NOVEL METHODOLOGY FOR PARTITIONING COMPLEX SYSTEMS FOR FAULT DIAGNOSIS PURPOSES

C. D. Bocaniala[1] and J. Sa da Costa[2]

[1]*University "Dunărea de Jos" of Galati, Computer Science and Engineering Dept.*
Domneasca 47, Galati 6200, Romania
Cosmin.Bocaniala@ugal.ro

[2]*Technical University of Lisbon, Instituto Superior Tecnico, Dept. of Mechanical Engineering, GCAR/IDMEC*
Avenida Rovisco Pais, Lisboa 1096, Portugal
sadacosta@dem.ist.utl.pt

Abstract: The criterion used for partitioning is d-*separation*: a parallel between causal independency property and vertex separation in digraphs. If X, Y and Z represent the vertex subsets of two neighbouring partition regions and respectively the border between them, the d-*separation* criterion is used to decide if "knowing Z renders Y irrelevant to X". It follows that diagnosis may be performed locally, inside each region, without communicating, via partition borders, with other regions. If borders are affected by faults, communication is needed. The described partitioning provides minimal borders between regions. It follows that communication process has minimal computational complexity. *Copyright © 2005 IFAC*

Keywords: Distributed models, Fault diagnosis, Feedback loops, Graph theoretic models, Minimization.

1. INTRODUCTION

Fault detection and isolation (FDI) methodologies use actuators and sensors measurements. When dealing with complex (large-scale) industrial installations, designing a fault diagnosis system becomes very difficult due to the large number of sensors and actuators. Any solution given to this problem must take into account the fact that practitioners prefer rather simplistic systems due to the fact that, in practice, simple and verifiable principles always win the competition versus complex methods that are usually characterized by instability, unpredictable behaviour and large computational burden (Patton, 1997). The distributed diagnosis framework described in this paper is able to achieve its goal by using simple and verifiable principles coming mainly from causal modelling and distributed computing. The main component of the framework is the methodology for partitioning the monitored system.

There are two main approaches in performing distributed fault diagnosis. One possible approach is to define a partition on the system structure and to assign one agent to each element of the partition. Each agent performs local diagnosis inside the area they are assigned to. Global diagnosis is obtained by defining a proper communication scheme among agents. This approach has been implemented for instance by Letia *et. al.* (2000), Fabre *et. al.* (2001), the DIAMOND project (Albert *et. al.*, 2001), and Koscielny (2004). The distributed diagnosis framework described in this paper aligns with this first approach.

The other possible approach to distributed fault diagnosis is to bring together the diagnosis expertise of different methodologies (the agents). Isermann and Ballé (1997) underline the fact that a single diagnosis method is inadequate for matching all challenges posed by a complex system. The analysis may be performed at the whole system level or at a lower level, i.e. taking into account subsystems and/or even single components such as a single sensor or a single actuator. In this case, the complexity burden is much larger as a central supervisor is needed to determine first the level of analysis and second the panel of methodologies used for each considered subpart of the system. A recent implementation of this approach is MAGIC project (Köpen-Seliger et. al., 2003; Lesecq et. al., 2003).

The methodologies mentioned above lack a coherent methodology that partitions the monitored system into a set of subsystems such that the independence level of local diagnosis process for each subsystem is maximal and such that the communication between different subsystems, required for formulating global diagnosis, is minimal. The described methodology fulfils the previous conditions as follows. It partitions the monitored system into *fully* independent subsystems. It also insures minimal borders between different subsystems which imply minimal communication. An important contribution of the paper is that it provides the described methodology with theoretical support.

The content of the paper is organized as follows. Section 2 briefly sketches the distributed fault diagnosis framework inside which the described partitioning methodology is used. Section 3 presents the feedback loops replacement methodology that allows a system model with feedback to be transformed into a model without feedback, yet preserving the temporal information encoded by the replaced feedback loops. This is needed as the partitioning methodology may be applied only on system models with no feedback. Section 4 brings in the methodology used to build the partitions of the monitored systems. Last section, Section 5, summarizes the paper contributions and gives a possible future research direction.

