound, complete and minimal diagnosis for the complete system. By compiling diagnoses for groups of components based on the interconnectivey graph, the algorithm efficiently synthesizes the local diagnoses computed in distributed components into a globally-sound system diagnosis using a graph-based message-passing approach.

1 INTRODUCTION

This article proposes a new technique for diagnosing distributed systems using a model-based approach. We assume that we have a system consisting of a set of inter-connected components, each of which computes a local (component) diagnosis.[1] We adopt the structure-based diagnosis framework of Darwiche [8] for synthesizing component diagnoses into globally-sound diagnoses, where we obtain the structure from the component connectivity. Unlike previous approaches that compute diagnoses using the system observations and a system description [8; 10], we transform the component diagnosis synthesis into the space of minimal diagnoses. Assuming that each component can compute a local minimal diagnosis based only on sensors internal to that component and knowledge only of the component system description, we describe an algorithm that guarantees a globally sound, complete and minimal diagnosis for the complete system. This algorithm uses as input the directed graph (digraph) describing the connectivity of distributed components,with arc directionality derived from the causal relation between the the components. Given that real-world graphs of this type are either tree-structured or can be converted to tree-structured graphs, we propose a graph-based message-passing algorithm which passes diagnoses as messages and synthesizes local diagnoses into a globally minimal diagnosis in a two-phase process. By compiling diagnoses for collections of components (as determined by the graph's topology), we can significantly improve the performance of distributed embedded systems. We show how this approach can be used for the distributed diagnosis of systems with arbitrary topologies by transforming such topologies into trees.

One important point to stress is that this approach synthesizes diagnoses computed locally, and places no restriction on the technique used to compute each local diagnosis (e.g., neural network, Bayesian network, etc.), provided that each local diagnosis is a least-cost or most-likely diagnosis. The synthesis approach takes this set of self-diagnosing sub-systems, together with the connectivity of these sub-systems, to compute globally-consistent diagnoses.

The approach presented in this article assumes that all faults are diagnosable (i.e., can be isolated) through a centralized algorithm. We examine whether a distributed approach can diagnose all faults, since a distributed algorithm can isolate faults no better than a centralized algorithm. Issues relating to restricted diagnosability of both centralized and distributed algorithms due to insufficient observable data (e.g., when the suite of sensors is insufficient to guarantee complete diagnosability) are examined in [21].

This article is organized as follows. Section 2 introduces the application model that we use to demonstrate our approach. Section 3 introduces our modeling formalism, and specifies our notion of centralized and distributed model. Section 4 describes how we diagnose distributed models. Section 5 surveys some related work on this topic. We summarize our conclusions in Section 6.

2 IN-FLIGHT ENTERTAINMENT EXAMPLE

Throughout this article we use a simplified example of an

[*]Research supported in part by The Office of Naval Research under contract number N00014-98-3-0012.

[1]Note that one can compute component diagnoses using any method which returns a minimal diagnosis (with respect to a specified minimality criterion).

In-Flight Entertainment (IFE) system. Figure 1 shows the schematic for an IFE system fragment where we have (1) a transmitter module (Tx) that generates 10 movie channels (consisting of both video and audio signals) and 10 audio channels; (2) two area distribution boxes (ADB); and (3) attached to each ADB_i we have two passenger units, P_{i1} and P_{i2}. For ADB j, passenger $i,\ i = 1,\ 2$ has a controller C_{ji} for selecting a video or audio channel, plus an audio unit α_i and video display v_i. Control signal C_{ji} is sent by passenger i to ADB_j and then to the transmitter, which in turn sends an RF signal (RF) to each passenger.

We adopt a notion of causal influence for describing how different components affect the value of a signal as it propagates through the system. For example, the RF signal causally influences the passenger audio and video outputs. In this model the observables are the control signals, plus for passenger i downstream of ADB_j sound (S_{ji}) and video-display (VD_{ji}). We assign a fault-mode to the transmitter and to each ADB and passenger unit.

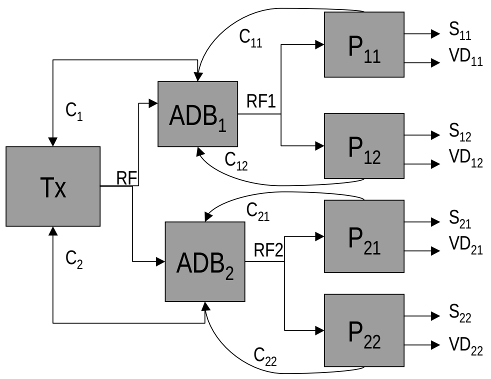

Figure 1: Schematic of IFE fragment, showing the main modules and the directed arcs of data-flows.

Our modeling approach makes the following assumptions. First, we can specify a system using an object-oriented approach. In other words, a system can be defined as a collection of components, which are connected together, e.g., physically, as in an HVAC system, or in terms of data transmission/reception, as in the IFE example. Our primary component consists of a *block*, which has properties: input set, output set, fault-mode, and equations. Given the fault-mode and input set, the equations provide a mapping to the output set. In other words, the inputs are the only nodes with causal arcs into the block, and the outputs are the only nodes with causal arcs out of the block. Typically, we have causal dependence of block outputs ω_i on inputs ℓ_i, *i.e.* $\omega_i \propto \ell_i$.[2]

This distributed model consists of a set of sub-models, or blocks, which may be connected together. In our IFE example, the transmitter block has inputs of control signals C_1 and C_2, and output an RF signal.

Second, we assume that each component computes diagnoses based on data local to the component. We do not place any restrictions on the type of algorithm used to compute the diagnosis, except that the diagnosis be a least-cost diagnosis. We will describe the cost function used by our synthesis algorithm in the following section.