2. DESCRIPTION OF THE DISTRIBUTED DIAGNOSIS FRAMEWORK

The complexity of a system resides in the number of its basic components, actuators and sensors. The causal model of a system may be encoded as a directed graph (digraph) where vertices represent the available actuators and sensors readings, and edges represent the causal links between these measurements. The complexity of the system is reflected in the complexity of the associated digraph. The described distributed fault diagnosis framework basically (*i*) considers the causal model of the system as a map, (*ii*) partitions this map into edge disjoint regions separated by borders formed by vertices, and (*iii*) assigns a dedicated agent to each region (Fig. 1). For step (*ii*), notice that each region may be treated recursively in the same manner as the initial map, therefore inducing a local hierarchy of agents. The local expertise of the agents, as well as the interaction between them is used to robustly detect and isolate the faults in the system. The use of this distributed scheme allows maintaining the focus only on those regions of the map that are affected by faults. Hence, monitoring a complex system becomes a tractable problem.

In order to comply with the natural requirement for a as small as possible diagnosis computational time, the previous partitioning is required to satisfy next conditions: (*i*) the agents should be able to independently asses the state of the system in the assigned area, and (*ii*) the interaction between different agents should be kept as small as possible. The complexity of the interaction between two agents is given by the number of vertices located on the borders between the corresponding regions.

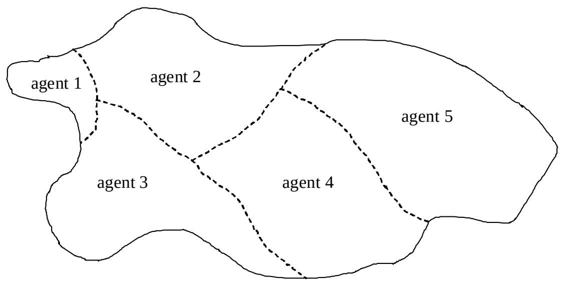

Fig. 1. Partitioning the causal model of a system

The first condition is fulfilled by using the d-*separation* criterion, introduced in (Pearl and Paz, 1985; Pearl and Verma, 1986), to split the map in separate regions. The criterion offers a parallel between the causal independency and the vertex separation in digraphs. If *X*, *Y* and *Z* represent three vertex subsets in a causal model, the d-*separation* criterion is able to determine if "knowing *Z* renders *Y* irrelevant to *X*". For the proposed partitioning, if *X* and *Y* represent the vertex subsets of two neighbouring regions, and if *Z* represents the vertex subset that constitutes the border between the two regions, then the d-*separation* criterion always holds, i.e. the regions *X* and *Y* are causally independent.

An important drawback of the d-*separation* criterion is that it can be applied exclusively on acyclic digraphs. That is why it is needed a methodology that allows cyclic causal models to be transformed into acyclic models. The transformation needs to be carried out without actually losing the structural and behavioural information given by feedback. This methodology is described in Section 3.

The second condition is fulfilled by using the multilevel hypergraph partitioning (Karypis, 2002). The multilevel partitioning paradigm is based on a very simple idea. First, the original hypergraph undergoes a sequence of successive approximations that represent smaller and smaller sized versions of the original configuration until the hypergraph is reduced to a few tenths of vertices. This is called the *coarsening* phase. At this point, some algorithms are used to compute a partitioning of the current form of the hypergraph. This second phase is called the initial *partitioning* phase. The final phase is to use the partitioning of the smallest hypergraph to derive the partitioning of the original hypergraph by successive projections of the current partition to the next level finer approximation of the original hypergraph. The last phase is called the *uncoarsening and refinement* phase.

The analyzed causal model is transformed into a hypergraph so that the following equivalence holds: the causal model has a minimal number of vertices on the partition borders if and only if the equivalent hypergraph has a minimal number of hyperedges cut by the partition borders.

The previous multilevel partitioning hypergraph algorithm has been implemented by its authors into an application called *hMeTiS*. The application, together with a User Manual, can be downloaded from http://www-users.cs.umn.edu/~karypis/hmetis/index.html.

3. FEEDBACK LOOPS REPLACEMENT METHODOLOGY

The section describes a feedback loops replacement methodology used to obtain an acyclic causal model from a cyclic causal model. The most important property of the obtained acyclic causal model is that it reflects not only the structural properties of the original cyclic causal model, but also its behaviour in time. The *initial model* is represented as a digraph where vertices stand for the sensor measurements at the initial time-step of the analysis, and edges stand for cause-effect relationships between them. In order to reflect the behaviour of the system in time, this initial model is replicated at each time step, i.e. when new sensor measurements are available. The vertices of the new replica correspond to the values of the sensor measurements at the current time-step. New edges, which reflect cause-effect relationships between vertices in the current replica of the model and vertices in the previous replicas, must be added. As it is detailed later in the section, adopting models built in the previous manner, offers the opportunity to replace a feedback loop of the system with an acyclic substructure by unfolding it in time. However, all structural information encoded by the cyclic model and all temporal information given by feedback are preserved. It is to be noticed that, as the number of the considered time-steps increase, some vertices of aged replicas of the initial model become causally irrelevant to the other vertices in the model and, therefore, they can be eliminated. The interval of time for which the causal dependencies between older vertices and newer vertices are relevant is called the *relevant time-window span* (Bocaniala, 2004). Thus, the model is dynamic in both positive and negative sense, i.e. vertices may be added and vertices may be eliminated as well.

The first subsection presents an algorithm that always provides an edge cut set for the feedback loops in a cyclic causal model. The algorithm uses the distribution of feedback loops on levels given by the level partitioning (Viswanadham *et al.*, 1987). On the basis of the algorithm in the first subsection, the second subsection presents the algorithm for building the acyclic causal model of a cyclic causal model.

3.1 The minimal edge cut set of a feedback loop

Viswanadham *et al.* (1987) describe in their book an algorithm for structuring a digraph based on the *reachability* relation on the digraph vertex set. The reachability relation R is defined as follows. Given two vertices v_i and v_j, $v_i\ R\ v_j$ if and only if there is a directed path from v_i to v_j. Structuring a digraph with respect to the reachability relation actually builds a partition on the vertex set into equivalence classes called *levels*. Balakrishnan (1997) defines a *strongly connected component* (SCC) of a digraph as a maximal set of interconnected feedback loops. It follows that the set of the SCCs of a causal model concentrates the whole feedback structure of the model. The level partitioning algorithm insures that there is only one SCC per level, i.e. the maximum possible number of SCCs equals the number of levels. Therefore, given the level partitioning of a cyclic causal model, the task of finding an edge cut set that breaks all loops in the system reduces to finding an edge cut set for each SCC given by the level partitioning.

In the following, the algorithm that always provides an edge cut set for a SCC is given. The edge cut set will be required to be *minimal* in the sense that, if possible, each loop is cut on only one edge. Notice that there may be cut edges that break more than one loop. The most favourable situation is when the number of this kind of edges is maximal. The algorithm that computes the *minimal edge cut set* (MECS) for a SCC uses the breadth-first search (BFS) procedure when traversing the SCC. The MECS for the whole causal model is the reunion of the MECS computed for all its SCCs.

Algorithm 1 (The minimal edge cut set of a SCC)

Step 1. Choose randomly one vertex r in the SCC and consider it the root of the BFS tree. Build the BFS tree.

Step 2. An edge that does not belong to the BFS tree is called a left-out edge. For each layer of the BFS tree, for each vertex v on that layer, for each left-out edge e originating from v do the following.

Step 2.1. Check all directed paths containing v and e if (*i*) do not contain any edge in the MECS, and if (*ii*) contain at least an ancestor w of v in BFS tree. If the previous two conditions are satisfied, then there is at least one loop, i.e. the loop containing v, e and w, which is not yet cut. By adding edge e to MECS this loop, which contains v, e and w, and possibly other loops will be cut by e.