[2]The causal function $\propto$ can be be generalized to include propositions, relations, probabilistic functions, qualitative differential equations, etc. We don't address such a generalization here.

3 MODEL-BASED DIAGNOSTICS USING CAUSAL NETWORKS

This section formalizes our modeling and inference approach to diagnostics and control reconfiguration. We first introduce the model-based formalism, and then extend these notions to capture a distributed model-based formalism.

3.1 FLAT (CENTRALIZED) MODELS

We adopt and extend the model-based representation for diagnosis of Darwiche [8]. We model the system using a causal network:

Definition 1 *A system description is a four-tuple* $\Phi = (\mathcal{V}, \mathcal{G}, \Sigma)$, *where*

- $\mathcal{V}$ *is a set of variables comprising two variable types:* $\mathcal{A}$ *is a set of variables (called assumables) representing the failure modes of the components,* $\mathbf{V}$ *is a set of non-assumable variables (*$\mathbf{V} \cap \mathcal{A} = \emptyset$*) representing system properties other than failure modes;*
- $\mathcal{G}$ *is a directed acyclic graph (DAG) called a causal structure whose nodes are members in* $\mathbf{V} \cup \mathcal{A}$ *and whose directed arcs represent causal relations between pairs of nodes;*
- *and* Σ *is a set of propositional sentences (called the domain axioms) constructed from members in* $\mathbf{V} \cup \mathcal{A}$ *based on the topological structure of* $\mathcal{G}$.

This definition of system description differs from the standard definition (called SD in [22]) only in that we include a graph $\mathcal{G}$ to complement the domain axioms set of failure modes (commonly called COMPS) and non-assumable variables.

The set of non-assumable variables consists of two exclusive subsets: $\mathbf{V}_{obs}$ (the set of observables) and $\mathbf{V}_{unobs}$ (the set of unobservables).

We can capture structural properties of the system description using the directed acyclic graph, or DAG, $\mathcal{G}$.[3] For example, if an actuator determines if a motor is on or not, we say that the actuator causally influences the motor. More generally, A may directly causally influence B if A is a predecessor of B in $\mathcal{G}$. We use $B \propto A$ to denote the direct causal influence of the value of B by the value of A.[4] Through transitivity, we can deduce *indirect* causal influence. For example, if $B \propto A$ and $C \propto B$, then A indirectly influences C.

This captures the notion of direct causal influence, i.e., a node N and those nodes that are directly causally affected by N, using a *clan*. We define the notion of the clan of a node N of a DAG $\mathcal{G}$ in terms of graphical relationships as follows:

[3]In other system description specifications, e.g. [12], these structural relations are captured using logical sentences.

[4]This notion of causal influence does not guarantee that A influences B, but that A *may* influence B.

Definition 2 (Clan) *: Given a DAG $\mathcal{G}$, the clan $Y(N_i)$ of a node $N_i \in \mathcal{G}$ consists of the node N_i together with its children in $\mathcal{G}$.*

We adopt the notion of clan because we are interested in synthesizing diagnoses computed at a set of distributed nodes organized in a tree structure. The intuition behind the algorithm is as follows: given local diagnoses, we start at the parents of leaves in the decomposition tree and move up the tree to the root, identifying if any node's diagnosis is affected by the diagnoses of its children, and if so, synthesizing those diagnoses. To perform each synthesis operation, we use a clan.

A clan is dual to the well-known notion of *family,* which is typically defined as a node together with its parents in $\mathcal{G}$. This notion is important because we need to synthesize local diagnostics within tree-structured systems, and the clan provides a more efficient means for doing so than the family for tree-structured systems. For simplicity of notation, we will denote the clan for node N_i, $Y(N_i)$, as Y_i.

It is also important to define restrictions of subsets of observables:

Definition 3 (Restriction) *We denote by θ_i the restriction of an instantiation θ of variables V to the instantiation of a subset V_i of V. We denote the restriction of variable set T to variables in sub-system description Φ_i by T^{Φ_i}.*

One of the key elements of diagnosing a system is the instantiation of observables, since a diagnosis is computed for abnormal observable instantiations.

Definition 4 (Instantiation) *θ^{Φ_i} is an instantiation of observables ${V_{obs}}^{\Phi_i}$ for system description Φ_i. Θ^{Φ_i} denotes the set of all instantiations of observables ${V_{obs}}^{\Phi_i}$.*

We specify failure-mode instantiations and partition the possible states into normal states and faulty states as follows:

Definition 5 (Mode-Instantiation) *$\mathcal{A}^*$ is an instantiation of behavior modes for mode-set $\mathcal{A}$. Further, we decomposition $\mathcal{A}^*$ such that $\mathcal{A}^* = \mathcal{A}^F \cup \mathcal{A}^\emptyset$, where $\mathcal{A}^\emptyset$ denotes normal system behaviour, i.e. all modes are normal, and $\mathcal{A}^F$ denotes a system fault, which may consist of simultaneous faults in multiple components.*

An assumable (behavior-mode variable) specifies the discrete set of behavior-states that a component can have, e.g., and AND-gate can be either *OK, stuck-at-0,* or *stuck-at-1.* Our IFE-system, with component-set $\{Tx, ABD_1, ADB_2, P_{11}, P_{12}, P_{21}, P_{22}\}$, can have a mode-instantiation in which all components are OK except P_{11}, which is in *audio-fail* mode. In this case we have $\mathcal{A}^\emptyset = \{Tx - mode = OK, ABD_1 - mode = OK, ADB_2 - mode = OK, P_{12} - mode = OK, P_{21} - mode = OK, P_{22} - mode = OK\}$ and $\mathcal{A}^F = \{P_{11} - mode =$*audio-fail*$\}$.

3.2 DISTRIBUTED SYSTEM DESCRIPTIONS

This section describes our distributed formalism, which applies to collections of interconnected components, or blocks. We assume that a distributed system description is provided either by the user or is deduced from the physical constraints of available local diagnostic agents and physical connectivity. For example, many engineering systems, such as commercial aircraft, are subdivided into Line-Replaceable Units (LRUs), based on a number of factors, such as fault-isolation capabilities, physical constraints, and ease of repair. An LRU typically consists of a number of connected sub-systems, as in the Passenger Unit of the IFE example, which consists of circuit-cards to select audio/video channels and to drive the audio and video output devices. It is standard practice in commercial aircraft to isolate faults only to the LRU-level, and replace faulty components only at the LRU-level.

Definition 6 (Decomposition Function) *a decomposition function is a mapping $\psi(\Phi) = \Phi_{dist}$ that decomposes a centralized system description Φ into a distributed system description $\Phi_{dist} = \{\Phi_1, ..., \Phi_m\}$. The distributed system description induced by a decomposition function ψ is defined by a decomposition Π over the system variables $\mathcal{V}$, i.e. a collection $\mathcal{X} = \{X_1, ..., X_m\}$ of nonempty subsets of $\mathcal{V}$ such that (1) $\forall i = 1, ..., m,\ X_i \in 2^{\mathcal{V}}$; (2) $\mathcal{V} = \cup_i (X_i | X_i \in \Pi)$. When $\xi_{ij} = X_i \cap X_j \neq \emptyset$, we call ξ_{ij} the separating set, or sepset, of variables between Φ_i and Φ_j.*

We can describe a distributed system description in terms of a *decomposition graph.* A decomposition graph is a graphical representation of the system model, when viewed as a collection of connected blocks. In this graph each vertex corresponds to a block, and each directed edge corresponds to a directed (causal) link between two blocks. Figure 2 shows the decomposition graph for the extended IFE example.[5]

A *decomposition graph* is a directed tree, or D-tree, which is defined as follows:

Definition 7 *A D-tree $\mathcal{T}_\mathcal{D}$ is a directed graph with vertices $V(\mathcal{T}_\mathcal{D})$ with a vertex r_0, called the root, with the property that for every vertex $r \in V(\mathcal{T}_\mathcal{D})$ there is a unique directed walk from r_0 to r.*

Definition 8 *A decomposition graph $G_\mathcal{X}$ is an edge-labeled D-tree $G(\mathcal{X}, \mathcal{E}, \xi)$ with (1) vertices $\mathcal{X} = \{X_1, ..., X_m\}$, where each vertex consists of a collection of variables of $\mathcal{G}$, (2) directed edges join pairs of vertices with non-empty intersections, and arc direction is specified by the causal direction of the arcs between blocks in the decomposition graph, i.e., $\mathcal{E} = \{(X_j, X_k) | X_i \cap X_j \neq \emptyset,\ X_k \propto X_j\}$, and (3) edge labels (or separators) defined by the edge intersections, $\xi = \{\xi_{ij} | X_i \cap X_j \neq \emptyset\}$.*

We assume that in a distributed system description, for any block all sensor data is local, and all equations describing distributed subsystems refer to local sensor data and local conditions.

3.3 DIAGNOSIS SPECIFICATION

We define the notion of diagnosis as follows:

Definition 9 (Diagnosis) *Given a system description Φ with domain axioms Σ and an instantiation θ of $\mathbf{V}_{obs}$, a diagnosis $D(\theta)$ is an instantiation of behavior modes $\mathcal{A}^F \cup \mathcal{A}^\emptyset$ such that $\Sigma \cup \theta \cup \mathcal{A}^F \cup \mathcal{A}^\emptyset \not\models \bot$.*

[5]We do not show the feedback loops of control requests ($C_1, C_2, C_{11} ..., C_{22}$) since all edges concerning observables can be cut [7].

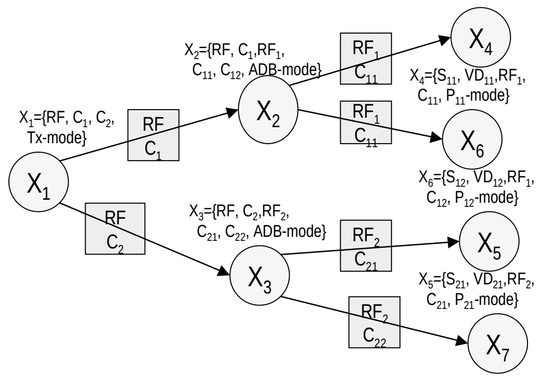
X2={RF, C1,RF1, C11, C12, ADB-mode}
RF1 C11
X4
X4={S11, VD11,RF1, C11, P11-mode}
X1={RF, C1, C2, Tx-mode}
RF C1
X2
RF1 C11
X6
X1
X6={S12, VD12,RF1, C12, P12-mode}
RF C2
X3={RF, C2,RF2, C21, C22, ADB-mode}
RF2 C21
X5
X3
X5={S21, VD21,RF2, C21, P21-mode}
RF2 C22
X7

Figure 2: Decomposition graph of extended IFE system description. Here an oval corresponds to a vertex, and a block corresponds to a sepset. We specify the variables associated with each vertex in the graph.

This diagnostic framework provides the capability to rank diagnoses using a likelihood weight κ_i assigned to each assumable $\mathcal{A}_i,\ i = 1, ..., m$. Using the likelihood algebra defined in [8], we can compute the likelihood assigned to each diagnosis for observation θ. We refer to a (diagnosis, weight) pair using $(D(\theta), \kappa)$. We use the weights to rank diagnoses, i.e., least-weight diagnoses are the most-likely. This provides a notion of *minimal diagnosis*, i.e. a diagnosis of weight κ such that there exists no lesser-weight diagnosis.