Step 2.2. Check if MECS remains minimal after adding e and eliminate the redundant cut edges. An edge from MECS is called redundant if the loops that it cuts are already cut by other edges from MECS.□

Theorem 1 Given a cyclic causal model, Algorithm 2 provides always a minimal edge cut set for each SCC.

Proof First of all, notice that each loop in the considered SCC contains at least one left-out edge. The justification is immediate. The BFS tree from Step 1 is acyclic. If the left-out edges are added to this tree then the obtained graph is the original SCC. The loops in original SCC have been “restored” by adding the left-out edges. It follows that MECS represents a subset of the left-out edges set. What is left to be proven is that the MECS provided by

Algorithm 1 really cuts all loops in the SCC and that it is minimal in the defined sense.

Let denote by *BFS(t)* the BFS tree with vertex *t* as root. Notice that, if the edge *e* in Step 2.1 of the algorithm is *v→u*, all directed paths containing *v* and *e* represent directed paths in *BFS(u)*. Using this observation, Step 2 may be interpreted as follows: if there is an ancestor *w* of *v* in *BFS(r)* from Step 1, such that *w* belongs to *BFS(u)* and such that the directed path between root *u* and *w* in *BFS(u)* does not contain any edge from MECS, then edge *e* is added to MECS. If each directed path in *BFS(u)* between *u* and one of its ancestors *w* in *BFS(r)* contains an edge *f* from MECS, then the loop containing *v*, *e* and *w* and possibly other loops are already cut by *f*. The previous discussion proves that, if there is any loop that contains edge *e* and that it is not yet cut by other edge in MECS, this loop will be cut by adding *e* to MECS in Step 2.1. It follows that MECS will cut all loops in the considered SCC. Moreover, MECS is already minimal in the sense that an edge enters MECS if and only if a loop not yet cut is detected. What is left to be investigated, so that MECS is minimal in the sense defined at the beginning of the subsection, is the elimination of redundant edges from Step 2.2.

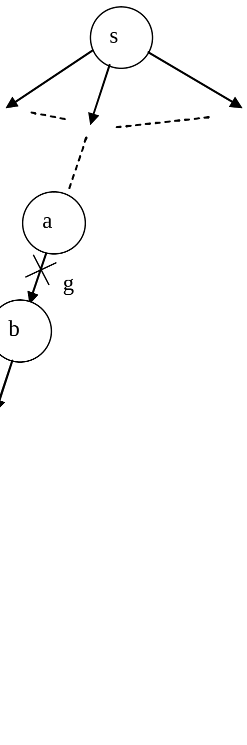

Fig. 2. Two possibly redundant edges in *BFS(s)* tree

Let denote by *AN(v)* the ancestors of *v* in *BFS(r)* and by *EL(u)* (from *eliminated*) all vertices *s* in *BFS(u)* such that the directed path between *u* and *s* is cut by an edge from MECS. Then the condition for edge *e* to enter MECS may be expressed as

$$\begin{aligned}&\forall e \text{ left-out edge from SCC},\\&e = v \to u,\ v \in BFS(r) \qquad (1)\\&e \in MECS \Leftrightarrow AN(v) \cap \left(BFS(u) - EL(u)\right) \neq \varnothing\end{aligned}$$

The redundant cut edges mentioned in Step 2.2 may appear in a *BFS(s)* tree, *s≠r*, as shown in Fig. 2. The directed path from *s* to *d* contains both edges *g* and *h*. The condition *s≠r* is given as both *g* and *h* represent left-out edges and, by definition, *BFS(r)* does not contain any left-out edge. As detailed above, edges *g* and *h* are cut with the purpose of disconnecting *a* and respectively *c* from the vertices in *AN(a)* respectively *AN(c)*. When both *g* and *h* appear on the directed path from *s* to *d* in the *BFS(s)* tree, *s≠r*, the fact that they are cut may be interpreted as disconnecting *a* and respectively *c* from the vertices in *AN(a)* ∩ *SubBFS(s,b)* respectively *AN(c)* ∩ *SubBFS(s,d)*, where *SubBFS(s,t)* represents the subtree of *BFS(s)* having the root *t*. If edge *g* is fixed and for any edge *h* and any vertex *s*

$$\begin{aligned}&(1)\ g \text{ and } h \text{ belong to the path between}\\&s \text{ and } d \text{ in } BFS(s) \qquad (2)\\&(2)\ AN(a) \cap SubBFS(s,b) \subseteq AN(c) \cap SubBFS(s,d)\end{aligned}$$

then *g* may be eliminated from MECS in Step 2.2. It follows that MECS is minimal in the sense defined at the beginning of the subsection.□