3.4 LOCAL/GLOBAL DIAGNOSTICS

Our methodology rests on the determination of when component diagnoses are independent, in which case the global diagnosis is just the conjunction of the component diagnoses. We apply the decomposition theorem of [8] to this case of distributed diagnostics:

Theorem 1 *If we have a system description Φ consisting of two component system descriptions Φ_1 and Φ_2, and a system observation θ, if the variables shared by Φ_1 and Φ_2 all appear in θ, then*

$$D^{\Phi}(\theta) \equiv D^{\Phi_1}(\theta_1) \wedge D^{\Phi_2}(\theta_2).$$

This theorem states that a diagnosis is decomposable provided that the system observation contains the variables shared between Φ_1 and Φ_2. However, what happens when the observation θ does not contain all variables shared between Φ_1 and Φ_2? One solution [8] is to decompose the computation of D^{Φ} by performing a case-analysis of all shared variables ξ_{12}. However, this case-analysis approach is exponential in $|\xi_{12}|$, the number of variables on which we do case-analysis. Hence if we wanted to embed the diagnostics code, such a case-analysis might be too time-consuming when performed on a system-level model.

In the following we assume that each component computes a *local diagnosis*, i.e., a diagnosis based only on local observables and on equations containing only local variables. In contrast a *global diagnosis* is one based on global observables and on equations describing all system variables. Our task is to integrate these local component diagnoses into a globally sound, minimal and consistent diagnosis, since for many systems the diagnostics generated locally are either incomplete or not minimal.

Note that we can obtain global diagnostics for a modular system by composing local blocks and diagnosing the entire system model. However, it is true in many cases that global and local diagnostics may differ. We now define a notion of correspondence between local and global diagnoses.

The conjunction of the set of distributed system descriptions is defined as $D_{dist}(\theta) = \bigwedge_{\Phi_k \in B} D^{\Phi_k}(\theta)$, and we know that $D_{dist}(\theta) = D(\theta)$ only when $\theta \equiv \bigcup i, j\xi_{ij}$.

We can compute the diagnoses for this set of distributed system descriptions either using an on-line algorithm, or by pre-computing the set of diagnoses for $D_{dist}(\theta)$. In the following, we outline the compiled method of diagnosis.

We define a table, called a clan table, to specify local and global diagnoses for collections of blocks. This table compiles the local case-analysis required by Theorem 1. We will show later how to use this table for our diagnosis synthesis algorithm.

Definition 10 *A* **clan (or local/global diagnosis) table** *for block-set $B = \{\Phi_i, ...\Phi_j\}$ is a table consisting of tuples (observable-intantiation, global diagnosis, weight) for all abnormal instantiations of observables θ in B.*

Note that we can use the compositionality of blocks to show that any time we compose a system description from multiple blocks, we obtain "global" diagnostics for that composed system description when we compute diagnoses over the composed system description. Hence the "global" diagnosis for each collection of blocks is computed from a system description generated from the composition of the system descriptions of the blocks in B, using the observables from B.

Example 1 Table 1 contrasts the local and global diagnoses for a set of scenarios where the set B of blocks is an ADB with downstream passenger units. In these scenarios, we compute the (probabilistically) most-likely diagnosis, assuming that all faults are equally likely, i.e., have weight 1. Moreover, in defining a local diagnosis in Table 1, we report the conjunction of all local diagnoses, i.e. the local diagnosis is *ADB-diagnosis* $\wedge$ P_1*-diagnosis* $\wedge$ P_1*-diagnosis*. In scenarios 1, 2 and 4, the local and global diagnoses are identical. However, in scenarios 3, 5 and 6, they differ: the passenger units each assume a local fault, whereas the transmitter unit is the faulty one (since a single transmitter fault is much more likely the two simultaneous faults, one in each passenger unit).[6]

Given this potential for discrepancy between local and global diagnoses, we map the decomposition graph into a representation, the clan graph, from which we can synthesize globally sound and complete minimal diagnoses from local minimal diagnoses. Figure 3 shows the clan graph for the extended IFE example.

[6]These differences arise due to different instantiations of the RF signal in the local and global diagnosis. We hide the details of the case-analysis of shared variables for simplicity of presentation.

Scenario	ADB_1 Unit		Pass. Unit$_{11}$		Pass. Unit$_{12}$		Diagnosis	
	C_{11}	C_{12}	S_{11}	VD_{11}	S_{12}	VD_{12}	LOCAL	GLOBAL
1	audio	audio	nom.	none	nom.	none	–	–
2	audio	audio	**none**	none	nom.	none	P_{11}*-audio-fail*	P_{11}*-audio-fail*
3	audio	audio	**none**	none	**none**	none	P_{11}*-audio-fail*$\wedge$ P_{12}*-audio-fail*	*Xaudio*
4	video	video	nom.	nom.	nom.	**none**	P_{12}*-video-fail*	P_{12}*-video-fail*
5	video	video	nom.	**none**	nom.	**none**	P_{11}*-video-fail*$\wedge$ P_{12}*-video-fail*	*Xvideo*
6	audio	video	**none**	none	**none**.	**none**	P_{11}*-audio-fail*$\wedge$ P_{12}*-video-fail*	ADB_1*-fail*

Table 1: Diagnostic Scenarios. We denote a nominal passenger output of nominal using nom., and abnormal observable data in bold-face. *Xaudio* denotes degraded audio, and *Xvideo* denotes degrated video.

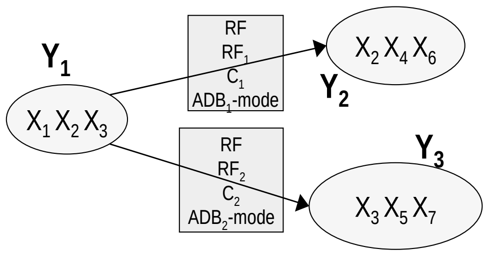

Figure 3: Clan graph of extended IFE system description.