Corollary 1 Given a cyclic causal model, there is always a minimal edge cut set (MECS) that renders the causal model acyclic.

Proof Theorem 1 insures that there is always a MECS for each SCC of a cyclic causal model. It follows that the reunion of these MECS, i.e. the MECS of the cyclic causal model, always exists and it renders acyclic the initial cyclic causal model.□

3.2 Transforming a cyclic causal model into an acyclic model by feedback loops unfolding in time

Given the algorithm in the previous subsection, it is now possible to give an algorithm that computes the acyclic causal model of a cyclic causal model by performing feedback loop unfolding in time. The algorithm must be provided with the relevant time-window span constant c_{max} (see the introductory part of section).

Algorithm 2 (Feedback loops unfolding in time for obtaining an acyclic causal model corresponding to a cyclic causal model)

Step 1. If the analyzed causal model is cyclic, then first obtain the initial model (see the introductory part of this section) by eliminating the minimal edge cut set (MECS) from the cyclic causal model.

Step 2. Let *t* be the initial time-step. If S_t is an element of the initial model, then its instance at the *i*-th time-step, *i=1*, …, c_{max}, is noted as S_{t+i*T}. The possible connections in the final acyclic model are detailed in the following.

Step 2.1. All vertices S_{t+j*T}, $0 \leq j < i$, will have an outgoing connection with S_{t+i*T}.

Step 2.2. If U_t is another element of the initial model, $U_t \neq S_t$, so that S_t and U_t are connected in the initial model, then all pairs S_{t+i*T} and U_{t+i*T} will have the same type of connection.

Step 2.3. Finally, for each edge $U \rightarrow S$ or $S \rightarrow U$ in MECS, the connection $U_{t+(i-1)*T} \rightarrow S_{t+i*T}$ or respectively $S_{t+(i-1)*T} \rightarrow U_{t+i*T}$ is added to the model.□

Theorem 2 Each vertex in the acyclic causal model obtained by applying Algorithm 2 to a cyclic causal model, receives all input values that it is supposed to receive and provides all output values that it is supposed to provide.

Proof The proof represents an analysis of Algorithm 2. First, the connections between vertices at the *i*-th step must be identical with the connections that exist in the initial model. This is insured by Step 2.2. The loss of connectivity information caused by the feedback loop replacement is recovered via unfolding in time, Step 2.3.□

Corollary 2 The acyclic causal model obtained by applying Algorithm 2 to a cyclic causal model preserves all structural information and all temporal information given by the initial cyclic causal model.

Proof It is an immediate consequence of Theorem 2.□

4. PARTITIONING METHODOLOGY

This section presents the algorithm that performs the proposed partitioning. The number *k* of regions must be decided by the user. The decision must take into account the fact that the whole set of vertices is going to be distributed inside each region of the partition as well as on the borders of the partition. The goal is to obtain a partition that (*i*) has a minimal vertex-cut set and that (*ii*) has all pairs of neighbouring regions causally independent (d-*separated*). The uncertainty of this decision consists in the fact that the algorithm used guarantees minimal borders, but neither it is able to estimate the number of vertices located on them nor it is able to estimate how many vertices belong to each partition member. Future research need to find methodologies able to eliminate this uncertainty. One possible direction is to insert principles from algorithms that provide minimal d-*separation* sets (Tian *et. al.*, 1998) into multilevel partitioning algorithm.