Definition 11 (Clan graph) *: A clan graph $G_{\mathcal{Y}}$ of a DAG $\mathcal{G}(V,E)$ of vertices V and edges E is an edge-labeled D-tree $G(\mathcal{Y},\mathcal{E},\xi)$ defined as follows: (1) vertices $\mathcal{Y} = \{Y_1,...,Y_m\}$, where each node Y_i consists of a clan of $\mathcal{G}$; (2) edges defined by non-empty intersections between pairs of vertices $\mathcal{E} = \{(Y_j,Y_k)|Y_i \cap Y_j \neq \emptyset\}$; and (3) separators defined by the edge intersections $\xi = \{\xi_{ij} = Y_i \cap Y_j\}$.*

The following section shows how we use the clan graph for distributed diagnosis.

4 DISTRIBUTED MODEL-BASED DIAGNOSIS

This section describes our distributed model-based diagnosis algorithm. We first map the directed graph of the system into a tree using tree-decomposition techniques, and then employ a message-passing algorithm on the tree.

4.1 TREE-DECOMPOSITION

The work on tree-decomposition stems from work on treewidth and graph minors [23]. A good review of the literature can be found in [5]. We define the basic notions below.

Definition 12 *A* tree decomposition *of an undirected graph $G = (V,E)$ is a pair $(\mathcal{X},T)$ with $T = (I,F)$ a tree, and $\mathcal{X} = \{X_i | i \in I\}$ is a family of subsets of V, one for each node of T, such that*

1. *$\bigcup_{i \in I} X_i = V$;*
2. *for all edges $\{v,w\} \in E$ there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$, and*
3. *for all $i,j,k \in I$ if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.*

The last property is known as the running-intersection property within the BN community. The clique-tree algorithm computes a tree-decomposition in which each node of the tree is a clique, and undirected edges correspond to shared variables between cliques.

Given a tree-decomposition, inference complexity is based on the treewidth, defined as follows. The *width* of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G. The treewidth bears close relations to the maximal vertex degree and maximal clique of a graph, so it provides a measure of the complexity of diagnostic inference, among other things. If a graph has a low treewidth then inference on the graph is guaranteed to be easy. The task of computing treewidth is NP-hard [2]. Many algorithms exist that, given a graph with n variables, will compute an optimal treewidth in time polynomial in n but exponential in the treewidth k; see, for example, [4].

Directed Tree-Decomposition

The difference between the standard literature on tree-decompositions and the task addressed here is that the standard literature focuses on undirected graphs, and we focus on directed graphs. We capture and exploit the directionality of causal relations during all phases of diagnostic inference. For example, if we have an abstract hierarchical specification of a system and compute diagnostics for each abstract hierarchical block, we still preserve the directionality of causality among the abstract blocks. We exploit this directionality using a diagnostic synthesis algorithm operating on a *directed* tree.

Definition 13 *A D-tree $\mathcal{T}_{\mathcal{D}}$ is a directed graph with vertices $\mathcal{V}_{\mathcal{T}_{\mathcal{D}}}$ and a vertex V_0, called the root, with the property that for every vertex $V \in \mathcal{V}_{\mathcal{T}_{\mathcal{D}}}$ there is a unique directed walk from V_0 to V.*

The tree-decomposition results have been generalized to directed graphs in [16], and we make use of some of those results here. The key change is that we need to preserve ordering of edges during the decomposition process. To capture such properties, we first need to define a notion of variable ordering, called Z-normality.

Definition 14 *Let $\mathcal{G}$ be a digraph and let $Z \subseteq \mathcal{V}$. A set S is Z-normal if and only if the vertex-sets of the strong components of $\mathcal{G} \setminus Z$ can be numbered $S_1, S_2, ..., S_d$ such that*

1. *if $1 \le i \le j \le d$, then no edge of $\mathcal{G}$ has a head in S_i and tail in S_j, and*
2. *either $S = \emptyset$ or $S = S_i \cup S_{i+1} \cdots \cup S_j$ for some integers i,j with $1 \le i \le j \le d$.*

Definition 15 *A* D-tree decomposition *of a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair $(\mathcal{X}, \mathcal{T_D})$ with $\mathcal{T_D} = (\mathcal{I}, \mathcal{F})$ a D-tree, and $\mathcal{X} = \{X_i | i \in \mathcal{I}\}$ is a family of subsets of $\mathcal{V}$, one for each node of $\mathcal{T_D}$, and the edges are numbered $\mathcal{J} = \{1, ..., l\}$ with $\mathcal{F} = \{F_j : j \in \mathcal{J}\}$, such that*

1. $\bigcup_{i \in \mathcal{I}} X_i = \mathcal{V}$;
2. *for all edges $\{v, w\} \in \mathcal{E}$ there exists an $i \in \mathcal{I}$ with $v \in X_i$ and $w \in X_i$, and*
3. *for all $i, j, k \in \mathcal{I}$ if j is on the path from i to k in $\mathcal{T_D}$, then $X_i \cap X_k \subseteq X_j$;*
4. *if $j \in \mathcal{J}$, then $\bigcup_i \{X_i : i \in \mathcal{I}, i > j\}$ is X_j-normal.*

The *width* of a tree decomposition is the least integer w such that for all $i \in \mathcal{I}$, $|X_i \cup \bigcup X_j| \leq w + 1$, where the union is taken over all edges $j \in \mathcal{J}$ incident with i. $\max_{i \in \mathcal{I}} |X_i| - 1$. The *treewidth* of a graph $\mathcal{G}$ is the least integer w such that $\mathcal{G}$ has a D-tree-decomposition of width w.

For the class of applications addressed in this article, the input graphs $\mathcal{G}$ for the system description are digraphs, and the decomposition graph and clan graph are both D-tree decompositions of $\mathcal{G}$. For more general digraph topologies, by applying an algorithm for generating D-tree decompositions, we can convert the digraphs into a decomposition graph, and apply the diagnostic synthesis approach. Many of the properties of undirected tree-decompositions hold for the directed case [16].

4.2 DIAGNOSIS OF SYSTEMS WITH TREE-STRUCTURED GRAPHS

We now describe an approach to diagnosing systems with tree-structured decomposition graphs.

We assume that:

- We are provided with the component system descriptions and their connectivity;
- There is a single root in the decomposition graph (which is a component with no parent-components), and each leaf is a component with no child-component;
- Nodes have indices starting at the root (X_1), increasing based on a breadth-first expansion from the root and ending at the leaves, labeled $X_{n-s}, ..., X_n$;
- Each component computes a local diagnosis based on local observables.

We base our approach on synthesizing diagnoses, starting from the leaf components and ending up at the root of the tree. We first decompose the decomposition graph into a clan graph. Based on the clan graph we construct a clan table for each node in the graph.

This algorithm is inspired by the Bayesian network clique-tree approach of [17], but replaces the clique-tree with an analogous clan-tree, and passes diagnoses as messages. Analogous to the clique-tree method's clique-table pre-computation, this approach requires pre-computing clan-tables, but for embedded systems this results in computationally simpler algorithms than those adopted in the past.

Under this scheme, we pre-compute clan tables for each clan in $\mathcal{G_Y}$. Given an observation θ for blocks $X_i, ..., X_k$, where $X_i, ..., X_k$ are members of a clan $Y \in \mathcal{G_Y}$, each block computes diagnostics locally. We then compute the most likely fault-mode assignment for Y through a process we call *diagnostics synthesis*, which entails table-lookup in the clan table of the minimal diagnosis given θ. The algorithm synthesizes final diagnoses, going from the leaves to the root. This guarantees a sound, complete and globally minimum system diagnosis.

In this approach we first need to pre-compute the clan table, and then use that table for diagnostic synthesis. We can pre-compute the clan table from a set of blocks $\{\Phi_1, ..., \Phi_k\}$ as follows:

1. Generate the decomposition graph $G_{\mathcal{X}}$ from $\{\Phi_1, ..., \Phi_k\}$, with indices increasing in a breadth-first manner from the root.
2. Generate the clan graph $G_{\mathcal{Y}}$ of $G_{\mathcal{X}}$.
3. Compute the clan table for each clan Y_i in $G_{\mathcal{Y}}$.

Given an observation θ, the diagnostic synthesis algorithm is as follows:

```
1. Given observation $\theta$, each block $B_i$ computes its local
   diagnosis $D^{\Phi_i}(\theta)$ and likelihood $\kappa(D^{\Phi_i})$.
2. Mark all nodes $X_i$, $i = 1, ..., n$ with flag=0;
3. Loop for $j = n$ to 1:
   (a) If flag=0 for $X_j$ do:
       For each node $X_i$ in the clan $Y(X_j)$, look up
       corresponding clan diagnosis $D^{\Phi_Y}(\theta)$ and weight
       $\kappa(D^{\Phi_Y}(\theta))$ in the clan-table;
           If $\kappa(D^{\Phi_Y}(\theta)) < \sum_{k:\Phi_k \in Y} \kappa(D^{\Phi_k})$,
       • revise fault-mode assignment to nodes in $Y(N_j)$,
         by (a) setting the minimum-weight diagnosis
         mode-variable; (b) if any local diagnosis $D'$ is
         synthesized, update $D'$
       • reassign values to variables in $Y$ based on $D$ and
         $\theta$
       • if reassignment is sound pass message with fault
         report $D^{\Phi_Y}(\theta)$.
       • Set flag for all $X_i \in Y(X_j)$ to 1;
```

Theorem 2 *Given a tree-structured decomposition graph $\mathcal{G_X}$ and local component diagnoses, diagnostics synthesis will compute a sound and globally consistent set of fault mode assignments for components $\mathcal{X} \in \mathcal{G_X}$ within $O(|\mathcal{Y}|)$ message-passing steps, where $\mathcal{G_Y}$ is the clan graph generated from $\mathcal{G_X}$.*

Example 2 Diagnosis Synthesis in a Clan: Consider Scenario 3 of Table 1. For this observation θ, the total set of possible clan diagnoses is: (P_{11}, *audio-fail*) $\wedge$ (P_{12}, *audio-fail*) $\vee$ (ADB_1, *Xaudio*). The weights of the diagnoses are 2 and 1, respectively.

In computing diagnoses on a purely local basis, the resulting diagnosis is (P_{11}, *audio-fail*) $\wedge$ (P_{12}, *audio-fail*), with weight 2. Note however there is a family diagnosis of weight 1, (ADB_1, *Xaudio*), which is selected since it is of lower weight than the distributed diagnosis. We now instantiate each local component with θ, and set diagnoses as follows: (P_{11}, $\emptyset$), (P_{12}, $\emptyset$), (ADB_1, *Xaudio*). There exists a consistent set of local variable instantantiations for this assignment, so no further message-passing is necessary.

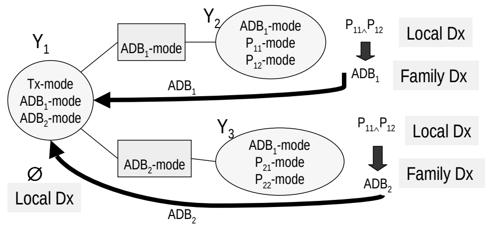

Figure 4: Diagnosis synthesis procedure, Step 1: (a) local diagnoses synthesized at clans, and (b) clan diagnoses are passed between families, as noted by dark arrows.