Algorithm 3 (Partitioning a causal model into minimally separated and causally independent regions)

Step 1. If the input causal model *CM* contains feedback loops, use Algorithm 2 to perform feedback loops replacement in order to obtain the corresponding acyclic causal model (*ACM*).

Step 2. Compute the *moral* graph *MG* corresponding to *ACM*. The moral graph of an acyclic digraph is built by connecting first all pairs of vertices that are parents of the same vertex and, then, giving up edge orientation (Lauritzen *et. al.*, 1990). The "*morality*" of the obtained graph is insured by the fact that all vertices that share a child vertex are now "*married*" by connecting edges.

Step 3. Transform the *MG* graph into a hypergraph *HG* so that (i) the edges of *MG* represent the vertices of *HG* and (ii) each hyperedge *h* of *HG* corresponds to a vertex *v* in *MG* as follows,

$$h = \{e \in MG \ / \ e \text{ is an incoming/outgoing edge in/from } v\} \tag{3}$$

Step 4. Use the *hMeTiS* application, with the *k* parameter decided by the user, to partition *HG* into *k* parts. For more details see (Bocaniala, 2004).

Step 4.1. The vertex-cut set in *MG* corresponds to the hyperedge-cut set of *HG*.

Step 4.2. The regions in the *MG* partition are delimited using the edge labelling of *MG* provided by the *HG* partition. The vertex-cut set on *MG* determines a partition of *ACM* into causally independent regions.□

Theorem 3 Each hyperedge-cut set in *HG* has a correspondent vertex-cut set in *MG* of the same size.

Proof When partitioning *HG* using *hMeTiS* in Step 4 of Algorithm 3, each hyperedge *h* in *HG* may or may not be cut by the provided partition. In the following these two possible situations are analyzed.

If a hyperedge *h* in *HG* is cut by the HG partition provided by Step 4, this fact has the following interpretation. The elements in *h* span more then one region of the *HG* partition. But, the elements in *h* are all edges in MG with one end in a vertex *v* from *MG* (Eq. 3). Since the partition regions in *MG* are determined by the edge labelling provided for *HG* (Step 4.2), it follows that the edges in *MG* with one end in *v*, span more than one region of the *MG* partition. It follows that *v* represents a vertex located on the borders of the partition in MG.

If a hyperedge *h* in *HG* is not cut by the *HG* partition provided by Step 4, this fact has the following interpretation. The elements in *h* span one single region of the *HG* partition. But, the elements in *h* are all edges in *MG* with one end in a vertex *v* from *MG* (Eq.3). Since the partition regions in *MG* are determined by the edge labelling provided for *HG* (Step 4.2), it follows that the edges in *MG* with one end in *v*, span one single region of the *MG* partition. It follows that *v* represents a vertex located inside one of the partition regions of MG.

From the previous two analyses, it results that each hyperedge *h* in the hyperedge-cut set provided by Step 4 has a corresponding vertex *v* in the edge-cut set of *MG*. It follows that the claim in the theorem text is true, i.e. each hyperedge-cut set in *HG* has a correspondent vertex-cut set in *MG* of the same size.□

Corollary 3 Given a causal model of a system, Algorithm 3 provides a partition of its acyclic form that (*i*) has a minimal vertex-cut set and that (*ii*) has all pairs of neighbouring regions causally independent, i.e d-*separated* by the minimal vertex-cut set.

Proof This corollary is an immediate consequence of Theorem 3. The hyperedge-cut set of *HG* provided by *hMeTiS* is minimal (Step 4). Theorem 3 proved that the vertex-cut set in *MG* induced by the hyperedge-cut set in *HG* have the same cardinal. It follows that the induced vertex-cut set in *MG* is also minimal.

One vertex set separating two regions in *MG* will d-*separate* the two regions in the acyclic form *ACM* of the original causal model *CM* (Lauritzen *et. al.*, 1990). It follows that the vertex-cut set induced on *MG* by the hyperedge-cut set in *HG* will d-*separate* each pair of neighbouring regions in *ACM*.□

5. CONCLUSIONS

The paper described a novel methodology for partitioning complex system for fault diagnosis. The methodology partitions the causal model associated with the monitored system into minimally vertex-separated and causally independent (d-*separated*) regions.