Example 3 Message-Passing: Figure 4 shows the first stage of this procedure. In the graph we show nodes where the variables are restricted to fault mode variables, to simplify the description of message-passing of instantations of mode variables. First, the local diagnoses are computed at each node in the decomposition graph: all four passenger units register a fault, and no other nodes in the decomposition graph register faults. As a shorthand, we denote a fault-weight pair using variable-names for faults, with $\emptyset$ denoting a nominal mode. Then, these faults are synthesized at each clan using the clan-table: fault-weight pair $(P_{11} \wedge P_{12},\ 2)$ is synthesized into $(ADB_1,\ 1)$, and fault $(P_{21} \wedge P_{22},\ 2)$ is synthesized into $(ADB_2,\ 1)$. Second, the synthesized faults $(ADB_1,\ 1)$ and $(ADB_2,\ 1)$ are sent to the adjacent node in the clan graph, Y_1.

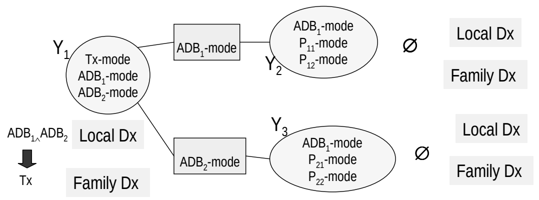

Figure 5: Diagnosis synthesis procedure, Step 2: global diagnoses computed following family diagnosis message-passing.

Figure 5 shows the second stage of this procedure. Fault-weight pair $(ADB_1 \wedge ADB_2,\ 2)$ is synthesized into $(Tx,\ 1)$ at clan Y_1, and all other fault-modes are set to nominal. This is the global minimum-weight fault.

4.3 COMPLEXITY ISSUES

The complexity of logical resolution within a distributed framework have been discussed in [1]. Here, our task is model-based diagnosis within a tree-structured topology.

This approach is based on computing diagnoses for the clans of $\mathcal{G}$. Hence, it never needs to diagnose a system description for the entire graph $\mathcal{G}$, but only for the clans of $\mathcal{G}$. As noted in Theorem 2, once the clan tables are computed, given any local component diagnoses, the algorithm is linear in the number of nodes in the clan-graph.

The worst-case complexity of computing a clan table is exponential in the number of variables in the clan table. The memory requirements for storing the clan tables are defined as follows. In the worst case, for a clan with mode variables $\mathcal{A}_1, ..., \mathcal{A}_m$, where each mode variable has $|\omega_{A_i}|$ faulty values, a clan table stores an entry for each of the $\times_i |\omega_{A_i}|$ multiple-fault combinations. For single-fault scenarios, a clan table must store only $\sum_i |\omega_{A_i}|$ entries.

The main issue is the time-complexity of generating the clan tables. For tree-structured systems the complexity of diagnosing $\mathcal{G}$ is exponential in the clan size, and the complexity is bounded by the largest clan of $\mathcal{G}$. Hence the complexity of initially computing diagnoses is the same for the centralized and distributed approaches. However, for embedded applications, the distributed approach has a complexity advantage, since only clan-table lookup and simple message-passing are required.

5 RELATED WORK

Our approach to distributed diagnosis has been preceded by many pieces of related work, and we review several here. Note that this review examines the most relevant work, and does not claim to be exhaustive.

One of the most closely-related pieces of work describes techniques for distributed logical inference [1; 20]. This work focuses on how to perform logical reasoning and query answering, proposing sound and complete message passing algorithms, by exploiting the tree structure of distributed theories. They examine the complexity of computation, propose specialized algorithms for first-order resolution and focused consequence finding, and propose algorithms for optimally partitioning a theory that is not already distributed. In some ways, our task can be considered a special case of the general problem that Amir and McIlraith examine. Logical inference computes a model, whereas diagnostic inference computes a *minimal* model in the assumables, a subset of the language of the theory. We leverage many aspects of the specific diagnosis problem in our work, aspects that serve to distinguish both our approach and our results. These include the notion of causality, which imposes a directionality on the tree structure and the inference, and the notion of preference. In addition, the task of diagnostic inference depends critically on two classes of distinguished variables, assumables (the literals of interest) and observables (the inputs), and distributed diagnosability depends on how assumables and observables are distributed among the collection of blocks. In addition, if the variables common between two blocks are observable, then from a distributed diagnostics point of view those blocks are independent [7].

The approach presented here bears some relation to diagnostic approaches on trees. Stumptner and Wotawa [25] have an algorithm for diagnosing tree-structured systems. This approach assumes a centralized system defined at the component level whereas our approach deals with distributed systems that can be defined at any level of abstraction. In addition, our assumption of sub-systems computing their own diagnoses means that our diagnostic synthesis process is a single-pass algorithm from the leaves of the tree to the root, whereas Stumptner and Wotawa need a two-pass approach since they must first enumerate all component diagnoses. A second major tree-based method uses a clique-tree decomposition of a system, e.g., the diagnostic method of [13]. A clique-tree is a representation that is used for many kinds of inference in addition to diagnosis, including probabilistic inference and constraint satisfaction. The tree we generate is a directed tree with a fixed root, and the nodes of the tree are generated based on the clan property; a clique-tree is undirected (with an arbitrary root), and the nodes of the tree are generated based on the family property. One can think of the D-tree as a directed variant of a clique-tree, which is optimized for diagnostic inference. In addition, our approach uses the ordering of the D-tree to require message-passing in a single direction only; in contrast, message propagation in clique trees is bi-directional.