The fact that each region is causally independent by the rest of the model allows performing the diagnosis of that region locally, without needing to communicate with the rest of the model. This property allows maintaining the diagnosis focus exclusively on those regions of the map that are affected by faults. Hence, monitoring a complex system becomes a tractable problem.

Moreover, if the causal independence property is affected by faults, the communication between different regions needed has a minimal computational complexity. This is due to the fact that communication takes place via partition borders, which are minimal.

The proposed partitioning methodology has the potential to be used for fault-tolerant control purposes. Fault-tolerant control is concerned with making a controlled system able to maintain control objectives, despite the occurrence of a fault. For instance, if faults that produce structural changes that do not require system shutdown occur, the causal model of the system and the associated distributed fault diagnosis system also suffer modifications and may be updated by re-partitioning using the same algorithm.

REFERENCES

Albert, M., Längle, T., Wörn, H., Kazi, A., Brighenti, A., Senior, C., Revuelta Seijo, S., Sanz Bobi, M. A., Villar, J. (2001). Distributed architecture for monitoring and diagnosis, EU ESPRIT Project DIAMOND. http://www.ipr.ira.uka.de/~~kamara/diamond.

Balakrishnan, V.K. (1997). *Graph theory, Schaum's Outlines*. McGraw-Hill, New York.

European Community's FP4, COPERNICUS project (http://www.eng.hull.ac.uk/research/control/Copernicus/contentscop2.htm).

Fabre, E., Benveniste, A., Jard, C. (2002). Distributed diagnosis for large discrete events in dynamic systems. In: *Preprints of the 15th IFAC World Congress*, Barcelona, Spain.

Isermann, R., Ballé, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice* **5(5)**, 709-719.

Karypis, G. (2002). *Multilevel Hypergraph Partitioning*, Technical Report 02-25, Department of Computer Science and Engineering, University of Minnesota, USA.

Koscielny, J. (2004). Diagnostics of industrial processes in decentralised structures, In: *Fault Diagnosis. Models, Artificial Intelligence, Applications* (J. Korbicz, J. M. Koscielny Z. Kowalczuk and W. Cholewa. (Ed)). Springer.

Köpen-Seliger, B., Marcu, T., Capobianco, M., Gentil, S., Albert, M., Latzel, S. (2003). MAGIC: An integrated approach for diagnostic data management and operator support. In: *Proceedings of the IFAC Symposium SAFEPROCESS'03*, 187-192. Washinton, USA.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N., Leimer, H. G. (1990). Independence properties of directed Markov fields, *Networks* **20**: 409-505.

Letia, I. A., Craciun, F., Kope, Z., Netin, A. (2000). Distributed diagnosis by BDI agents. In: *Proceedings of the OASTED International Conference on Applied Informatics*.

Lesecq, S., Gentil, S., Exel, M., Garcia-Beltran, C. (2003). Diagnostic tools for a multi-agent monitoring system. In: *Proceedings of IMACS IEEE CESA Multi-Conference on Computing Engineering in Systems Applications*, Lille, France.

Patton, R. J. (1997). Fault-tolerant control: The 1997 situation. In: *Proceedings of the IFAC Symposium SAFEPROCESS'97,* 1033-1055, Hull, UK.

Pearl J. and Paz A. (1985). *Graphoids: A Graph-Based Logic for Reasoning about Relevance Relationships*, Technical Report CSD-850038, Computer Science Department, Cognitive Systems Laboratory, University of California, Los Angeles, USA.

Pearl J. and Verma T. (1986). *Formal Properties of Probabilistic Dependencies and their Graphical Representations*, Technical Report CSD-860019, Computer Science Department, Cognitive Systems Laboratory, University of California, Los Angeles, USA.

Tian, J., Verma T. and Pearl J. (1998). *Finding Minimal d-Separators*, Technical Report CSD-980007, Computer Science Department, Cognitive Systems Laboratory, University of California, Los Angeles, USA.