Our work also bears some relation to papers describing distributed solutions to Constraint Satisfaction Problems (CSPs) [26; 15]. As with the work on distributed logical inference [1], the task of distributed CSPs is finding a satisfying assignment to the variables, when constraints are distributed in a collection of subsets of constraints. Hence the underlying tasks of distributed diagnosis and CSP satisfiability are different. One issue in this work that is similar to diagnostic reasoning is the recording of minimal sets of unsatisfiable clauses as nogoods [15]. The computation of nogoods is a key step to computing diagnoses [10].

There have been several proposals for using the ATMS [9] in a distributed manner, e.g., [11; 19; 3; 18]. Our approach differs from this work in that our approach uses system topology explicitly, whereas these other approaches do not make as extensive a use of topology.

The compilation approach proposed in this article bears some relation to prior work.[7] [24] presents an empirical comparison of centralized compilation techniques as applied to several areas, of which diagnosis is one. Our future work includes examining the applicability of these compilation techniques within our distributed framework. Compilation is also examined in [20], but (as mentioned earlier) as applied to a different task, logical resolution.

There has been some prior work on distributed model-based diagnosis. For example, the approach in [14] assumes that the diagnosis computed by each distributed agent is globally correct, and examine the case where agents must cooperate to diagnose components whose status is unknown. Our approach makes the more realistic assumption that diagnoses are not necessarily globally sound, and derives a very different global synthesis algorithm.

6 SUMMARY AND CONCLUSIONS

This document has described a mechanism for computing distributed diagnoses using system topology and observability properties. This algorithm takes as input minimal diagnoses computed within distributed components, and uses system topology to integrate these diagnoses into a globally sound and minimal system diagnosis.

We are in the process of applying this approach to two real-world domains, that of In-Flight Entertainment and diagnosis of HVAC systems.

The approach presented here provides a mechanism for designing systems with predictable distributed diagnostics properties. A given decomposition graph can be rated according to its diagnosability and efficiency. Additionally, given a system description, we can apply D-tree decomposition algorithms to the system DAG to assist in identifying small-treewidth decompositions, if any exist. Further, if a system has no small treewidth decomposition, one can then recommend system re-design to be facilitate efficiently computing distributed diagnoses.

[7]A review of compilation can be found in [6].

References

[1] E. Amir and S. McIlraith. Paritition-based logical reasoning. In *Proc. KR '2000*, pages 389–400. Morgan Kaufmann, 2000.

[2] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Meth.*, 8:277–284, 1987.

[3] C. Beckstein, R. Fuhge, and G. Kraetzschmar. Supporting assumption-based reasoning in a distributed environment. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 3–17, Hidden Valley, Pennsylvania, 1993.

[4] Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[5] H. Bodlander. Treewidth: Algorithmic techniques and results. In *Proceedings 22nd International Symposium on Mathematical Foundations of Computer Science, MFCS'97*, volume 1295 of Lecture Notes in Computer Science, pages 29–36. Springer-Verlag, 1997.

[6] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

[7] A. Darwiche and G. Provan. Exploiting system structure in model-based diagnosis of discrete-event systems. In *Proc. 7th Intl. Workshop on Principles of Diagnosis*, pages 95–105, 1996.

[8] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

[9] J. de Kleer. An Assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[10] J. de Kleer and B. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1987.

[11] A. Dragoni. Distributed belief revision versus distributed truth maintenance: preliminary report. In *Atti del 3zo Incontro del Gruppo AI*IA di Interesse Speciale su Inteligenza Artificiale Distribuita*, pages 64–73, Rome, Italy, 1993.

[12] O. Dressler and Peter Struss. The consistency-based approach to the automated diagnosis of devices. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 267–311. CSLI Publications, Stanford, CA, USA, 1996.

[13] Yousri El Fattah and Rina Dechter. Diagnosing tree-decomposable circuits. In *IJCAI*, pages 1742–1749, 1995.

[14] Peter Frohlich, Iara de Almeida Mora, Wolfgang Nejdl, and Michael Schroeder. Diagnostic agents for distributed systems. In *ModelAge Workshop*, pages 173–186, 1997.

[15] K. Hirayama and M. Yokoo. The effect of nogood learning in distributed constraint satisfaction. In *Proceedings of the 20th IEEE International Conf. on Distributed Computing Systems*, pages 169–177, 2000.

[16] T. Johnson, N. Robertson, P. Seymour, and R. Thomas. Directed tree-width. *to appear in J. Combin. Theory Ser. B.*

[17] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *Royal Statistical Society*, 50:154–227, 1988.

[18] Benedita Malheiro and Eugenio Oliveira. Solving conflicting beliefs with a distributed belief revision approach. In *IBERAMIA-SBIA*, pages 146–155, 2000.

[19] Cindy L. Mason and Rowland R. Johnson. DATMS: A framework for distributed assumption based reasoning. In Les Gasser and Michael N. Huhns, editors, *Distributed Artificial Intelligence*, volume 2, pages 293–317. Pitman, 1989.

[20] Sheila A. McIlraith and Eyal Amir. Theorem proving with structured theories. In *Proc. IJCAI*, pages 624–634. Morgan Kaufmann, 2001.

[21] G. Provan. Distributed Diagnosability Properties of Discrete Event Systems. In *Proc. American Control Conference*, Anchorage, AK, May 2002.

[22] R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–96, 1987.

[23] N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of treewidth. *J. Algorithms*, 7:309–322, 1986.

[24] Laurent Simon and Alvaro del Val. Efficient consequence finding. In *IJCAI*, pages 359–370, 2001.

[25] Markus Stumptner and Franz Wotawa. Diagnosing tree-structured systems. *Artificial Intelligence*, 127(1):1–29, 2001.

[26] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *Knowledge and Data Engineering*, 10(5):673–685, 1998.